

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-13^{*}

©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 label and ref	11
1.3	User interface	3	5.1.2	Keys for wrap and display	11
1.3.1	Internal counters	3	5.1.3	Keys for debug and checking	11
1.3.2	Support for multicol	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for minipage	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 keyval system	6	7	The way of non-enumerated lists	19
4.1	Keys for label and ref	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 add code	8	12	Index of Implementation	126
4.4	Keys for start, series and resume	9			
4.5	Keys for multicol	9			
4.6	Keys for minipage	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 multicol, vertical alignment

^{*}This file describes a documentation for v1.0, last revised 2024-06-13.

[†]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 / 139

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

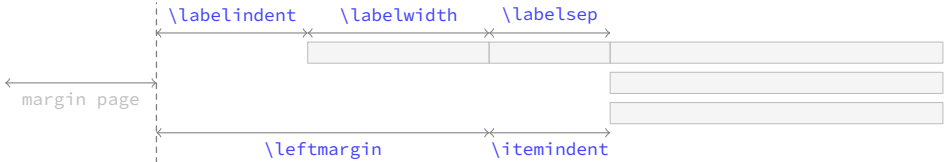


Figure 1: Representation of horizontal lengths in `enumitem`.

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

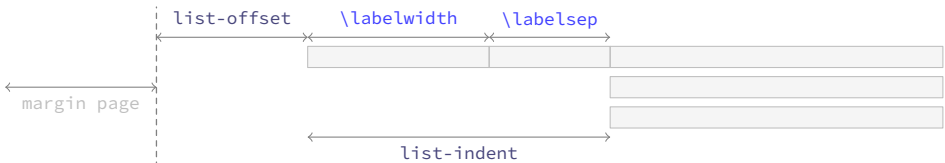


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

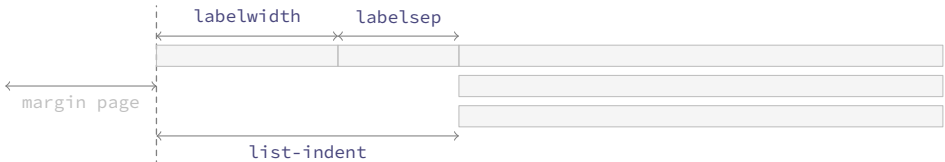


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Support for multicols

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.

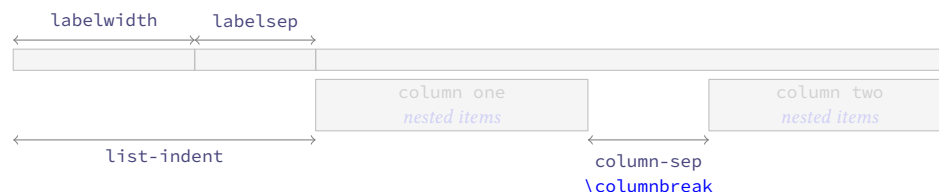


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 L^AT_EX, `\item` and `\item[\langle custom \rangle]` commands work in the usual way. The environment can be nested with at most “four levels” and the options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Example with columns=2

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

2.2 The environment enumext*

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

Some considerations to take into account for this environment:

- The environment cannot be nested within itself, 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 `\itemwidthdim` 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.
2. This text is in the first level.
- X This text is in the first level.
- ★ 3. This text is in the first level.

2.3 The command \item*

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

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` works like the numbered `\item`, but placing a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the 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 |
| 8. The tenth | two columns | | |

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

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

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

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

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

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

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

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={\store name}]{\begin{enumext}[save-ans={\store name}]}
  \item Text \anskey{answer}      \item Text \anskey{answer}
  \item Text                      \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{\store name: position}`, where $\langle position \rangle$ corresponds to the position occupied by the “*stored content*” in the $\langle store name \rangle$ returned by the `show-pos` key. For example `\ref{test:4}` will return `3`. (b) which corresponds to the location of the “*stored content*” at position `4` within the environment in which the key `save-ans=test` was set.

`mark-ref = { \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>(a) Question.</p> <p>[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 content \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.

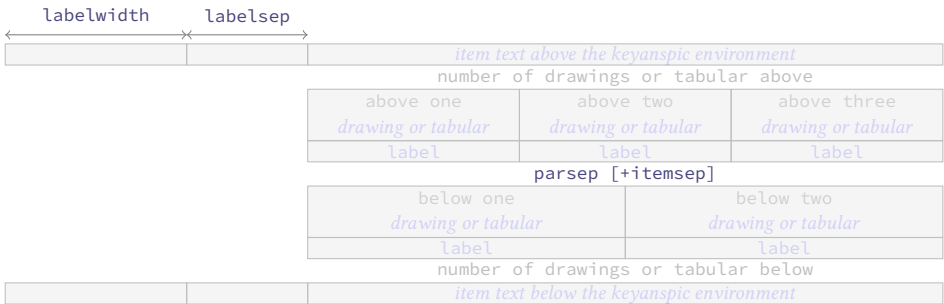


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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* `{\store name}` set by `save-ans` key.


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


Example


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


1. Question with images.


A)


B)


C)


D)


* E)[note]

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

B

B

A



D

E

5. Question with image on left side:

A

value

B

value

C

value

D

correct

E

value



Test keys

1. B, $x = 5$

2. D

3. C, some note

4. E, A duck


5. D, other note
- *


*

*


*

Example 4

A “simple worksheet” using ducks :) .

- 

Factor $x^2 - 2x + 1$



Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)



True False

- (a) $\alpha > \delta$
- (b) $\mathbb{E}\mathbb{T}\mathbb{E}\mathbb{X}$ 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 \labelbx { s +m }
{%
  \IfBooleanTF{#1}
  {\strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}}%
  {\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}}%
}

```

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

long

SoMeThInG A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit,

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] MITTELBAACH, 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 2024-06-13 – First public release.

10 Index of Documentation

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

C

Document class:

article2

book2

exam2

letter2

report2

\columnbreak4, 12

\columnsep10

Commands provide by enumext:

\anskey11-13

\anspic11, 12, 15

\getkeyans12, 16

\item*5-7, 11, 12, 14

\item5-7, 9, 10, 12, 14

\miniright10

\printkeyans6, 11, 16

\setenumext5-7, 11, 12, 14, 16

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

anskey*11-13

enumext*4-14, 16

enumext4-9, 11-14, 16, 19

keyans*4-14

keyanspic4, 6, 8, 11-13, 15, 19

keyans4-9, 11-15, 19

Environments:

enumerate1, 3, 5, 21

figure5

list3, 9, 21

minipage3-5, 10, 21

multicols3, 4, 10

table5

task5

F

\footnote5

I

\itemsep8

K

Keys for command provide by enumext:

break-col12

item-join12

item-pos*13

item-star12, 13

item-sym*13

Keys for environments provide by enumext:

above*8

above8

after9, 10

align7, 20

base-fix8

before*9

before9

below*8

below8

check-ans12

columns-sep4, 10

columns4, 8, 10

first9

font7

item-pos*5, 6

item-sym*5, 6

itemindent9

itemsep8, 15

labelsep3, 5-7, 9, 10, 12, 20

labelwidth3, 6, 7, 9, 10, 12, 20

labelwith5

label7, 9, 14, 19, 20

list-indent3, 9

list-offset3, 9, 20

listparindent9

mark-ans12

mark-pos12

mark-ref11

mini-env4, 8, 10

mini-right*6, 10

mini-right6, 10

mini-sep4, 10

no-store11-13

noitemsep8

nosep8, 19

parsep8, 15

partopsep8

ref4, 7

resume*6, 10, 11

resume6, 9-11

rightmargin9

save-ans4, 6, 9-16

save-key10, 11, 16

save-ref4, 7, 11-13, 16

save-sep11

series6, 9-11

show-ans11, 12

show-length7

show-pos11, 12, 16

start9, 10

topsep8

widest7

wrap-ans12

wrap-label*7, 20

wrap-label7

wrap-opt12

L

\label4

Labels provide by enumext:

\Alph*7, 14

\Roman*7

\alph*7

\arabic*7

\roman*	7	l3seq	1, 21
\labelsep	3, 7	multicol	1, 2, 4, 21
\labelwidth	3, 7	scontents	1, 2, 13
\linewidth	10	task	5, 6
\listparindent	9	xsim	2
P		\parsep	8
Packages:		\partopsep	8
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   {2024-06-13}
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:` (§11.6.1).

```
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` (§11.10) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.13).

```
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 (§11.13).

```
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 (§11.24).

```
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` (§11.14) and `label` (§11.12) 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 (§11.17). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:N` (§11.37.1).

```

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 §11.21).

```

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 (§11.22.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.20, §11.22).

```

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 (§11.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§11.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.12). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§11.19).

```

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 (§11.25.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§11.29) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the $\langle\textit{body}\rangle$ and the $\langle\textit{keys}\rangle$ of the environment `anskey*` (§11.30).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§11.35.2) and `\anspic*` (§11.40.1) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```
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` (§11.46).

```
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` (§11.45), `show-pos` key (§11.26), `item-sym*` key (§11.33), `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 (§11.40.2).

```
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
```

Internal variables used by “*internal check answer*” mechanism (§11.25.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
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

```

(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 (§11.9). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

146 \bool_new:N \l__enumext_hyperref_bool
147 \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` (§11.9) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

148 \tl_new:N \l__enumext_newlabel_arg_one_tl
149 \tl_new:N \l__enumext_newlabel_arg_two_tl
150 \tl_new:N \l__enumext_write_aux_file_tl
151 \cs_set_protected:Npn \__enumext_tmp:n #1
152 {
153   \tl_new:c { l__enumext_label_copy_#1_tl }
154 }
155 \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` (§11.34).

```

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

```

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

```

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

```

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

```

159 \cs_set_protected:Npn \__enumext_tmp:n #1
160 {
161   \bool_new:c { l__enumext_item_starred_#1_bool }
162   \int_new:c { l__enumext_item_column_pos_#1_int }
163   \int_new:c { g__enumext_item_count_all_#1_int }
164   \int_new:c { l__enumext_joined_item_#1_int }
165   \int_new:c { l__enumext_joined_item_aux_#1_int }
166   \int_new:c { l__enumext_tmpa_#1_int }
167   \dim_new:c { l__enumext_tmpa_#1_dim }
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 Public dimension

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

```
185 \dim_zero_new:N \itemwidthdim
```

11.6 Some utility functions

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

```
186 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
187 {
188   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
189 }
```

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

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

`__enumext_before_env:nn`

```
190 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
191 {
192   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
193 }
194 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
195 {
196   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
197 }
```

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

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

```
198 \cs_new:Nn \__enumext_level:
199 {
200   \int_to_roman:n { \__enumext_level_int }
201 }
```

(End of definition for `__enumext_level:.`)

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

`__enumext_if_is_int:nF`
`__enumext_if_is_int:nTF`

```
202 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
203 {
204   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
205   { \prg_return_true: }
206   { \prg_return_false: }
207 }
```

(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.

```
208 \cs_new_protected:Nn \__enumext_regex_counter_style:
209 {
210   \tl_map_inline:Nn \c__enumext_counter_style_tl
211   {
212     \regex_replace_once:nnN { \c{##1}\* }
213     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
214   }
215 }
```

(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.

```

216 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
217 {
218   * ~ #2
219   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
220   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
221 }

```

(End of definition for _enumext_show_length:nnn.)

11.6.1 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.

```

222 \cs_new_protected:Nn \__enumext_is_not_nested:
223 {
224   \str_case:en { \@currentenv }
225   {
226     {enumext}
227     {
228       \bool_lazy_and:nnT
229       { \bool_not_p:n { \g__enumext_standar_bool } }
230       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
231       {
232         \bool_gset_true:N \g__enumext_standar_bool
233       }
234     }
235     {enumext*}
236     {
237       \bool_lazy_and:nnT
238       { \bool_not_p:n { \g__enumext_starred_bool } }
239       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
240       {
241         \bool_gset_true:N \g__enumext_starred_bool
242       }
243     }
244   }
245 }

```

The function _enumext_is_on_first_level: will set the variables \\l__enumext_standar_first_bool (§11.25.1), \\l__enumext_starred_first_bool (§11.25.1) and \\l__enumext_anskey_env_bool (§11.30) to “*true*” only if the environment is not nested and we are in the “*first level*” of it . We will also save the *start line number* of each environment in the variable \\g__enumext_start_line_tl and the *name* of each environment in the variable \\g__enumext_envir_name_tl to use in messages related to the `check-ans` key and `.log` file.

```

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

```

```

271         \bool_set_true:N \l__enumext_anskey_env_bool
272         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
273         \tl_gset:Ne \g__enumext_start_line_tl
274         {
275             on ~ line ~ \exp_not:V \inputlineno
276         }
277     }
278 }

```

(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.

```

279 \cs_new_protected:Nn \__enumext_keyans_start_line:
280 {
281     \str_case:en { \@currentenvir }
282     {
283         {keyans}
284         {
285             \tl_set:Ne \l__enumext_check_start_line_env_tl
286             {
287                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
288             }
289         }
290         {keyans*}
291         {
292             \tl_set:Ne \l__enumext_check_start_line_env_tl
293             {
294                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
295             }
296         }
297         {keyanspic}
298         {
299             \tl_set:Ne \l__enumext_check_start_line_env_tl
300             {
301                 in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
302             }
303         }
304     }
305 }

```

(End of definition for __enumext_keyans_start_line:.)

11.6.2 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:

```

306 \cs_new_protected:Nn \__enumext_reset_global_vars:
307 {
308     \__enumext_reset_global_int:
309     \__enumext_reset_global_bool:
310     \__enumext_reset_global_tl:
311 }
312 \cs_new_protected:Nn \__enumext_reset_global_int:
313 {
314     \int_gzero:N \g__enumext_item_number_int
315     \int_gzero:N \g__enumext_item_anskey_int
316     \int_gzero:N \g__enumext_item_answer_diff_int
317 }
318 \cs_new_protected:Nn \__enumext_reset_global_bool:
319 {
320     \bool_gset_false:N \g__enumext_check_ans_key_bool
321     \bool_gset_false:N \g__enumext_standar_bool
322     \bool_gset_false:N \g__enumext_starred_bool
323 }
324 \cs_new_protected:Nn \__enumext_reset_global_tl:
325 {
326     \tl_gclear:N \g__enumext_store_name_tl
327     \tl_gclear:N \g__enumext_start_line_tl
328     \tl_gclear:N \g__enumext_envir_name_tl
329 }

```


(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.

```

330 \cs_new_protected:Nn \__enumext_log_global_vars:
331 {
332   \msg_log:nneeee { enumext } { prop-seq-int-hook }
333   { \g__enumext_store_name_tl }
334   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
335   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
336   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
337 }

```

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.

```

338 \cs_new_protected:Nn \__enumext_log_answer_vars:
339 {
340   \msg_log:nneeee { enumext } { item-answer-hook }
341   { \int_use:N \g__enumext_item_number_int }
342   { \int_use:N \g__enumext_item_anskey_int }
343   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
344 }

```

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

11.7 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` 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.

```

345 \__enumext_at_begin_document:n
346 {
347   \cs_new_eq:NN \__enumext_start_list:nn \list
348   \cs_new_eq:NN \__enumext_stop_list: \endlist
349   \cs_new_eq:NN \__enumext_item_std:w \item
350 }

```

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

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

```

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

```

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

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

```

351 \__enumext_at_begin_document:n
352 {
353   \cs_new_eq:NN \__enumext_minipage:w \minipage
354   \cs_new_eq:NN \__enumext_endminipage: \endminipage
355 }

```

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

11.8 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 `\if@minipage` 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 (§11.38) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.42)

```

356 \cs_new_protected:Nn \__enumext_internal_mini_page:
357 {
358   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
359   {
360     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
361     {
362       \__enumext_minipage:w [ t ] { ##1 }
363       \legacy_if_gset_false:n { @minipage }
364       \vspace { 0pt }
365     }
366     { \__enumext_endminipage: }
367   }
368 }
```

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

11.9 Compatibility with hyperref and footnotehyper

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

```

369 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
370 \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”.

```

371 \cs_new_protected:Nn \__enumext_after_hyperref:
372 {
373   \IfPackageLoadedTF { hyperref }
374   {
375     \msg_info:nnn { enumext } { package-load } { hyperref }
376     \bool_set_true:N \l__enumext_hyperref_bool
377     \IfHyperBoolean{hyperfootnotes}
378     {
379       \typeout{hyperfootnotes=true}
380       \bool_set_true:N \l__enumext_footnotes_key_bool
381     }
382     { \typeout{hyperfootnotes=false} }
383   }
384   { }
```

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`.

```

385   \bool_if:NT \l__enumext_footnotes_key_bool
386   {
387     \IfPackageLoadedTF { footnotehyper }
388     {
389       \msg_info:nnn { enumext } { package-load } { footnotehyper }
390     }
391     {
392       \typeout{No ~ footnotehyper ~ load}
393       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
394       \bool_set_false:N \l__enumext_footnotes_key_bool
395     }
396   }
```

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.

```

397 \bool_if:NTF \l__enumext_hyperref_bool
398 {
399     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
400     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
401 }
402 {
403     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
404     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
405 }
406 }

```

(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.

```

407 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
408 {
409     \protected@write \@auxout { }
410     {
411         \token_to_str:N \newlabel {#1}
412         {
413             {#2}
414             \bool_if:NT \l__enumext_hyperref_bool
415             { { \thepage } {#2} {#1} }
416             { }
417         }
418     }
419     \__enumext_hypertarget:nn {#1} { }
420     \__enumext_phantomsection:
421 }

```

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

11.10 Definition of counters

`__enumext_define_counters:Nn` 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.

```

422 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
423 {
424     \cs_if_exist:cTF { c@ #2 }
425     { \msg_fatal:nnn { enumext } { counters } { #2 } }
426     {
427         \tl_set:Nn #1 { #2 }
428         \newcounter { #2 }
429     }
430 }

```

(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    431 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii   432 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii  433 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv   434 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXv    435 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXvi   436 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXvii  437 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXviii 438 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

11.11 Definition of labels

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

`__enumext_register_counter_style:Nn`

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

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

```
449 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
450 {
451   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
452   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
453 }
454 \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.

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

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

11.12 Setting keys associated with label

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
471 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
472 {
473   \keys_define:nn { enumext / #1 }
474   {
475     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
476     font      .value_required:n = true,
477     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
478     labelsep  .initial:n   = {0.3333em},
479     labelsep  .value_required:n = true,
480     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
481     labelwidth .value_required:n = true,
```

```

482     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
483     wrap-label .initial:n = {##1},
484     wrap-label .value_required:n = true,
485     wrap-label* .code:n = {
486         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
487         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
488     },
489     wrap-label* .value_required:n = true,
490 }
491 }
492 \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.

```

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

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

```

(End of definition for align.)

11.13 Setting label and ref keys

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

11.13.1 Define and set label and ref keys for enumext environment

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.

```

label
ref
\__enumext_label_i_tl
\__enumext_label_ii_tl
\__enumext_label_iii_tl
\__enumext_label_iv_tl
541 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
542 {
543   \keys_define:nn { enumext / #1 }
544   {
545     label .code:n = {
546       \__enumext_label_style:cnv { \__enumext_label_#2_tl }
547       { \__enumext_counter_#2_tl } {##1}
548       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
549       \__enumext_current_widest_dim
550     },
551     label .initial:n = #3,
552     label .value_required:n = true,
553     ref .code:n = \__enumext_standar_ref:n {##1},
554     ref .value_required:n = true,
555   }
556 }
557 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
558 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
559 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
560 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }
```

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

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

```

561 \cs_new_protected:Npn \__enumext_standar_ref:n #1
562 {
563   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
564   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
565   {
566     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
567   }
568   {
569     \tl_set_eq:Nc
570     \__enumext_ref_the_count_tl { \__enumext_counter_ \__enumext_level: _tl }
571     \__enumext_regex_counter_style:
572     \tl_set_eq:Nc
573     \__enumext_ref_the_count_tl { \__enumext_the_counter_ \__enumext_level: _tl }
574     \tl_put_right:ce { \__enumext_renew_the_count_ \__enumext_level: _tl }
575     {
576       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
577       { \exp_not:V \__enumext_ref_key_arg_tl }
578     }
579   }
580 }
```

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

```

581 \cs_new_protected:Npn \__enumext_standar_ref:
582 {
583   \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
584   {
585     \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
586   }
587 }
```

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

11.13.2 Define and set label and ref keys for enumext* and keyans* environments

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

```

\l__enumext_label_vii_tl 588 \cs_set_protected:Npn \l__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 589 {
590   \keys_define:nn { enumext / #1 }
591   {
592     label .code:n = {
593       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
594       { l__enumext_counter_#2_tl } {##1}
595       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
596       \l__enumext_current_widest_dim
597     },
598     label .initial:n = #3,
599     label .value_required:n = true,
600     ref .code:n = \l__enumext_starred_ref:n {##1},
601     ref .value_required:n = true,
602   }
603 }
604 \l__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
605 \l__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

(End of definition for label and others.)

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

```

606 \cs_new_protected:Npn \l__enumext_starred_ref:n #1
607 {
608   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
609   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
610   {
611     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
612     {
613       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
614     }
615     {
616       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
617       \__enumext_regex_counter_style:
618       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
619       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
620       {
621         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
622         { \exp_not:V \l__enumext_ref_key_arg_tl }
623       }
624     }
625   }
626   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
627   {
628     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
629     {
630       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
631     }
632     {
633       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
634       \__enumext_regex_counter_style:
635       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
636       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
637       {
638         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
639         { \exp_not:V \l__enumext_ref_key_arg_tl }
640       }
641     }
642   }
643 }

```

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.

```

644 \cs_new_protected:Nn \__enumext_starred_ref:
645 {
646   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
647   {

```

```

648     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
649     {
650         \tl_use:N \l__enumext_renew_the_count_vii_tl
651     }
652 }
653 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
654 {
655     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
656     {
657         \tl_use:N \l__enumext_renew_the_count_viii_tl
658     }
659 }
660 }

```

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

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

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

```

\__enumext_label_v_tl
\__enumext_label_vi_tl
661 \keys_define:nn { enumext / keyans }
662 {
663     label .code:n = {
664         \__enumext_label_style:cnv { \__enumext_label_v_tl }
665         { \__enumext_counter_v_tl } {#1}
666         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
667         \__enumext_current_widest_dim
668         \__enumext_label_style:cnv { \__enumext_label_vi_tl }
669         { \__enumext_counter_vi_tl } {#1}
670         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
671         \__enumext_current_widest_dim
672     },
673     label .initial:n = \Alph*,
674     label .value_required:n = true,
675     ref .code:n = \__enumext_keyans_ref:n {#1},
676     ref .value_required:n = true,
677 }

```

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

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

```

678 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
679 {
680     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
681     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
682     {
683         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
684     }
685     {
686         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
687         \__enumext_regex_counter_style:
688         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
689         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
690         {
691             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
692             { \exp_not:V \l__enumext_ref_key_arg_tl }
693         }
694     }
695 }

```

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

```

696 \cs_new_protected:Nn \__enumext_keyans_ref:
697 {
698     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
699     {
700         \tl_use:N \l__enumext_renew_the_count_v_tl
701     }
702 }

```

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

11.14 Setting start and widest keys

```
\__enumext_start_from:NNn
\__enumext_start_from:ccn
```

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

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

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

```
703 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
704 {
705   \__enumext_if_is_int:nTF { #3 }
706   {
707     \int_set:Nn #2 {#3}
708   }
709   {
710     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
711     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
712     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
713     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
714   }
715 }
716 \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.

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

(End of definition for `__enumext_widest_from:nNNn`.)

```
start
widest
\l__enumext_start_X_int
```

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

```
733 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
734 {
735   \keys_define:nn { enumext / #1 }
736   {
737     start .code:n = {
738       \__enumext_start_from:ccn
739       { \l__enumext_label_#2_tl }
740       { \l__enumext_start_#2_int } {##1}
741     },
742     start .initial:n = 1,
743     widest .code:n = {
744       \__enumext_widest_from:nccn {#2}
745       { \l__enumext_label_#2_tl }
746       { \l__enumext_labelwidth_#2_dim } {##1}
747     },

```

```

748         widest .value_required:n = true,
749         start .value_required:n = true,
750     }
751 }
752 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.15 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
753 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
754 {
755     \keys_define:nn { enumext / #1 }
756     {
757         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
758         topsep .initial:n = {#3},
759         topsep .value_required:n = true,
760         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
761         partopsep .initial:n = {#4},
762         partopsep .value_required:n = true,
763         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
764         parsep .initial:n = {#5},
765         parsep .value_required:n = true,
766         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
767         itemsep .initial:n = {#6},
768         itemsep .value_required:n = true,
769         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
770         noitemsep .value_forbidden:n = true,
771         nosepp .meta:n = {
772             itemsep = 0pt, parsep = 0pt,
773             topsep = 0pt, partopsep = 0pt,
774         },
775         nosepp .value_forbidden:n = true,
776     }
777 }

```

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

```

778 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
779 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
780 { 4.0pt plus 2.0pt minus 1.0pt }
781 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
782 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
783 { 2.0pt plus 1.0pt minus 1.0pt }
784 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
785 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
786 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
787 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
788 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
789 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
790 { 2.0pt plus 1.0pt minus 1.0pt }
791 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
792 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
793 { 4.0pt plus 2.0pt minus 1.0pt }
794 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
795 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
796 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

11.16 Setting base-fix key

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

```

base-fix
\__enumext_nested_base_line_fix:
797 \cs_set_protected:Npn \__enumext_tmp:n #1
798 {
799     \keys_define:nn { enumext / #1 }

```

```

800     {
801         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
802         base-fix .initial:n = false,
803         base-fix .value_forbidden:n = true,
804     }
805 }
806 \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 (§11.38) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.42)

```

807 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
808 {
809     \bool_lazy_and:nnT
810     { \bool_if_p:N \l__enumext_standar_first_bool }
811     { \bool_if_p:N \l__enumext_base_line_fix_bool }
812     {
813         \mode_leave_vertical:
814         \vspace { -\baselineskip }
815         \keys_set:nn { enumext / level-1 }
816         {
817             topsep = 0pt, above = 0pt, above* = 0pt,
818         }
819     }
820     \bool_lazy_and:nnT
821     { \bool_if_p:N \l__enumext_starred_first_bool }
822     { \bool_if_p:N \l__enumext_base_line_fix_bool }
823     {
824         \mode_leave_vertical:
825         \vspace { -\baselineskip }
826         \keys_set:nn { enumext / enumext* }
827         {
828             topsep = 0pt, above = 0pt, above* = 0pt,
829         }
830     }
831     \bool_set_false:N \l__enumext_base_line_fix_bool
832 }

```

• This key is enabled by default in the command `\printkeyans` (§11.45).

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

11.17 Setting keys for horizontal spaces

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

```

833 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
834 {
835     \keys_define:nn { enumext / #1 }
836     {
837         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
838         itemindent .value_required:n = true,
839         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
840         rightmargin .value_required:n = true,
841         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
842         listparindent .value_required:n = true,
843         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
844         list-offset .value_required:n = true,
845         list-indent .code:n =
846             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
847             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
848         list-indent .value_required:n = true,
849     }
850 }
851 \clist_map_inline:nn
852 {
853     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
854 }
855 { \__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.

```

856 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
857 {
858   \keys_define:nn { enumext / #1 }
859   {
860     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
861     itemindent .value_required:n = true,
862     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
863     rightmargin .value_required:n = true,
864     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
865     listparindent .value_required:n = true,
866     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
867     list-offset .value_required:n = true,
868     list-indent .meta:n = { list-offset = ##1 },
869     list-indent .value_required:n = true,
870   }
871 }
872 \clist_map_inline:nn
873 {
874   {enumext*}{vii}, {keyans*}{viii}
875 }
876 { \__enumext_tmp:nn #1 }

```

11.17.1 Functions for setting the fake itemindent

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

```

877 \cs_set_protected:Nn \__enumext_fake_item:
878 {
879   \dim_compare:nNnT
880   { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
881   >
882   { \c_zero_dim }
883   {
884     \tl_set:ce { \__enumext_fake_item_indent_ \__enumext_level: _tl }
885     {
886       \exp_not:N \mode_leave_vertical:
887       \exp_not:n { \skip_horizontal:n {
888         { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
889         \ignorespaces
890       }
891     }
892   }
893 \cs_set_protected:Nn \__enumext_keyans_fake_item:
894 {
895   \dim_compare:nNnT
896   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
897   {
898     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
899     {
900       \exp_not:N \mode_leave_vertical:
901       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
902     }
903   }
904 }
905 \cs_set_protected:Nn \__enumext_fake_item_vii:
906 {
907   \dim_compare:nNnT
908   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
909   {
910     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
911     {
912       \exp_not:N \mode_leave_vertical:
913       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
914     }
915   }
916 }
917 \cs_set_protected:Nn \__enumext_fake_item_viii:

```



```

918 {
919   \dim_compare:nNt
920   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
921   {
922     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
923     {
924       \exp_not:N \mode_leave_vertical:
925       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
926     }
927   }
928 }

```

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

11.18 Setting show-length key

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

```

929 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
930 {
931   \keys_define:nn { enumext / #1 }
932   {
933     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
934     show-length .initial:n = false,
935   }
936 }
937 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.19 Setting before, after and first keys

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

```

938 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
939 {
940   \keys_define:nn { enumext / #1 }
941   {
942     before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
943     before .value_required:n = true,
944     before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
945     before* .value_required:n = true,
946     after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
947     after .value_required:n = true,
948     first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
949     first .value_required:n = true,
950   }
951 }
952 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.19.1 Functions for before, after and first keys in enumext

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

```

953 \cs_new_protected:Nn \__enumext_before_args_exec:
954 {
955   \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
956 }

```

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

```

957 \cs_new_protected:Nn \__enumext_before_keys_exec:
958 {
959   \tl_use:c { \l__enumext_before_no_starred_key_ \__enumext_level: _tl }
960 }

```

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

```

961 \cs_new_protected:Nn \__enumext_after_stop_list:
962 {
963   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
964 }

```

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`.

```

965 \cs_new_protected:Nn \__enumext_after_args_exec:
966 {
967   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
968 }

```

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

11.19.2 Functions for before, after and first keys in keyans

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:

```

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.

```

969 \cs_new_protected:Nn \__enumext_before_args_exec_v:
970 {
971   \tl_use:N \l__enumext_before_starred_key_v_tl
972 }

```

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 $\langle keys \rangle$.

```

973 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
974 {
975   \tl_use:N \l__enumext_before_no_starred_key_v_tl
976 }

```

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

```

977 \cs_new_protected:Nn \__enumext_after_stop_list_v:
978 {
979   \tl_use:N \l__enumext_after_stop_list_v_tl
980 }

```

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

```

981 \cs_new_protected:Nn \__enumext_after_args_exec_v:
982 {
983   \tl_use:N \l__enumext_after_list_args_v_tl
984 }

```

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

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

```

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

```

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.

```

985 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
986 {
987   \tl_use:N \l__enumext_before_starred_key_vii_tl
988 }
989 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
990 {
991   \tl_use:N \l__enumext_before_starred_key_viii_tl
992 }

```

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

```

993 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
994 {
995   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
996 }
997 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
998 {

```

```

999     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1000 }

```

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

```

1001 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1002 {
1003     \tl_use:N \l__enumext_after_stop_list_vii_tl
1004 }
1005 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1006 {
1007     \tl_use:N \l__enumext_after_stop_list_viii_tl
1008 }

```

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`.

```

1009 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1010 {
1011     \tl_use:N \l__enumext_after_list_args_vii_tl
1012 }
1013 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1014 {
1015     \tl_use:N \l__enumext_after_list_args_viii_tl
1016 }

```

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

11.20 Setting keys for multicols and minipage

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

```

1017 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1018 {
1019     \keys_define:nn { enumext / #1 }
1020     {
1021         mini-env    .dim_set:c = { l__enumext_minipage_right_#2_dim },
1022         mini-env    .value_required:n = true,
1023         mini-sep    .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1024         mini-sep    .initial:n = 0.3333em,
1025         mini-sep    .value_required:n = true,
1026         columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1027         columns-sep .value_required:n = true,
1028         columns     .int_set:c = { l__enumext_columns_#2_int },
1029         columns     .initial:n = 1,
1030         columns     .value_required:n = true,
1031     }
1032 }
1033 \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.

```

1034 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1035 {
1036     \keys_define:nn { enumext / #1 }
1037     {
1038         mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1039         mini-right .value_required:n = true,
1040         mini-right* .code:n = {
1041             \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1042             \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1043         },
1044         mini-right* .value_required:n = true,
1045     }
1046 }
1047 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.21 Adjustment of vertical spaces for multicol

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

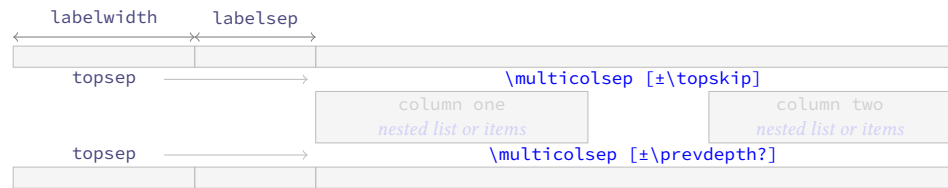


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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

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

11.21.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

1048 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1049 {
1050   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip {
1051     {
1052       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
1053       }
1054     }
1055   \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip {
1056     {
1057       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
1058     }
1059   }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

1060 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1061 {
1062   \int_case:nn { \__enumext_level_int }
1063   {
1064     { 2 } {
1065       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip } {
1066         {
1067           \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
1068         }
1069       }
1070     { 3 } {
1071       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip } {
1072         {
1073           \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
1074         }
1075       }
1076     { 4 } {
1077       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip } {
1078         {
1079           \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
1080         }
1081       }

```

```

1082     }
1083 }

```

(End of definition for `__enumext_add_pre_parse:`)

`__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)*.

```

1084 \cs_new_protected:Nn \__enumext_multi_addvspace:
1085 {
1086   \__enumext_multi_set_vskip:
1087   \mode_if_vertical:T
1088   {
1089     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1090     {
1091       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1092     }
1093     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1094     {
1095       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1096     }
1097   }
1098   \par\nopagebreak
1099   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1100 }

```

(End of definition for `__enumext_multi_addvspace:`)

11.21.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1101 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1102 {
1103   \skip_set:Nn \l__enumext_multicols_above_v_skip
1104   {
1105     \l__enumext_topsep_v_skip
1106   }
1107   \skip_set:Nn \l__enumext_multicols_below_v_skip
1108   {
1109     \l__enumext_topsep_v_skip
1110   }
1111 }
1112 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1113 {
1114   \__enumext_keyans_multi_set_vskip:
1115   \mode_if_vertical:T
1116   {
1117     \skip_add:Nn \l__enumext_multicols_above_v_skip
1118     {
1119       \skip_use:N \l__enumext_partopsep_v_skip
1120     }
1121     \skip_add:Nn \l__enumext_multicols_below_v_skip
1122     {
1123       \skip_use:N \l__enumext_partopsep_v_skip
1124     }
1125   }
1126   \par\nopagebreak
1127   \addvspace{ \l__enumext_multicols_above_v_skip }
1128 }

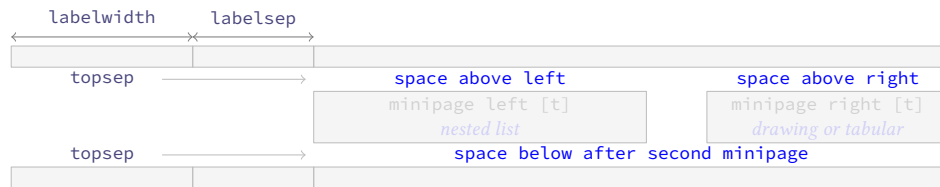
```

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

11.22 Adjustment of vertical spaces for minipage

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

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.

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

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

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

11.22.1 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 $\langle horizontal mode \rangle$, then we will make the settings for the $\langle vertical mode \rangle$ in which `\partopsep` comes into play.

First determine if the `multicols` 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`.

```
1129 \cs_new_protected:Nn __enumext_mini_set_vskip:
1130 {
1131   \int_compare:nNnTF
1132     { \int_use:c { l__enumext_columns_ __enumext_level: _int } } > { 1 }
1133     {
```

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.

```
1134   \skip_if_eq:nTF
1135     { \skip_use:c { l__enumext_topsep_ __enumext_level: _skip } } { \c_zero_skip }
1136   {
1137     \skip_set:Nn \l__enumext_minipage_left_skip
1138       {
1139         -0.150\box_dp:N \strutbox
1140       }
1141     \skip_set:Nn \l__enumext_minipage_right_skip
1142       {
1143         0.695\box_dp:N \strutbox
1144       }
1145     \skip_set:Nn \l__enumext_minipage_after_skip
1146       {
1147         \box_dp:N \strutbox
1148       }
1149     __enumext_zero_parsep:
1150   }
1151   {
1152     \skip_set:Nn \l__enumext_minipage_left_skip
1153       {
1154         \skip_use:c { l__enumext_topsep_ __enumext_level: _skip }
1155       }
1156     \skip_set:Nn \l__enumext_minipage_right_skip
1157       {
1158         0.695\box_dp:N \strutbox
1159       }
1160     \skip_set:Nn \l__enumext_minipage_after_skip
1161       {
1162         1.85\box_dp:N \strutbox
1163         + \skip_use:c { l__enumext_topsep_ __enumext_level: _skip }
1164       }
1165   }
```



```

1165     }
1166   }
1167   {

```

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.

```

1168     \skip_if_eq:nnTF
1169     { \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip } } { \c_zero_skip }
1170     {
1171       \skip_set:Nn \l__enumext_minipage_left_skip
1172       {
1173         0.5\box_dp:N \strutbox
1174         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1175       }
1176       \skip_set:Nn \l__enumext_minipage_right_skip
1177       {
1178         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1179       }
1180       \skip_set:Nn \l__enumext_minipage_after_skip
1181       {
1182         1.6\box_dp:N \strutbox
1183       }
1184     }
1185   {
1186     \skip_set:Nn \l__enumext_minipage_left_skip
1187     {
1188       0.5875\box_dp:N \strutbox
1189       - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1190     }
1191     \skip_set:Nn \l__enumext_minipage_right_skip
1192     {
1193       + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1194       + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1195     }
1196     \skip_set:Nn \l__enumext_minipage_after_skip
1197     {
1198       0.325\box_dp:N \strutbox
1199       + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1200     }
1201   }
1202 }
1203 }

```

(End of definition for `\l__enumext_mini_set_vskip:`)

`\l__enumext_zero_parsep:`

The function `\l__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.

```

1204 \cs_new_protected:Nn \l__enumext_zero_parsep:
1205 {
1206   \int_case:nn { \l__enumext_level_int }
1207   {
1208     { 2 } {
1209       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1210       {
1211         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1212       }
1213     }
1214     { 3 } {
1215       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1216       {
1217         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1218       }
1219     }
1220     { 4 } {
1221       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1222       {
1223         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1224       }
1225     }

```

```

1226     }
1227 }

```

(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*.

```

1228 \cs_new_protected:Nn \__enumext_mini_addvspace:
1229 {
1230   \__enumext_mini_set_vskip:
1231   \mode_if_vertical:T
1232   {
1233     \skip_add:Nn \l__enumext_minipage_left_skip
1234     {
1235       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1236     }
1237     \skip_add:Nn \l__enumext_minipage_after_skip
1238     {
1239       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1240     }
1241   }
1242   \par\nopagebreak
1243   \addvspace { \l__enumext_minipage_left_skip }
1244 }

```

(End of definition for `__enumext_mini_addvspace:`.)

11.22.2 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`.

```

1245 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1246 {
1247   \skip_zero_new:N \l__enumext_minipage_after_skip
1248   \skip_zero_new:N \l__enumext_minipage_left_skip
1249   \skip_zero_new:N \l__enumext_minipage_right_skip
1250   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1251   {
1252     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1253     {
1254       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1255       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1256       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1257       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1258       {
1259         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1260       }
1261     }
1262     {
1263       \skip_set:Nn \l__enumext_minipage_left_skip
1264       {
1265         \skip_use:N \l__enumext_topsep_v_skip
1266       }
1267       \skip_set:Nn \l__enumext_minipage_right_skip
1268       {
1269         0.705\box_dp:N \strutbox
1270       }
1271       \skip_set:Nn \l__enumext_minipage_after_skip
1272       {
1273         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1274       }
1275     }
1276   }
1277   {
1278     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1279     {
1280       \skip_set:Nn \l__enumext_minipage_left_skip
1281       {

```

```

1282         0.5\box_dp:N \strutbox
1283         + \l__enumext_partopsep_v_skip
1284     }
1285     \skip_set:Nn \l__enumext_minipage_right_skip
1286     {
1287         \l__enumext_partopsep_v_skip
1288     }
1289     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1290 }
1291 {
1292     \skip_set:Nn \l__enumext_minipage_left_skip
1293     {
1294         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1295     }
1296     \skip_set:Nn \l__enumext_minipage_right_skip
1297     {
1298         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1299     }
1300     \skip_set:Nn \l__enumext_minipage_after_skip
1301     {
1302         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1303     }
1304 }
1305 }
1306 }

```

(End of definition for `\l__enumext_keyans_mini_set_vskip:`)

`\l__enumext_keyans_mini_addvspace:`

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

```

1307 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1308 {
1309     \l__enumext_keyans_mini_set_vskip:
1310     \mode_if_vertical:T
1311     {
1312         \skip_add:Nn \l__enumext_minipage_left_skip
1313         {
1314             \l__enumext_partopsep_v_skip
1315         }
1316         \skip_add:Nn \l__enumext_minipage_after_skip
1317         {
1318             \l__enumext_partopsep_v_skip
1319         }
1320     }
1321     \par\nopagebreak
1322     \addvspace { \l__enumext_minipage_left_skip }
1323 }

```

(End of definition for `\l__enumext_keyans_mini_addvspace:`)

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

`\l__enumext_mini_set_vskip_vii:`

`\l__enumext_mini_set_vskip_viii:`

The functions `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `\l__enumext_mini_env*` environment in `enumext*` and `keyans*`.

```

1324 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1325 {
1326     \skip_zero_new:N \l__enumext_minipage_left_skip
1327     \skip_gzero_new:N \g__enumext_minipage_right_skip
1328     \skip_gzero_new:N \g__enumext_minipage_after_skip
1329     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1330     {
1331         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1332         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1333     }
1334     {
1335         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1336         \skip_gset:Nn \g__enumext_minipage_right_skip

```

```

1337         {
1338             \l__enumext_topsep_vii_skip
1339         }
1340         \skip_gset:Nn \g__enumext_minipage_after_skip
1341         {
1342             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1343         }
1344     }
1345 }
1346 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1347 {
1348     \skip_zero_new:N \l__enumext_minipage_after_skip
1349     \skip_zero_new:N \l__enumext_minipage_left_skip
1350     \skip_zero_new:N \l__enumext_minipage_right_skip
1351     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1352     {
1353         \skip_set:Nn \l__enumext_minipage_left_skip
1354         {
1355             0.5\box_dp:N \strutbox
1356         }
1357         \skip_set:Nn \l__enumext_minipage_right_skip
1358         {
1359             \l__enumext_partopsep_viii_skip
1360         }
1361         \skip_set:Nn \l__enumext_minipage_after_skip
1362         {
1363             1.6\box_dp:N \strutbox
1364         }
1365     }
1366     {
1367         \skip_set:Nn \l__enumext_minipage_left_skip
1368         {
1369             0.5875\box_dp:N \strutbox
1370         }
1371         \skip_set:Nn \l__enumext_minipage_right_skip
1372         {
1373             \l__enumext_topsep_viii_skip
1374         }
1375         \skip_set:Nn \l__enumext_minipage_after_skip
1376         {
1377             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1378         }
1379     }
1380 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

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

```

1381 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1382 {
1383     \__enumext_mini_set_vskip_vii:
1384     \par\nopagebreak
1385     \addvspace { \l__enumext_minipage_left_skip }
1386 }
1387 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1388 {
1389     \__enumext_mini_set_vskip_viii:
1390     \par\nopagebreak
1391     \addvspace { \l__enumext_minipage_left_skip }
1392 }

```

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

11.22.4 The command `\miniright`

The command `\miniright` will close the `__enumext_mini_env*` environment on the “left side”, open the `__enumext_mini_env*` environment on the “right side” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “right side” environment. The *starred argument* ‘`*`’ inhibits the use

of `\centering` command i.e. the usual \LaTeX justification is maintained in the `__enumext_mini_env*` on the “right side”.

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

```

1393 \NewDocumentCommand \miniright { s }
1394 {
1395   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1396   {
1397     \msg_error:nnn { enumext } { wrong-miniright-place }
1398   }
1399   \int_compare:nNt { \l__enumext_level_int } = { 0 }
1400   {
1401     \msg_error:nnn { enumext } { wrong-miniright-place }
1402   }
1403   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1404   {
1405     \__enumext_keyans_mini_right_cmd:n {#1}
1406   }
1407   { \__enumext_mini_right_cmd:n {#1} }
1408 }

```

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

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

```

1409 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1410 {
1411   \dim_compare:nNtF
1412   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1413   {
1414     \__enumext_multicols_stop:
1415     \end{__enumext_mini_env*}
1416     \hfill
1417     \begin{__enumext_mini_env*}
1418     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1419     \par\addvspace { \l__enumext_minipage_right_skip }
1420     \bool_if:nF {#1}
1421     {
1422       \centering
1423     }
1424     \int_gzero:N \g__enumext_minipage_stat_int
1425   }
1426   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1427 }

```

(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.

```

1428 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1429 {
1430   \dim_compare:nNtF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1431   {
1432     \__enumext_keyans_multicols_stop:
1433     \end{__enumext_mini_env*}
1434     \hfill
1435     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1436     \par\addvspace { \l__enumext_minipage_right_skip }
1437     \bool_if:nF {#1}
1438     {
1439       \centering
1440     }

```

```

1441         \int_gzero:N \g__enumext_minipage_stat_int
1442     }
1443     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1444 }

```

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

11.23 Setting above and below keys

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

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

```

above* \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below  {
below* \keys_define:nn { enumext / #1 }
{
    above .skip_set:c = { \__enumext_vspace_above_#2_skip },
    above .value_required:n = true,
    above* .code:n = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
    \keys_set:nn { enumext / #1 } { above = {##1} },
    above* .value_required:n = true,
    below .skip_set:c = { \__enumext_vspace_below_#2_skip },
    below .value_required:n = true,
    below* .code:n = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
    \keys_set:nn { enumext / #1 } { below = {##1} },
    below* .value_required:n = true,
}
}
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.23.1 Functions for above and below keys in enumext

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

```

1462 \cs_new_protected:Nn \__enumext_vspace_above:
1463 {
1464     \skip_if_eq:nnF
1465     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1466     {
1467         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1468         {
1469             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1470         }
1471         {
1472             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1473         }
1474     }
1475 }

```

(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.

```

1476 \cs_new_protected:Nn \__enumext_vspace_below:
1477 {
1478     \skip_if_eq:nnF
1479     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1480     {
1481         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1482         {
1483             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1484         }
1485         {
1486             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1487         }
1488     }
1489 }

```


(End of definition for `__enumext_vspace_below:`.)

11.23.2 Functions for above and below keys in keyans

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

```

1490 \cs_new_protected:Nn \__enumext_vspace_above_v:
1491 {
1492   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1493   {
1494     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1495     {
1496       \vspace*{ \l__enumext_vspace_above_v_skip }
1497     }
1498     { \vspace { \l__enumext_vspace_above_v_skip } }
1499   }
1500 }
```

(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.

```

1501 \cs_new_protected:Nn \__enumext_vspace_below_v:
1502 {
1503   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1504   {
1505     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1506     {
1507       \vspace*{ \l__enumext_vspace_below_v_skip }
1508     }
1509     { \vspace { \l__enumext_vspace_below_v_skip } }
1510   }
1511 }
```

(End of definition for `__enumext_vspace_below_v:`.)

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

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

`__enumext_vspace_above_viii:`

```

1512 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1513 {
1514   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1515   {
1516     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1517     {
1518       \vspace*{ \l__enumext_vspace_above_vii_skip }
1519     }
1520     { \vspace { \l__enumext_vspace_above_vii_skip } }
1521   }
1522 }
1523 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1524 {
1525   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1526   {
1527     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1528     {
1529       \vspace*{ \l__enumext_vspace_above_viii_skip }
1530     }
1531     { \vspace { \l__enumext_vspace_above_viii_skip } }
1532   }
1533 }
```

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

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

`__enumext_vspace_below_viii:`

```

1534 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1535 {
1536   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1537   {
1538     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1539     {
```

```

1540         \vspace*{ \l__enumext_vspace_below_vii_skip }
1541     }
1542     { \vspace { \l__enumext_vspace_below_vii_skip } }
1543 }
1544 }
1545 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1546 {
1547     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1548     {
1549         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1550         {
1551             \vspace*{ \l__enumext_vspace_below_viii_skip }
1552         }
1553         { \vspace { \l__enumext_vspace_below_viii_skip } }
1554     }
1555 }

```

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

11.24 Setting series, resume and resume* keys

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

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1556 \cs_set_protected:Npn \__enumext_tmp:n #1
1557 {
1558     \keys_define:nn { enumext / #1 }
1559     {
1560         series .str_set:N = \l__enumext_series_str,
1561         series .value_required:n = true,
1562         resume .code:n = \__enumext_resume_series:n {##1},
1563         resume* .code:n = \__enumext_resume_starred:,
1564         resume* .value_forbidden:n = true,
1565     }
1566 }
1567 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {##1} }

```

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

11.24.1 Internal functions for series key

`__enumext_filter_series:n` 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.

```

1568 \cs_new:Npn \__enumext_filter_series:n #1
1569 {
1570     \use:e
1571     {
1572         \keyval_parse:NNn
1573         \__enumext_filter_series_key:n
1574         \__enumext_filter_series_pair:nn {##1}
1575     }
1576 }

```

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.

```

1577 \cs_new:Npn \__enumext_filter_series_key:n #1
1578 {
1579     \str_case:nnF {##1}
1580     {
1581         { resume } {}
1582         { resume* } {}
1583         { base-fix } {}
1584     }
1585     { , { \exp_not:n {##1} } }
1586 }

```

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.

```

1587 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1588 {

```

```

1589     \str_case:nnF {#1}
1590     {
1591         { series } {}
1592         { resume } {}
1593         { start } {}
1594         { save-ans } {}
1595         { save-key } {}
1596     }
1597     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1598 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`; in this case we must make sure it is set to false so that it does not overwrite the default filtered *keys*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§11.38) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.42).

```

1599 \cs_new_protected:Npn \__enumext_parse_series:n #1
1600 {
1601     \str_if_empty:NTF \l__enumext_series_str
1602     {
1603         \bool_if:NF \l__enumext_resume_active_bool
1604         {
1605             \__enumext_resume_last:n {#1}
1606         }
1607     }
1608     {
1609         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1610         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1611             { \__enumext_filter_series:n {#1} }
1612         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1613         {
1614             \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1615         }
1616     }
1617 }

```

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.

```

1618 \cs_new_protected:Npn \__enumext_resume_last:n #1
1619 {
1620     \bool_if:NT \l__enumext_standar_first_bool
1621     {
1622         \tl_gclear:N \g__enumext_standar_series_tl
1623         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1624     }
1625     \bool_if:NT \l__enumext_starred_first_bool
1626     {
1627         \tl_gclear:N \g__enumext_starred_series_tl
1628         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1629     }
1630 }

```

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

11.24.2 Internal function to save counter value

```
\__enumext_resume_save_counter:
```

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

- The variables `__enumext_series_str` and `__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments, the integer variable with `__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§11.38) and the `enumext*` environment definition (§11.42).

```

1631 \cs_new_protected:Nn \__enumext_resume_save_counter:
1632 {
1633   \bool_if:NT \g__enumext_standar_bool
1634   {
1635     \tl_if_empty:NF \__enumext_series_str
1636     {
1637       \int_gset_eq:cN
1638       { g__enumext_series_ \__enumext_series_str_int } \value{enumXi}
1639     }
1640     \tl_if_empty:NTF \__enumext_resume_name_tl
1641     {
1642       \str_if_empty:NT \__enumext_series_str
1643       {
1644         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1645       }
1646     }
1647     {
1648       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1649       {
1650         \int_gset_eq:cN
1651         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXi}
1652       }
1653     }
1654     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1655     {
1656       \int_gset_eq:cN
1657       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXi}
1658     }
1659   }
1660   \bool_if:NT \g__enumext_starred_bool
1661   {
1662     \tl_if_empty:NF \__enumext_series_str
1663     {
1664       \int_gset_eq:cN
1665       { g__enumext_series_ \__enumext_series_str_int } \value{enumXvii}
1666     }
1667     \tl_if_empty:NTF \__enumext_resume_name_tl
1668     {
1669       \str_if_empty:NT \__enumext_series_str
1670       {
1671         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1672       }
1673     }
1674     {
1675       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1676       {
1677         \int_gset_eq:cN
1678         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXvii}
1679       }
1680     }
1681     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1682     {
1683       \int_gset_eq:cN
1684       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXvii}
1685     }
1686   }
1687 }

```

(End of definition for `__enumext_resume_save_counter:`)

11.24.3 Internal functions for resume key

`__enumext_resume_series:n`

The function `__enumext_resume_series:n` will handle the argument passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function `__enumext__resume_counter:` is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is not present, if the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key, otherwise it

will verify that the `\g__enumext_series_⟨series name⟩_tl` variable set by the `series` key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```

1688 \cs_new_protected:Npn \__enumext_resume_series:n #1
1689 {
1690   \tl_if_empty:NTF {#1}
1691   {
1692     \__enumext_resume_counter:n { }
1693   }
1694   {
1695     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1696     {
1697       \__enumext_resume_counter:n {#1}
1698       \bool_if:NT \g__enumext_standar_bool
1699       {
1700         \keys_set:nv { enumext / level-1 }
1701         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1702       }
1703       \bool_if:NT \g__enumext_starred_bool
1704       {
1705         \keys_set:nv { enumext / enumext* }
1706         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1707       }
1708     }
1709     {
1710       \bool_if:NT \g__enumext_standar_bool
1711       {
1712         \msg_error:nnn { enumext } { unknown-series } {#1}
1713       }
1714       \bool_if:NT \g__enumext_starred_bool
1715       {
1716         \msg_error:nnn { enumext } { unknown-series } {#1}
1717       }
1718     }
1719   }
1720 }

```

(End of definition for `__enumext_resume_series:n`)

```

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

```

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

```

1721 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1722 {
1723   \bool_set_true:N \l__enumext_resume_active_bool
1724   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1725   \tl_if_empty:NTF \l__enumext_resume_name_tl
1726   {
1727     \__enumext_resume_counter:
1728   }
1729   {
1730     \__enumext_resume_counter_series:
1731   }
1732   \__enumext_resume_counter_save_ans:
1733 }

```

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.

```

1734 \cs_new_protected:Nn \__enumext_resume_counter:
1735 {
1736   \bool_if:NT \g__enumext_standar_bool
1737   {
1738     \int_gincr:N \g__enumext_resume_int
1739     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1740   }
1741   \bool_if:NT \g__enumext_starred_bool
1742   {
1743     \int_gincr:N \g__enumext_resume_vii_int

```

```

1744         \int_set_eq:Nn \l__enumext_start_vii_int \g__enumext_resume_vii_int
1745     }
1746 }

```

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

```

1747 \cs_new_protected:Nn \__enumext_resume_counter_series:
1748 {
1749     \bool_if:NT \g__enumext_standar_bool
1750     {
1751         \int_set:Nn \l__enumext_start_i_int
1752         {
1753             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1754         }
1755     }
1756     \bool_if:NT \g__enumext_starred_bool
1757     {
1758         \int_set:Nn \l__enumext_start_vii_int
1759         {
1760             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1761         }
1762     }
1763 }

```

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.

```

1764 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1765 {
1766     \bool_lazy_and:nnT
1767     { \bool_if_p:N \l__enumext_standar_first_bool }
1768     { \bool_if_p:N \l__enumext_store_active_bool }
1769     {
1770         \int_set:Nn \l__enumext_start_i_int
1771         {
1772             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1773         }
1774     }
1775     \bool_lazy_and:nnT
1776     { \bool_if_p:N \l__enumext_starred_first_bool }
1777     { \bool_if_p:N \l__enumext_store_active_bool }
1778     {
1779         \int_set:Nn \l__enumext_start_vii_int
1780         {
1781             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1782         }
1783     }
1784 }

```

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

11.24.4 Internal function for `resume*` key

`__enumext_resume_starred:`

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

```

1785 \cs_new_protected:Nn \__enumext_resume_starred:
1786 {
1787     \bool_if:NT \g__enumext_standar_bool
1788     {
1789         \tl_if_empty:NF \g__enumext_standar_series_tl
1790         {
1791             \__enumext_resume_counter:n { }
1792             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1793         }
1794     }
1795     \bool_if:NT \g__enumext_starred_bool
1796     {
1797         \tl_if_empty:NF \g__enumext_starred_series_tl
1798         {

```

```

1799         \__enumext_resume_counter:n { }
1800         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1801     }
1802 }
1803 }

```

(End of definition for __enumext_resume_starred:.)

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

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

11.25.1 Setting save-ans key

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

```

1804 \cs_set_protected:Npn \__enumext_tmp:n #1
1805 {
1806     \keys_define:nn { enumext / #1 }
1807     {
1808         save-ans .code:n = \__enumext_storing_set:n {##1},
1809         save-ans .value_required:n = true,
1810     }
1811 }
1812 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.25.2 Internal functions for save-ans key

`__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:`.

```

1813 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1814 {
1815     \msg_term:nnVV { enumext } { save-ans-log }
1816     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1817 }
1818 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1819 {
1820     \msg_term:nnVV { enumext } { save-ans-log-hook }
1821     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1822 }

```

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

`__enumext_storing_set:n` and `__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.

```

1823 \cs_new_protected:Npn \__enumext_storing_set:n #1
1824 {
1825     \tl_set:Nx \l__enumext_store_name_tl {#1}
1826     \tl_if_empty:NTF \l__enumext_store_name_tl
1827     {
1828         \bool_lazy_or:nnT
1829         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1830         {
1831             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1832         }
1833     }
1834     {
1835         \bool_lazy_or:nnT
1836         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1837         {
1838             \__enumext_start_save_ans_msg:
1839             \__enumext_storing_exec:
1840         }
1841     }
1842 }

```


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

```

1843 \cs_new_protected:Nn \__enumext_storing_exec:
1844 {
1845   \bool_set_true:N \l__enumext_store_active_bool
1846   \bool_set_true:N \l__enumext_check_answers_bool
1847   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1848   \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1849   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1850   {
1851     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1852     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1853   }
1854   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1855   {
1856     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1857     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1858   }
1859   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1860   {
1861     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1862     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1863   }
1864 }

```

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

11.25.3 The check answer mechanism

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

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

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

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

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

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

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

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

11.25.4 Setting check-ans and no-store keys

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

```

check-ans  \cs_set_protected:Npn \__enumext_tmp:n #1
no-store   {
1865       \keys_define:nn { enumext / #1 }
1866       {
1867         {
1868           check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1869           check-ans .initial:n = false,
1870           check-ans .value_required:n = true,
1871           no-store .code:n = {
1872             \bool_set_false:N \__enumext_check_answers_bool
1873             \bool_set_false:N \__enumext_check_ans_key_bool
1874           },
1875           no-store .value_forbidden:n = true,
1876         }
1877       }
1878     }
1879   \clist_map_inline:nn
1880   {
1881     level-1, level-2, level-3, level-4, enumext*
1882   }
1883   { \__enumext_tmp:n {#1} }

```

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

11.25.5 Set-up check answer mechanism

The function `__enumext_check_ans_active:` will first check the state of the variable `__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `__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.

```

1884 \cs_new_protected:Nn \__enumext_check_ans_active:
1885 {
1886   \tl_if_empty:NF \__enumext_store_name_tl
1887   {
1888     \bool_if:NT \__enumext_check_answers_bool
1889     {
1890       \__enumext_check_ans_level:
1891     }
1892   }
1893 }

```

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 `__enumext_item_number_bool` to “false”.

```

1894 \cs_new_protected:Nn \__enumext_check_ans_level:
1895 {
1896   \int_case:nn { \__enumext_level_int }
1897   {
1898     { 1 }{
1899       \bool_lazy_all:NT
1900       {
1901         { \bool_if_p:N \g__enumext_starred_bool }
1902         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1903       }
1904       {
1905         \int_gdecr:N \g__enumext_item_number_int
1906         \bool_set_false:N \__enumext_item_number_bool
1907       }
1908     }
1909     { 2 }{
1910       \int_gdecr:N \g__enumext_item_number_int
1911       \bool_set_false:N \__enumext_item_number_bool
1912     }
1913     { 3 }{
1914       \int_gdecr:N \g__enumext_item_number_int
1915       \bool_set_false:N \__enumext_item_number_bool
1916     }
1917     { 4 }{
1918       \int_gdecr:N \g__enumext_item_number_int
1919       \bool_set_false:N \__enumext_item_number_bool

```

```

1920         }
1921     }

```

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.

```

1922     \int_case:nn { \l__enumext_level_h_int }
1923     {
1924         { 1 }{
1925             \bool_lazy_all:nT
1926             {
1927                 { \bool_if_p:N \g__enumext_standar_bool }
1928                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1929             }
1930             {
1931                 \int_gdecr:N \g__enumext_item_number_int
1932                 \bool_set_false:N \l__enumext_item_number_bool
1933             }
1934         }
1935     }
1936 }

```

(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.

```

1937 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1938 {
1939     \bool_lazy_and:nnT
1940     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1941     { \bool_if_p:N \g__enumext_standar_bool }
1942     {
1943         \bool_gset_true:N \g__enumext_check_ans_key_bool
1944     }
1945     \bool_lazy_and:nnT
1946     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1947     { \bool_if_p:N \g__enumext_starred_bool }
1948     {
1949         \bool_gset_true:N \g__enumext_check_ans_key_bool
1950     }
1951 }

```

(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:`.

```

1952 \cs_new_protected:Nn \__enumext_item_answer_diff:
1953 {
1954     \int_gset:Nn \g__enumext_item_answer_diff_int
1955     {
1956         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1957     }
1958 }

```

(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:`.

`__enumext_check_ans_msg_less:`
`__enumext_check_ans_msg_same_ok:`
`__enumext_check_ans_msg_greater:`

```

1959 \cs_new_protected:Nn \__enumext_check_ans_show:
1960 {
1961     \int_case:nn { \g__enumext_item_answer_diff_int }
1962     {
1963         { -1 }{ \__enumext_check_ans_msg_less: }
1964         { 0 }{ \__enumext_check_ans_msg_same_ok: }
1965         { 1 }{ \__enumext_check_ans_msg_greater: }
1966     }
1967 }

```

```

1968 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1969 {
1970   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1971   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1972 }
1973 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1974 {
1975   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1976   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1977 }
1978 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1979 {
1980   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1981   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1982 }

```

(End of definition for __enumext_check_ans_show: and others.)

```

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

```

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

```

1983 \cs_new_protected:Nn \__enumext_check_ans_log:
1984 {
1985   \int_case:nn { \g__enumext_item_answer_diff_int }
1986   {
1987     { -1 } { \__enumext_check_ans_log_msg_less: }
1988     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1989     { 1 } { \__enumext_check_ans_log_msg_greater: }
1990   }
1991 }
1992 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1993 {
1994   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1995   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1996 }
1997 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1998 {
1999   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2000   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2001 }
2002 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2003 {
2004   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2005   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2006 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.25.6 Check for \item* and \anspic* commands

```
\__enumext_check_starred_cmd:n
```

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

```

2007 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2008 {
2009   \int_compare:nNt
2010   { \g__enumext_check_starred_cmd_int } = { 0 }
2011   {
2012     \msg_warning:nnnV
2013     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2014   }
2015   \int_compare:nNt
2016   { \g__enumext_check_starred_cmd_int } > { 1 }
2017   {
2018     \msg_warning:nnnV
2019     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2020   }
2021   \int_gzero:N \g__enumext_check_starred_cmd_int
2022   \tl_clear:N \l__enumext_check_start_line_env_tl
2023 }

```

(End of definition for __enumext_check_starred_cmd:n.)

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*`.

```

2024 \cs_set_protected:Npn \__enumext_tmp:n #1
2025 {
2026   \keys_define:nn { enumext / #1 }
2027   {
2028     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2029     wrap-ans .initial:n = \fbox{##1},
2030     wrap-ans .value_required:n = true,
2031     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2032     wrap-opt .initial:n = [{##1}],
2033     wrap-opt .value_required:n = true,
2034     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2035     save-sep .initial:n = {, ~ },
2036     save-sep .value_required:n = true,
2037     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2038     mark-ans .initial:n = \textasteriskcentered,
2039     mark-ans .value_required:n = true,
2040     mark-pos .choice:,
2041     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2042     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2043     mark-pos / unknown .code:n =
2044       \msg_error:nneee { enumext } { unknown-choice }
2045       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2046     mark-pos .initial:n = right,
2047     mark-pos .value_required:n = true,
2048     show-ans .bool_set:N = \__enumext_show_answer_bool,
2049     show-ans .initial:n = false,
2050     show-ans .value_required:n = true,
2051     show-pos .bool_set:N = \__enumext_show_position_bool,
2052     show-pos .initial:n = false,
2053     show-pos .value_required:n = true,
2054     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2055     mark-ref .initial:n = \textasteriskcentered,
2056     mark-ref .value_required:n = true,
2057     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2058     save-ref .initial:n = false,
2059     save-ref .value_required:n = true,
2060   }
2061 }
2062 \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`.

```

2063 \cs_set_protected:Npn \__enumext_tmp:n #1
2064 {
2065   \keys_define:nn { enumext / #1 }
2066   {
2067     mark-pos .choice:,
2068     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2069     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2070     mark-pos .initial:n = right,
2071     mark-pos .value_required:n = true,
2072     show-ans .bool_set:N = \__enumext_show_answer_bool,
2073     show-ans .initial:n = false,
2074     show-ans .value_required:n = true,
2075     show-pos .bool_set:N = \__enumext_show_position_bool,
2076     show-pos .initial:n = false,
2077     show-pos .value_required:n = true,
2078   }
2079 }
2080 \clist_map_inline:nn { keyans, keyans* } { \__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 `__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `__enumext_store_save_key_X_tl` with the filtered *⟨keys⟩*.

```

2081 \cs_new_protected:Npn __enumext_store_active_keys:n #1
2082 {
2083   \bool_if:cF { __enumext_store_save_key_ __enumext_level: _bool }
2084   {
2085     \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2086     \tl_set:ce
2087       { __enumext_store_save_key_ __enumext_level: _tl }
2088       { __enumext_filter_save_key:n {#1} }
2089   }
2090 }
2091 \cs_new_protected:Npn __enumext_store_active_keys_vii:n #1
2092 {
2093   \bool_if:NF __enumext_store_save_key_vii_bool
2094   {
2095     \tl_clear:N __enumext_store_save_key_vii_tl
2096     \tl_set:Nc __enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2097   }
2098 }

```

(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 `__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.

```

2099 \cs_set_protected:Npn __enumext_tmp:n #1
2100 {
2101   \keys_define:nn { enumext / enumext* }
2102   {
2103     save-key .code:n = __enumext_parse_save_key_vii:n {##1},
2104     save-key .value_required:n = true,
2105   }
2106   \keys_define:nn { enumext / #1 }
2107   {
2108     save-key .code:n = __enumext_parse_save_key:n {##1},
2109     save-key .value_required:n = true,
2110   }
2111 }
2112 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { __enumext_tmp:n {#1} }

```

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

```

__enumext_parse_save_key:n
__enumext_parse_save_key_vii:n

```

The functions `__enumext_parse_save_key:n` and `__enumext_parse_save_key_vii:n` will be responsible for storing the filtered *⟨keys⟩* in the variable `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2113 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2114 {
2115   \bool_set_true:c { __enumext_store_save_key_ __enumext_level: _bool }
2116   \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2117   \tl_set:ce
2118     { __enumext_store_save_key_ __enumext_level: _tl }
2119     { __enumext_filter_save_key:n {#1} }
2120 }

```

```

2121 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2122 {
2123   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2124   \tl_clear:N \l__enumext_store_save_key_vii_tl
2125   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2126 }

```

(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

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

```

2127 \cs_new:Npn \__enumext_filter_save_key:n #1
2128 {
2129   \use:e
2130   {
2131     \keyval_parse:NNn
2132     \__enumext_filter_save_key_key:n
2133     \__enumext_filter_save_key_pair:nn {#1}
2134   }
2135 }

```

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

```

2136 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2137 {
2138   \str_case:nnF {#1}
2139   {
2140     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2141   }
2142   { , { \exp_not:n {#1} } }
2143 }

```

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

```

2144 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2145 {
2146   \str_case:nnF {#1}
2147   {
2148     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2149     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2150     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2151     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2152     { mini-right* } {}
2153   }
2154   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2155 }

```

(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

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

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

```

2156 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2157 {
2158   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2159   {
2160     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2161   }
2162   { #1 }
2163 }
2164 \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

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.

```
2165 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2166 {
2167   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2168 }
2169 \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

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.

```
2170 \cs_new_protected:Nn \__enumext_store_level_open:
2171 {
2172   \bool_if:NT \__enumext_check_answers_bool
2173   {
2174     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2175     {
2176       \__enumext_store_addto_seq:n
2177       {
2178         \item \begin{enumext}
2179       }
2180     }
2181     {
2182       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2183       {
2184         \item \begin{enumext} [
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 }
2194 \cs_new_protected:Nn \__enumext_store_level_close:
2195 {
2196   \bool_if:NT \__enumext_check_answers_bool
2197   {
2198     \__enumext_store_addto_seq:n { \end{enumext} }
2199   }
2200 }
```

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

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

```
2201 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2202 {
2203   \bool_if:NT \__enumext_check_answers_bool
2204   {
2205     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2206     {
2207       \__enumext_store_addto_seq:n
2208       {
2209         \item \begin{enumext*}
2210       }
2211     }
2212     {
2213       \tl_put_left:Nn l__enumext_store_save_key_vii_tl
2214       {
2215         \item \begin{enumext*}[
2216       }
2217       \tl_put_right:Nn l__enumext_store_save_key_vii_tl
```

```

2218         {
2219         }
2220     }
2221     \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2222 }
2223 }
2224 }
2225 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2226 {
2227     \bool_if:NT \l__enumext_check_answers_bool
2228     {
2229         \__enumext_store_addto_seq:n { \end{enumext*} }
2230     }
2231 }

```

(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

```

2232 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2233 {
2234     \mode_leave_vertical:
2235     \skip_horizontal:n { -\dim_use:N #2 }
2236     \makebox[0pt][ r ]
2237     {
2238         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2239         {
2240             \tl_use:N \l__enumext_mark_answer_sym_tl
2241         }
2242     }
2243     \skip_horizontal:n { \dim_use:N #2 }
2244 }
2245 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

11.27 The internal label and ref

The function __enumext_store_internal_ref: handles the internal “*label and ref*” system used by the `save-ref` and `mark-ref` keys for \anskey will allow to execute \ref{<store name : position>} and will return 1. (a) . i . A.

__enumext_store_internal_ref:

First we will remove the dots “.” from the current *(labels)*, we do not want to get double dots in our references, then we will place this in the variable \l__enumext_newlabel_arg_two_tl.

```

2246 \cs_new_protected:Nn \__enumext_store_internal_ref:
2247 {
2248     \cs_set_protected:Npn \__enumext_tmp:n ##1
2249     {
2250         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2251         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2252         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2253         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2254     }
2255     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2256     \cs_set:Npn \__enumext_tmp:n ##1
2257     { . \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.

```

2258     \bool_lazy_all:nT
2259     {
2260         { \bool_if_p:N \g__enumext_starred_bool }
2261         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2262     }
2263     {
2264         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

2265         { \tl_use:N \l__enumext_label_copy_vii_tl }
2266     }
2267 \bool_lazy_all:nT
2268 {
2269     { \bool_if_p:N \l__enumext_standar_bool }
2270     { \bool_if_p:N \g__enumext_starred_bool }
2271     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2272 }
2273 {
2274     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2275     {
2276         \tl_use:N \l__enumext_label_copy_vii_tl
2277         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2278     }
2279 }

```

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.

```

2280 \bool_lazy_all:nT
2281 {
2282     { \bool_if_p:N \l__enumext_standar_bool }
2283     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2284     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2285     { \bool_not_p:n { \l__enumext_starred_bool } }
2286 }
2287 {
2288     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2289     {
2290         \tl_use:N \l__enumext_label_copy_i_tl
2291         \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2292     }
2293 }
2294 \cs_set:Npn \__enumext_tmp:n ##1
2295 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2296 \bool_lazy_all:nT
2297 {
2298     { \bool_if_p:N \l__enumext_standar_bool }
2299     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2300     { \bool_not_p:n { \g__enumext_starred_bool } }
2301     { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
2302 }
2303 {
2304     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2305     {
2306         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2307         . \tl_use:N \l__enumext_label_copy_vii_tl
2308     }
2309 }

```

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

```

2310 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2311 {
2312     \l__enumext_store_name_tl \c_colon_str
2313     \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2314 }

```

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

```

2315 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2316 {
2317     \__enumext_newlabel:nn
2318     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2319     { \l__enumext_newlabel_arg_two_tl }
2320 }
2321 \l__enumext_write_aux_file_tl
2322 }

```

(End of definition for `__enumext_store_internal_ref:.`)

11.28 Common functions for \anskey and anskey* environment

_enumext_store_anskey_code:n

The internal function _enumext_store_anskey_code:n first we pass the *argument* to the *prop list*, then checks the state of the variable _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*.

```

2323 \cs_new_protected:Npn \_enumext_store_anskey_code:n #1
2324 {
2325   \int_gincr:N \g__enumext_item_anskey_int
2326   \_enumext_store_addto_prop:n {#1}
2327   \bool_if:NT \_enumext_store_ref_key_bool
2328   {
2329     \_enumext_store_internal_ref:
2330   }
2331   \_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 _enumext_store_anskey_arg_tl which we will “*store*” in the *sequence*. First we clear the variable _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*.

```

2332 \tl_clear:N \_enumext_store_anskey_arg_tl
2333 \bool_lazy_and:nnT
2334 { \bool_if_p:N \_enumext_store_columns_break_bool }
2335 { \bool_not_p:n { \_enumext_starred_bool } }
2336 {
2337   \tl_put_left:Nn \_enumext_store_anskey_arg_tl { \columnbreak }
2338 }
2339 \tl_put_right:Nn \_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 _enumext_store_anskey_arg_tl.

```

2340 \bool_lazy_and:nnT
2341 { \bool_not_p:n { \_enumext_starred_bool } }
2342 { \int_compare_p:nNn { \_enumext_store_item_join_int } > { 1 } }
2343 {
2344   \tl_put_right:Ne \_enumext_store_anskey_arg_tl
2345   {
2346     ( \exp_not:V \_enumext_store_item_join_int )
2347   }
2348 }

```

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

```

2349 \bool_if:NTF \_enumext_store_item_star_bool
2350 {
2351   \tl_put_right:Nn \_enumext_store_anskey_arg_tl { * }
2352   \tl_if_empty:NF \_enumext_store_item_symbol_tl
2353   {
2354     \tl_put_right:Ne \_enumext_store_anskey_arg_tl
2355     {
2356       [ \exp_not:V \_enumext_store_item_symbol_tl ]
2357     }
2358   }
2359   \dim_compare:nT
2360   {
2361     \_enumext_store_item_symbol_sep_dim != \c_zero_dim
2362   }
2363   {
2364     \tl_put_right:Ne \_enumext_store_anskey_arg_tl
2365     {
2366       [ \exp_not:V \_enumext_store_item_symbol_sep_dim ]
2367     }
2368   }
2369   \tl_put_right:Nn \_enumext_store_anskey_arg_tl {#1}
2370 }
2371 {
2372   \tl_put_right:Nn \_enumext_store_anskey_arg_tl {#1}
2373 }

```

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*.

```

2374 \bool_lazy_and:nnT

```

```

2375     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2376     { \bool_if_p:N \l__enumext_hyperref_bool }
2377     {
2378         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2379         {
2380             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2381             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2382         }
2383     }
2384     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2385 }

```

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

`__enumext_anskey_show_wrap_arg:n`

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

```

2386 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2387 {
2388     \par
2389     \bool_if:NT \l__enumext_starred_bool
2390     {
2391         \cs_set:Nn \__enumext_level: { vii }
2392     }
2393     \__enumext_print_keyans_box:cc
2394     { \l__enumext_labelwidth_ \__enumext_level: _dim }
2395     { \l__enumext_labelsep_ \__enumext_level: _dim }
2396     \__enumext_anskey_wrapper:n { #1 }
2397 }

```

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

`__enumext_anskey_show_wrap_left:n`

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

```

2398 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2399 {
2400     \bool_if:NT \l__enumext_show_answer_bool
2401     {
2402         \__enumext_anskey_show_wrap_arg:n { #1 }
2403     }
2404     \bool_if:NT \l__enumext_show_position_bool
2405     {
2406         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2407         {
2408             \group_begin:
2409             \exp_not:N \normalfont
2410             \exp_not:N \footnotesize [ \int_eval:n
2411             {
2412                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2413             }
2414             ]
2415             \group_end:
2416         }
2417         \__enumext_anskey_show_wrap_arg:n { #1 }
2418     }
2419 }

```

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

11.29 The command `\anskey`

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

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

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

```

2420 \keys_define:nn { enumext / anskey }
2421 {
2422     break-col .bool_set:N = \l__enumext_store_columns_break_bool,

```

```

2423     break-col .default:n = true,
2424     break-col .value_forbidden:n = true,
2425     item-join .int_set:N = \l__enumext_store_item_join_int,
2426     item-join .value_required:n = true,
2427     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2428     item-star .default:n = true,
2429     item-star .value_forbidden:n = true,
2430     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2431     item-sym* .value_required:n = true,
2432     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2433     item-pos* .value_required:n = true,
2434     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2435 }

```

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

```

2436 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2437 {
2438     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2439 }
2440 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2441 {
2442     \tl_if_blank:nTF {#2}
2443     {
2444         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2445     }
2446     {
2447         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2448     }
2449 }

```

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

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

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

```

2450 \NewDocumentCommand \anskey { o +m }
2451 {
2452     \__enumext_anskey_safe_outer:
2453     \group_begin:
2454         \bool_if:NT \l__enumext_check_answers_bool
2455         {
2456             \tl_if_novalue:nF {#1}
2457             {
2458                 \keys_set:nn { enumext / anskey } {#1}
2459             }
2460             \tl_if_blank:nTF {#2}
2461             {
2462                 \msg_error:nn { enumext } { anskey-empty-arg }
2463             }
2464             {
2465                 \__enumext_anskey_safe_inner:
2466                 \__enumext_store_anskey_code:n {#2}
2467             }
2468         }
2469     \group_end:
2470 }

```

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

11.29.1 Internal functions for the command

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

```

2471 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2472 {
2473     \bool_if:NF \l__enumext_store_active_bool

```

```

2474     {
2475         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2476     }
2477     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2478     {
2479         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2480     }
2481     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2482     {
2483         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2484     }
2485     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2486     {
2487         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2488     }
2489 }

```

The `__enumext_anskey_safe_inner:` function will first check if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2490 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2491 {
2492     \int_incr:N \l__enumext_anskey_level_int
2493     \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2494     {
2495         \msg_error:nn { enumext } { anskey-nested }
2496     }
2497     \bool_if:NF \l__enumext_item_number_bool
2498     {
2499         \msg_error:nn { enumext } { anskey-unnumber-item }
2500     }
2501     \mode_if_math:T
2502     {
2503         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2504     }
2505 }

```

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

11.30 The environment `anskey*`

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

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

`__enumext_undefine_anskey_env:` The function `__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `__enumext_execute_after_env:` (§11.31) which is executed after the environment in which the key `save-ans` is active.

```

2506 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2507 {
2508     \cs_undefine:c { anskey* }
2509     \cs_undefine:c { endanskey* }
2510     \cs_undefine:c { __scontents_anskey*_env_begin: }
2511     \cs_undefine:c { __scontents_anskey*_env_end: }
2512 }

```

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

```

2513 \__enumext_before_env:nn { enumext }
2514 {
2515     \bool_lazy_and:nnT
2516     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2517     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }

```



```

2518     {
2519         \cs_if_free:cf { __scontents_anskey*_env_begin: }
2520         {
2521             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2522         }
2523     }
2524 }
2525 \__enumext_before_env:nn { enumext* }
2526 {
2527     \bool_lazy_and:nnT
2528     { \int_compare:p:nNn { \l__enumext_level_int } = { 0 } }
2529     { \int_compare:p:nNn { \l__enumext_level_h_int } = { 0 } }
2530     {
2531         \cs_if_free:cf { __scontents_anskey*_env_begin: }
2532         {
2533             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2534         }
2535     }
2536 }

```

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.

```

2537 \__enumext_before_env:nn { anskey* }
2538 {
2539     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2540     {
2541         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2542     }
2543     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2544     {
2545         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2546     }
2547     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2548     {
2549         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2550     }
2551     \bool_if:NF \l__enumext_item_number_bool
2552     {
2553         \msg_error:nn { enumext } { anskey-unnumber-item }
2554     }
2555     \mode_if_math:T
2556     {
2557         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2558     }
2559 }

```

(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: ($11.25.1)` and we will execute it only if the variable `\l__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package `scontents`.

```

2560 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2561 {
2562     \bool_if:NT \l__enumext_anskey_env_bool
2563     {
2564         \newenvsc{anskey*}[store-env=#1,print-env=false]
2565         \__enumext_anskey_env_exec:
2566     }
2567 }
2568 \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`.

```

2569 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2570 {

```

```

2571 \keys_define:nn { scontents / scontents }
2572 {
2573   break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2574   break-col .default:n = true,
2575   break-col .value_forbidden:n = true,
2576   item-join .int_gset:N = \g__enumext_store_item_join_int,
2577   item-join .value_required:n = true,
2578   item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2579   item-star .default:n = true,
2580   item-star .value_forbidden:n = true,
2581   item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2582   item-sym* .value_required:n = true,
2583   item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2584   item-pos* .value_required:n = true,
2585   print-env .undefine:,
2586   store-env .undefine:,
2587   write-out .undefine:,
2588   unknown .code:n = { \__enumext_anskey_env_unknown:n {##1} },
2589 }
2590 }

```

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

```

2591 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2592 {
2593   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2594 }
2595 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2596 {
2597   \tl_if_blank:nTF {#2}
2598   {
2599     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2600   }
2601   {
2602     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2603   }
2604 }

```

The function `__enumext_anskey_env_reset_keys:` will leave the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` undefined. We will apply this function using the *hook* function `__enumext_after_env:nn`.

```

2605 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2606 {
2607   \keys_define:nn { scontents / scontents }
2608   {
2609     break-col .undefine:,
2610     item-join .undefine:,
2611     item-star .undefine:,
2612     item-sym* .undefine:,
2613     item-pos* .undefine:,
2614     write-out .code:n = {
2615       \bool_set_false:N \l__scontents_storing_bool
2616       \bool_set_true:N \l__scontents_writing_bool
2617       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2618     },
2619     write-out .value_required:n = true,
2620     print-env .meta:nn = { scontents } { print-env = ##1 },
2621     print-env .default:n = true,
2622     store-env .meta:nn = { scontents } { store-env = ##1 },
2623     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2624   }
2625 }

```

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*.

```

2626 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2627 {
2628   \group_begin:
2629     \int_set:Nn \tex_newlinechar:D { `^^J }
2630     \__scontents_rescan_tokens:x
2631     {

```

```

2632         \endgroup % This assumes \catcode\|=0... Things might go off otherwise.
2633         #1
2634     }
2635 }

```

(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)`.

```

2636 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2637 {
2638     \__enumext_before_env:nn { anskey* }
2639     {
2640         \__enumext_anskey_env_define_keys:
2641     }

```

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.

```

2642     \hook_if_empty:nF {env/anskey*/after}
2643     {
2644         \hook_gremove_code:nn {env/anskey*/after} { * }
2645     }
2646     \__enumext_after_env:nn { anskey* }
2647     {
2648         \__enumext_anskey_env_save_keys:
2649         \tl_clear:N \l__enumext_store_anskey_env_tl
2650         \tl_clear:N \l__enumext_store_anskey_opt_tl
2651         \bool_if:NT \l__enumext_check_answers_bool
2652         {
2653             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2654             {
2655                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2656             }
2657             \regex_match:nVTF
2658             { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2659             \l__enumext_store_anskey_env_tl
2660             {
2661                 \msg_error:nn { enumext } { anskey-empty-arg }
2662             }
2663             {
2664                 \__enumext_anskey_env_store:
2665             }
2666         }
2667         \__enumext_anskey_env_clean_vars:
2668         \__enumext_anskey_env_reset_keys:
2669     }
2670 }

```

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

(End of definition for `__enumext_anskey_env_exec:`.)

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

```

2671 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2672 {
2673     \bool_lazy_and:nnT
2674     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2675     { \bool_not_p:n { \l__enumext_starred_bool } }
2676     {
2677         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2678     }

```

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

```

2679     \bool_lazy_and:nnT
2680     { \bool_not_p:n { \l__enumext_starred_bool } }
2681     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }

```

```

2682     {
2683         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2684         {
2685             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2686         }
2687     }

```

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

```

2688     \bool_if:NT \g__enumext_store_item_star_bool
2689     {
2690         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2691         {
2692             ,item-star,
2693         }
2694         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2695         {
2696             \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2697             {
2698                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2699             }
2700         }
2701         \dim_compare:nT
2702         {
2703             \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2704         }
2705         {
2706             \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2707             {
2708                 ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2709             }
2710         }
2711     }
2712 }

```

The function `__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `__enumext_store_anskey_code:n` and `__enumext_rescan_anskey_env:n`.

```

2713 \cs_new_protected:Nn \__enumext_anskey_env_store:
2714 {
2715     \group_begin:
2716     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2717     {
2718         \exp_args:Ne
2719         \__enumext_store_anskey_code:n
2720         {
2721             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2722         }
2723     }
2724     {
2725         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2726         \exp_args:Ne
2727         \__enumext_store_anskey_code:n
2728         {
2729             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2730         }
2731     }
2732     \group_end:
2733 }

```

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

```

2734 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2735 {
2736     \bool_gset_false:N \g__enumext_store_columns_break_bool
2737     \int_gzero:N \g__enumext_store_item_join_int
2738     \bool_gset_false:N \g__enumext_store_item_star_bool
2739     \tl_gclear:N \g__enumext_store_item_symbol_tl
2740     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2741 }

```

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

11.31 Executing anskey*, check-ans and write .log

__enumext_execute_after_env:

The __enumext_execute_after_env: function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the __enumext_item_answer_diff: function and then will write the values of the global variables used to the .log file. If the key `check-ans` is active it will execute the function __enumext_check_ans_show: and show the result in the terminal, otherwise it will execute the function __enumext_check_ans_log: and write the results in the .log file, undefine the environment `anskey*` (§11.30) through the function __enumext_undefine_anskey_env: and finally we execute the function __enumext_reset_global_vars: returning the used variables to their original state.

```

2742 \cs_new_protected:Nn \__enumext_execute_after_env:
2743 {
2744   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2745   {
2746     \tl_if_empty:NF \g__enumext_store_name_tl
2747     {
2748       \__enumext_stop_save_ans_msg:
2749       \__enumext_item_answer_diff:
2750       \__enumext_log_global_vars:
2751       \__enumext_log_answer_vars:
2752       \bool_if:NTF \g__enumext_check_ans_key_bool
2753       {
2754         \__enumext_check_ans_show:
2755       }
2756       { \__enumext_check_ans_log: }
2757       \__enumext_undefine_anskey_env:
2758     }
2759     \__enumext_reset_global_vars:
2760   }
2761 }

```

(End of definition for __enumext_execute_after_env:.)

- This function is passed to the function __enumext_after_env:nn for the environments `enumext` (§11.38) and `enumext*` (§11.42) and it is executed only when the environments are not nested or at some level of these..

11.32 Common functions for keyans, keyans* and keyanspic

11.32.1 Storing content in prop list

__enumext_keyans_addto_prop:n

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

```

2762 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2763 {
2764   \tl_clear:N \l__enumext_store_current_label_tl
2765   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2766   {
2767     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2768   }
2769   {
2770     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2771   }
2772   \tl_if_novalue:nF { #1 }
2773   {
2774     % Set save-sep
2775     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2776     {
2777       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2778     }
2779     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2780   }
2781   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2782 }

```

(End of definition for __enumext_keyans_addto_prop:n.)

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

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

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.

```

2783 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2784 {
2785   \bool_if:NT \l__enumext_store_ref_key_bool
2786   {
2787     \cs_set_protected:Npn \__enumext_tmp:n ##1
2788     {
2789       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2790       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2791       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2792       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2793     }
2794     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2795     \__enumext_keyans_store_ref_aux_i:
2796   }
2797 }

```

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`.

```

2798 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2799 {
2800   \bool_if:NT \g__enumext_starred_bool
2801   {
2802     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2803   }
2804   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2805   {
2806     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2807     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2808   }
2809   \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2810   {
2811     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2812     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2813   }
2814   \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2815   {
2816     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2817     { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
2818   }
2819   \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
2820   {
2821     \l__enumext_store_name_tl \c_colon_str
2822     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2823   }
2824   \__enumext_keyans_store_ref_aux_ii:
2825 }

```

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.

```

2826 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2827 {
2828   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2829   {
2830     \__enumext_newlabel:nn
2831     { \exp_not:V \__enumext_newlabel_arg_one_tl }
2832     { \__enumext_newlabel_arg_two_tl }
2833   }
2834   \l__enumext_write_aux_file_tl
2835 }

```

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

11.32.3 Storing content in sequence

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

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

```
2836 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2837 {
2838   \tl_clear:N \l__enumext_store_current_label_tl
2839   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2840   {
2841     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2842   }
2843   {
2844     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2845   }
2846   \tl_if_novalue:nF { #1 }
2847   {
2848     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2849     {
2850       \tl_put_right:Ne \l__enumext_store_current_label_tl
2851       {
2852         \l__enumext_store_keyans_item_opt_sep_tl
2853       }
2854     }
2855     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2856   }
2857   \__enumext_keyans_addto_seq_link:
2858 }
```

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.

```
2859 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2860 {
2861   \bool_lazy_and:nnT
2862   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2863   { \bool_if_p:N \l__enumext_hyperref_bool }
2864   {
2865     \tl_put_right:Ne \l__enumext_store_current_label_tl
2866     {
2867       \hfill \exp_not:N \hyperlink
2868       {
2869         \exp_not:V \l__enumext_newlabel_arg_one_tl
2870       }
2871       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2872     }
2873   }
2874   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2875   \bool_if:NT \l__enumext_check_answers_bool
2876   {
2877     \int_gincr:N \g__enumext_item_anskey_int
2878   }
2879 }
```

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

11.32.4 The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the *\anskey* code, but, if I change the order of the operations the counter off *⟨label⟩* are incorrect.

```
\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:
```

Common function to show *starred commands* *\item** and *⟨position⟩* of stored content in *⟨prop list⟩* for *keyans* and *keyanspic*. Need add 1 to `\g__enumext_⟨store name⟩_prop` for *show-pos* key.

```
2880 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2881 {
2882   \tl_if_novalue:nF { #1 }
2883   {
```



```

2884         \tl_set:Nc \l__enumext_store_current_opt_arg_tl { #1 }
2885     }
2886     \bool_if:NT \l__enumext_show_answer_bool
2887     {
2888         \__enumext_keyans_show_ans:
2889     }
2890     \bool_if:NT \l__enumext_show_position_bool
2891     {
2892         \__enumext_keyans_show_pos:
2893     }
2894 }
2895 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2896 {
2897     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2898     {
2899         \bool_lazy_or:nnT
2900         { \bool_if_p:N \l__enumext_show_answer_bool }
2901         { \bool_if_p:N \l__enumext_show_position_bool }
2902         {
2903             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2904         }
2905     }
2906 }
2907 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2908 {
2909     \tl_put_left:Nn \l__enumext_label_v_tl
2910     {
2911         \__enumext_print_keyans_box:NN
2912         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2913     }
2914 }
2915 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2916 {
2917     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2918     {
2919         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2920         {
2921             \group_begin:
2922             \exp_not:N \normalfont
2923             \exp_not:N \footnotesize [ \int_eval:n
2924                 {
2925                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2926                 }
2927             ]
2928             \group_end:
2929         }
2930     }
2931     {
2932         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2933         {
2934             \group_begin:
2935             \exp_not:N \normalfont
2936             \exp_not:N \footnotesize [ \int_eval:n
2937                 {
2938                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2939                 }
2940             ]
2941             \group_end:
2942         }
2943     }
2944     \tl_put_left:Nn \l__enumext_label_v_tl
2945     {
2946         \__enumext_print_keyans_box:NN
2947         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2948     }
2949 }

```

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

11.33 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*
2950 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2951 {
2952   \keys_define:nn { enumext / #1 }
2953   {
2954     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2955     item-sym* .value_required:n = true,
2956     item-sym* .initial:n = { $\star$ },
2957     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2958     item-pos* .value_required:n = true,
2959   }
2960 }
2961 \clist_map_inline:nn
2962 {
2963   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2964 }
2965 { \__enumext_tmp:nn #1 }

```

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

11.34 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](#).

```

2966 \cs_new_protected:Nn \__enumext_footnotetext:nn
2967 {
2968   \footnotetext[#1]{#2}
2969 }
2970 \cs_new_protected:Nn \__enumext_renew_footnote:
2971 {
2972   \seq_gclear:N \__enumext_footnote_arg_seq
2973   \seq_gclear:N \__enumext_footnote_int_seq
2974   \RenewDocumentCommand \footnote { o +m }
2975   {
2976     \tl_if_novalue:nTF {##1}
2977     {
2978       \stepcounter{footnote}
2979       \int_gset_eq:Nc \__enumext_footnote_int { c@footnote }
2980     }
2981     {
2982       \int_gset:Nn \__enumext_footnote_int { ##1 }
2983     }
2984     \footnotemark [ \__enumext_footnote_int ]
2985     \seq_gput_right:Nn \__enumext_footnote_arg_seq { ##2 }
2986     \seq_gput_right:NV \__enumext_footnote_int_seq \__enumext_footnote_int
2987   }
2988 }
2989 \cs_new_protected:Nn \__enumext_print_footnote:
2990 {
2991   \seq_if_empty:NF \__enumext_footnote_int_seq
2992   {
2993     \seq_map_pairwise_function:NNN
2994     \__enumext_footnote_int_seq
2995     \__enumext_footnote_arg_seq
2996     \__enumext_footnotetext:nn
2997   }
2998 }

```

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

11.35 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.

11.35.1 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:` (§11.36).

```

2999 \cs_new_protected:Npn \__enumext_default_item:n #1
3000 {
3001   \tl_if_novalue:nTF {#1}
3002   {
3003     \bool_if:NT \l__enumext_check_answers_bool
3004     {
3005       \int_gincr:N \g__enumext_item_number_int
3006       \bool_set_true:N \l__enumext_item_number_bool
3007     }
3008     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3009     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3010   }
3011   {
3012     \bool_set_eq:cc
3013     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3014     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3015     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3016   }
3017 }

```

(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:` (§11.36).

```

3018 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3019 {
3020   \tl_if_novalue:nF {#1}
3021   {
3022     \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
3023   }
3024   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3025   \tl_if_novalue:nTF {#2}
3026   {
3027     \dim_set_eq:cc
3028     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3029     { \l__enumext_labelsep_ \__enumext_level: _dim }
3030   }
3031   {
3032     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3033   }
3034   \bool_if:NT \l__enumext_check_answers_bool
3035   {

```

```

3036         \int_gincr:N \g__enumext_item_number_int
3037         \bool_set_true:N \l__enumext_item_number_bool
3038     }
3039     \bool_set_true:c { \l__enumext_wrap_label_ \l__enumext_level: _bool }
3040     \l__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \l__enumext_level: _tl }
3041 }

```

(End of definition for \l__enumext_starred_item:nn.)

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

This function is passed to `\l__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.37.2).

```

3042 \cs_new_protected:Nn \l__enumext_redefine_item:
3043 {
3044     \RenewDocumentCommand \item { s o o }
3045     {
3046         \bool_if:nTF {##1}
3047         {
3048             \l__enumext_starred_item:nn {##2} {##3}
3049         }
3050         { \l__enumext_default_item:n {##2} }
3051     }
3052 }

```

(End of definition for \l__enumext_redefine_item:.)

11.35.2 The `\item` command in keys

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.

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

```

3053 \cs_new_protected:Npn \l__enumext_keyans_default_item:n #1
3054 {
3055     \tl_if_novalue:nTF { #1 }
3056     {
3057         \bool_set_true:N \l__enumext_wrap_label_v_bool
3058         \l__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3059     }
3060     {
3061         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3062         \l__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3063     }
3064 }

```

(End of definition for \l__enumext_keyans_default_item:n.)

`\l__enumext_keyans_starred_item:n` The function `\l__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 `\l__enumext_keyans_show_left:n` and will display the contents of that item using the internal copy `\l__enumext_item_std:w`, this is necessary to prevent incrementing the current “counter” of the original `\label`.

```

3065 \cs_new_protected:Npn \l__enumext_keyans_starred_item:n #1
3066 {
3067     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3068     \l__enumext_keyans_show_left:n { #1 }
3069     \bool_set_true:N \l__enumext_wrap_label_v_bool
3070     \l__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \l__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`.

```

3071     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3072     \l__enumext_keyans_addto_prop:n { #1 }
3073     \l__enumext_keyans_store_ref:
3074     \l__enumext_keyans_addto_seq:n { #1 }
3075     \int_gincr:N \g__enumext_check_starred_cmd_int
3076 }

```

(End of definition for \l__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 (§11.37.2).

```

3077 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3078 {
3079   \RenewDocumentCommand \item { s o }
3080   {
3081     \bool_if:nTF {##1}
3082     {
3083       \peek_remove_spaces:n
3084       {
3085         \__enumext_keyans_starred_item:n {##2}
3086       }
3087     }
3088     {
3089       \__enumext_keyans_default_item:n {##2}
3090     }
3091   }
3092 }

```

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

11.36 Redefining `\makeLabel` command

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

11.36.1 Redefining `\makeLabel` for `enumext`

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

```

3093 \cs_new_protected:Nn \__enumext_item_starred:
3094 {
3095   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3096   {
3097     \mode_leave_vertical:
3098     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3099     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
3100     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3101   }
3102 }

```

(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 (§11.37.2).

```

3103 \cs_new_protected:Nn \__enumext_make_label:
3104 {
3105   \RenewDocumentCommand \makeLabel { m }
3106   {
3107     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
3108     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
3109     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
3110     {
3111       \__enumext_item_starred:
3112       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3113     }
3114     { ##1 }
3115     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
3116     \tl_gclear:N \g__enumext_item_symbol_tl
3117   }
3118 }

```

(End of definition for `__enumext_make_label:`.)

11.36.2 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 (§11.37.2).

```

3119 \cs_new_protected:Nn \__enumext_keyans_make_label:
3120 {
3121   \RenewDocumentCommand \makeLabel { m }
3122   {
3123     \tl_use:N \l__enumext_label_fill_left_v_tl
3124     \tl_use:N \l__enumext_label_font_style_v_tl
3125     \bool_if:NTF \l__enumext_wrap_label_v_bool
3126     {
3127       \__enumext_wrapper_label_v:n { ##1 }
3128     }
3129     { ##1 }
3130     \tl_use:N \l__enumext_label_fill_right_v_tl
3131   }
3132 }

```

(End of definition for `__enumext_keyans_make_label:`.)

11.37 Second argument of the lists

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

11.37.1 Calculation of `\leftmargin` and `\itemindent`

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

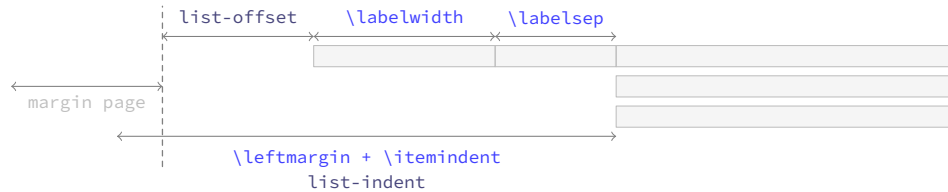


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

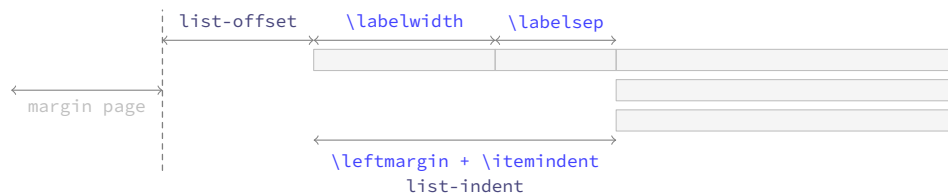


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

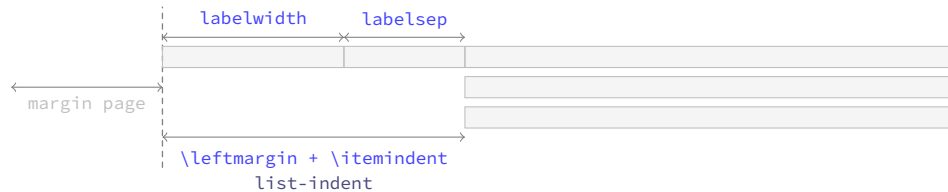


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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 (§11.37.2).

```

3133 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3134 {
3135   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3136   {
3137     \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3138     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3139   }
3140   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3141   {
3142     \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3143     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3144   }

```

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

```

3145   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3146   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3147   {
3148     \dim_set:Nn #6 { #1 + #2 - #4 }
3149     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3150   }
3151   {
3152     \dim_compare:nNnT { #4 } = { #1 + #2 }
3153     { \dim_set:Nn #6 { \c_zero_dim } }
3154     \dim_compare:nNnT { #4 } < { #1 + #2 }
3155     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3156     \dim_compare:nNnT { #4 } > { #1 + #2 }
3157     {
3158       \dim_set:Nn #6 { -#1 - #2 + #4 }
3159       \dim_set:Nn #6 { #6*-1 }
3160     }
3161     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3162   }
3163 }
3164 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

11.37.2 Setting second argument of the lists

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

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3165 \cs_set_protected:Npn \__enumext_tmp:n #1
3166 {
3167   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3168   {
3169     \__enumext_calc_hspace:ccccc
3170     { \l__enumext_labelwidth_#1_dim } { \l__enumext_labelsep_#1_dim }
3171     { \l__enumext_listoffset_#1_dim } { \l__enumext_leftmargin_tmp_#1_dim }
3172     { \l__enumext_leftmargin_#1_dim } { \l__enumext_itemindent_#1_dim }
3173     { \l__enumext_leftmargin_tmp_#1_bool }
3174     \clist_map_inline:nn
3175     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3176     { \dim_set_eq:cc {####1} { \l__enumext_####1_#1_dim } }
3177     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3178     { \skip_set_eq:cc {####1} { \l__enumext_####1_#1_skip } }
3179     \usecounter { enumX#1 }
3180     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \l__enumext_start_#1_int } - 1 } }
3181     \str_if_eq:nnTF {#1} { v }
3182     {
3183       \__enumext_keyans_redefine_item:
3184       \__enumext_keyans_make_label:
3185       \__enumext_keyans_ref:
3186       \__enumext_keyans_fake_item:
3187       \bool_if:cT { \l__enumext_show_length_#1_bool }
3188       {

```



```

3189         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3190     }
3191 }
3192 {
3193     \__enumext_redefine_item:
3194     \__enumext_make_label:
3195     \__enumext_standar_ref:
3196     \__enumext_fake_item:
3197     \bool_if:cT { l__enumext_show_length_#1_bool }
3198     {
3199         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3200     }
3201 }
3202 }
3203 }
3204 \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: 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.

```

3205 \cs_set_protected:Npn \__enumext_tmp:n #1
3206 {
3207     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3208     {
3209         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3210         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3211         \__enumext_calc_hspace:ccccc
3212         { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3213         { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3214         { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3215         { l__enumext_leftmargin_tmp_#1_bool }
3216         \clist_map_inline:nn
3217         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3218         { \dim_set_eq:cc {###1} { l__enumext_###1_#1_dim } }
3219         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3220         { \skip_set_eq:cc {###1} { l__enumext_###1_#1_skip } }
3221         \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3222         \skip_zero:N \partopsep
3223         \usecounter { enumX#1 }
3224         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3225         \__enumext_starred_ref:
3226         \str_if_eq:nnTF {#1} { vii }
3227         {
3228             \__enumext_fake_item_vii:
3229             \bool_if:cT { l__enumext_show_length_vii_bool }
3230             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3231         }
3232         {
3233             \__enumext_fake_item_viii:
3234             \bool_if:cT { l__enumext_show_length_#1_bool }
3235             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3236         }
3237     }
3238 }
3239 \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:.)

11.38 The environment enumext

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

```

3240 \NewDocumentEnvironment{enumext}{0}{}
3241 {
3242     \__enumext_safe_exec:
3243     \__enumext_parse_keys:n {#1}
3244     \__enumext_before_list:
3245     \__enumext_start_store_level:
3246     \__enumext_start_list:nn
3247     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }

```

```

3248     {
3249         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3250         \__enumext_before_keys_exec:
3251     }
3252     \__enumext_after_args_exec:
3253 }
3254 {
3255     \__enumext_stop_list:
3256     \__enumext_stop_store_level:
3257     \__enumext_after_list:
3258 }

```

(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:`.

```

3259 \cs_new_protected:Nn \__enumext_safe_exec:
3260 {
3261     \__enumext_internal_mini_page:
3262     \__enumext_is_not_nested:
3263     \int_incr:N \l__enumext_level_int
3264     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3265     { \msg_fatal:nn { enumext } { list-too-deep } }
3266     \bool_set_true:N \l__enumext_standar_bool
3267     \__enumext_is_on_first_level:
3268 }

```

(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`.

```

3269 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3270 {
3271     \tl_if_novalue:nF {#1}
3272     {
3273         \str_clear:N \l__enumext_series_str
3274         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3275         {
3276             \keys_set:nn { enumext / level-1 } {#1}
3277             \__enumext_parse_series:n {#1}
3278             \__enumext_nested_base_line_fix:
3279         }
3280         {
3281             \exp_args:Ne \keys_set:nn
3282             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3283         }
3284         \__enumext_store_active_keys:n {#1}
3285     }
3286 }

```

(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:`

```

3287 \cs_new_protected:Nn \__enumext_start_store_level:
3288 {
3289     \bool_lazy_all:nT
3290     {
3291         { \bool_if_p:N \l__enumext_store_active_bool }
3292         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3293         { \bool_if_p:N \g__enumext_standar_bool }
3294     }

```

```

3295     {
3296         \int_compare:nNt { \l__enumext_level_int } > { 1 }
3297         {
3298             \bool_set_true:c { \l__enumext_store_upper_level_ \l__enumext_level: _bool }
3299             \l__enumext_store_level_open:
3300         }
3301     }

```

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

```

3302     \bool_lazy_all:nT
3303     {
3304         { \bool_if_p:N \l__enumext_store_active_bool }
3305         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3306         { \int_compare_p:nNt { \l__enumext_level_h_int } = { 1 } }
3307     }
3308     {
3309         \int_compare:nNt { \l__enumext_level_int } > { 0 }
3310         {
3311             \bool_set_true:c { \l__enumext_store_upper_level_ \l__enumext_level: _bool }
3312             \l__enumext_store_level_open:
3313         }
3314     }
3315 }
3316 \cs_new_protected:Nn \l__enumext_stop_store_level:
3317 {
3318     \bool_if:cT { \l__enumext_store_upper_level_ \l__enumext_level: _bool }
3319     {
3320         \l__enumext_store_level_close:
3321     }
3322 }

```

(End of definition for `\l__enumext_start_store_level:` and `\l__enumext_stop_store_level:`)

`\l__enumext_before_list:` The function `\l__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.

```

3323 \cs_new_protected:Nn \l__enumext_before_list:
3324 {
3325     \l__enumext_vspace_above:
3326     \l__enumext_before_args_exec:

```

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

```

3327     \l__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 `\l__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 `\l__enumext_mini_env*` environment on the “left side”, always having a current `\linewidth` as *maximum width* between them.

```

3328     \dim_compare:nNt
3329     { \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim } } > { \c_zero_dim }
3330     {
3331         \dim_set:cn { \l__enumext_minipage_left_ \l__enumext_level: _dim }
3332         {
3333             \linewidth
3334             - \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim }
3335             - \dim_use:c { \l__enumext_minipage_hsep_ \l__enumext_level: _dim }
3336         }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `\l__enumext_mini_addvspace:` is called and the `\l__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 `\l__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.

```

3337         \bool_set_true:c { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3338         \int_gincr:N \g__enumext_minipage_stat_int
3339         \l__enumext_mini_addvspace:
3340         \nointerlineskip\noindent

```

```

3341         \begin{__enumext_mini_env*}
3342         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3343     }
3344     \__enumext_multicols_start:
3345 }

```

(End of definition for __enumext_before_list:.)

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

```

3346 \cs_new_protected:Nn \__enumext_multicols_start:
3347 {
3348     \int_compare:nNtT
3349     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3350     {
3351         \dim_compare:nNtT
3352         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3353         {
3354             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3355             {
3356                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3357                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3358                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3359                   - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3360             }
3361         }
3362         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3363         \skip_zero:N \multicolsep
3364         \int_compare:nNtT { \l__enumext_level_int } > { 1 }
3365         {
3366             \dim_zero:N \columnseprule
3367         }
3368     }
3369 }

```

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.

```

3368     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3369     {
3370         \__enumext_multi_addvspace:
3371     }
3372     \raggedcolumns
3373     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3374 }
3375 }

```

(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.

```

3376 \cs_new_protected:Nn \__enumext_multicols_stop:
3377 {
3378     \int_compare:nNtT
3379     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3380     {
3381         \end{multicols}
3382         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3383         {
3384             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3385         }
3386     }
3387 }

```

(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.

```

3388 \cs_new_protected:Nn \__enumext_after_list:
3389 {
3390   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3391   {
3392     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3393     {
3394       \msg_warning:nn { enumext } { missing-miniright }
3395       \miniright
3396     }
3397     \int_gzero:N \g__enumext_minipage_stat_int
3398     \end{__enumext_mini_env*}
3399     \par\addvspace { \l__enumext_minipage_after_skip }
3400   }
3401   { \__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`.

```

3402   \__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.

```

3403   \__enumext_after_stop_list:
3404   \__enumext_vspace_below:
3405   \bool_set_false:N \l__enumext_standar_bool
3406   \__enumext_resume_save_counter:
3407 }

```

(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`.

```

3408 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

11.39 The environment `keyans`

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

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

```

3409 \NewDocumentEnvironment{keyans}{0}{}
3410 {
3411   \__enumext_keyans_safe_exec:
3412   \__enumext_keyans_parse_keys:n {#1}
3413   \__enumext_before_list_v:
3414   \__enumext_start_list:nn
3415   { \tl_use:N \l__enumext_label_v_tl }
3416   {
3417     \__enumext_list_arg_two_v:
3418     \__enumext_before_keys_exec_v:
3419   }
3420   \__enumext_after_args_exec_v:
3421 }
3422 {
3423   \__enumext_check_starred_cmd:n { item }
3424   \__enumext_stop_list:
3425   \__enumext_after_list_v:
3426 }

```

(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.

```

3427 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3428 {
3429   \bool_if:NF \l__enumext_store_active_bool
3430   {

```

```

3431     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3432   }
3433   \int_incr:N \l__enumext_keyans_level_int
3434   \bool_set_true:N \l__enumext_keyans_env_bool
3435   \__enumext_keyans_start_line:
3436   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3437   \bool_set_false:N \l__enumext_store_active_bool
3438   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3439   {
3440     \msg_error:nn { enumext } { keyans-nested }
3441   }
3442   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3443   {
3444     \msg_error:nn { enumext } { keyans-wrong-level }
3445   }
3446 }

```

(End of definition for __enumext_keyans_safe_exec:.)

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

```

3447 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3448 {
3449   \keys_set:nn { enumext / keyans } {#1}
3450 }

```

(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.

```

3451 \cs_new_protected:Nn \__enumext_before_list_v:
3452 {
3453   \__enumext_vspace_above_v:
3454   \__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.

```

3455   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3456   {
3457     \dim_set:Nn \l__enumext_minipage_left_v_dim
3458     {
3459       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3460     }

```

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 T_EX macro \no_{interlineskip} to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3461   \bool_set_true:N \l__enumext_minipage_active_v_bool
3462   \int_gincr:N \g__enumext_minipage_stat_int
3463   \__enumext_keyans_mini_addvspace:
3464   \nointerlineskip\noindent
3465   \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3466 }

```

After these actions, the __enumext_keyans_multicols_start: function is called to handle the *multicols* environment.

```

3467   \__enumext_keyans_multicols_start:
3468 }

```

(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.

```

3469 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3470 {
3471   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3472   {

```

Set the default value for `\columnsep` when `columns-sep` key is `opt`.

```

3473 \dim_compare:nNtT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3474 {
3475   \dim_set:Nn \l__enumext_columns_sep_v_dim
3476   {
3477     (
3478       \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3479     ) / \l__enumext_columns_v_int
3480     - \l__enumext_listoffset_v_dim
3481   }
3482 }
3483 \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).

```

3484 \skip_zero:N \multicolsep
3485 \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.

```

3486 \bool_if:NF \l__enumext_minipage_active_v_bool
3487 {
3488   \__enumext_keyans_multi_addvspace:
3489 }
3490 \raggedcolumns
3491 \begin{multicols}{\l__enumext_columns_v_int }
3492 }
3493 }

```

(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.

```

3494 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3495 {
3496   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3497   {
3498     \end{multicols}
3499     \bool_if:NF \l__enumext_minipage_active_v_bool
3500     {
3501       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3502     }
3503   }
3504 }

```

(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.

```

3505 \cs_new_protected:Nn \__enumext_after_list_v:
3506 {
3507   \bool_if:NTF \l__enumext_minipage_active_v_bool
3508   {
3509     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3510     {
3511       \msg_warning:nn { enumext } { missing-miniright }
3512       \miniright
3513     }
3514     \int_gzero:N \g__enumext_minipage_stat_int
3515     \end{__enumext_mini-env*}
3516     \par\addvspace{ \l__enumext_minipage_after_skip }
3517   }
3518   { \__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.

```

3519 \bool_set_false:N \l__enumext_keyans_env_bool
3520 \__enumext_after_stop_list_v:
3521 \__enumext_vspace_below_v:
3522 }

```

(End of definition for `__enumext_after_list_v:`)

11.40 The environment `keyanspic` and `\anspic`

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

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

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

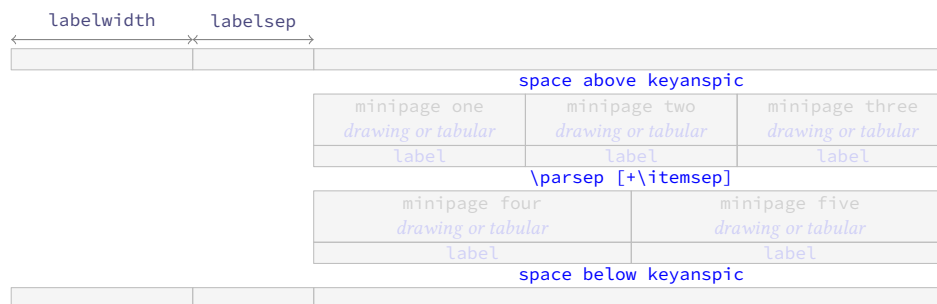


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

11.40.1 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.

```

3523 \NewDocumentCommand \anspic { s o +m }
3524 {

```

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

```

3525 \bool_if:NF \l__enumext_store_active_bool
3526 {
3527 \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3528 }
3529 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3530 {
3531 \msg_error:nn { enumext } { keyanspic-wrong-level }
3532 }
3533 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3534 {
3535 \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3536 }

```

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.

```

3537 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3538 {
3539 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3540 }
3541 }

```

(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.

```

3542 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3543 {
3544 \stepcounter { enumXvi }

```

```

3545 #3 \\
3546 \bool_if:nT { #1 }
3547 {
3548   \__enumext_keyans_addto_prop:n { #2 }
3549   \__enumext_keyans_store_ref:
3550   \__enumext_keyans_addto_seq:n { #2 }
3551   \int_gincr:N \g__enumext_check_starred_cmd_int
3552   \bool_lazy_or:nnT
3553   { \bool_if_p:N \l__enumext_show_answer_bool }
3554   { \bool_if_p:N \l__enumext_show_position_bool }
3555   {
3556     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3557     \__enumext_keyans_show_left:n { #2 }
3558     \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3559   }
3560 }
3561 \tl_use:N \l__enumext_label_font_style_v_tl
3562 \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3563 }

```

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

11.40.2 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.

```

3564 \NewDocumentEnvironment{keyanspic}{o}
3565 {
3566   \__enumext_keyans_pic_safe_exec:
3567   \__enumext_start_list:nn
3568   { }
3569   {
3570     \__enumext_keyans_pic_arg_two:
3571   }

```

We apply the “adjusted” vertical spacing above the environment

```

3572 \vspace { \l__enumext_keyans_pic_above_skip }
3573 }

```

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.

```

3574 {
3575   \tl_if_novalue:nTF { #1 }
3576   {
3577     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3578   }
3579   { \__enumext_keyans_pic_do:n { #1 } }
3580   \__enumext_stop_list:
3581   \__enumext_check_starred_cmd:n { anspic }
3582   \setcounter { enumXvi } { 0 }
3583   \vspace { \l__enumext_topsep_v_skip }
3584   %\bool_set_false:N \l__enumext_store_active_bool
3585 }

```

(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.

```

3586 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3587 {
3588   \int_incr:N \l__enumext_keyans_pic_level_int
3589   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3590   {
3591     \msg_error:nn { enumext } { keyanspic-nested }
3592   }
3593   \__enumext_keyans_start_line:
3594 }

```

(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`.

```

3595 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3596 {
3597   \dim_compare:nNtT { #1 } < { 0pt }
3598   { \skip_set:Nn #1 { -#1 } }
3599 }

```

(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.

```

3600 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3601 {

```

The first thing to do is to set the boolean variable `__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.

```

3602   \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
3603   \__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.

```

3604   \skip_add:Nn \parsep { \itemsep }
3605   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3606   \dim_zero:N \labelwidth
3607   \dim_zero:N \listparindent
3608   \dim_zero:N \labelsep
3609   \skip_zero:N \partopsep
3610   \skip_zero:N \itemsep

```

We set the value of `__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.

```

3611   \__enumext_keyans_pic_skip_abs:N \parsep
3612   \skip_set:Nn \__enumext_keyans_pic_above_skip
3613   {
3614     \box_dp:N \strutbox
3615     + \__enumext_topsep_v_skip
3616     - \parsep
3617   }
3618   \__enumext_item_std:w \scan_stop:
3619 }

```

(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`.

```

3620 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3621 {
3622   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3623 }
3624 \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 `__enumext_keyans_pic_body_seq` sequence inside them.

```

3625 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3626 {
3627   \dim_set:Nn \__enumext_keyans_pic_width_dim { \linewidth / #1 }
3628   \int_set:Nn \__enumext_keyans_pic_above_int { \__enumext_keyans_pic_below_int }
3629   \int_set:Nn \__enumext_keyans_pic_below_int { \__enumext_keyans_pic_above_int + #1 }
3630   \int_step_inline:nnn
3631   { \__enumext_keyans_pic_above_int + 1 }
3632   { \__enumext_keyans_pic_below_int }
3633   {
3634     \__enumext_minipage:w [ b ]{ \__enumext_keyans_pic_width_dim }
3635     \centering

```

```

3636         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3637     \__enumext_endminipage:
3638 }
3639 \par
3640 }

```

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

11.41 The horizontal environments

Generating horizontal list environments is NOT as simple as standard \TeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

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`).

11.41.1 Functions for item box width

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:

```

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

```

3641 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3642 {
3643     \dim_compare:nNt { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3644     {
3645         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3646         {
3647             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3648             / \l__enumext_columns_vii_int
3649         }
3650     }
3651     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3652     \dim_set:Nn \l__enumext_item_width_vii_dim
3653     {
3654         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3655         / \l__enumext_columns_vii_int
3656         - \l__enumext_labelwidth_vii_dim
3657         - \l__enumext_labelsep_vii_dim
3658     }

```

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

```

3659     \dim_compare:nNt { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3660     {
3661         \dim_sub:Nn \l__enumext_item_width_vii_dim
3662         {
3663             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
3664             / \l__enumext_columns_vii_int
3665         }
3666         \dim_add:Nn \l__enumext_columns_sep_vii_dim
3667         {
3668             \l__enumext_rightmargin_vii_dim
3669         }
3670     }
3671 }

```

Same implementation for the `keyans*` environment.

```

3672 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3673 {
3674     \dim_compare:nNt { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3675     {
3676         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3677         {
3678             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3679             / \l__enumext_columns_viii_int
3680         }
3681     }
3682     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3683     \dim_set:Nn \l__enumext_item_width_viii_dim
3684     {

```

```

3685     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3686     / \l__enumext_columns_viii_int
3687     - \l__enumext_labelwidth_viii_dim
3688     - \l__enumext_labelsep_viii_dim
3689   }
3690   \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3691   {
3692     \dim_sub:Nn \l__enumext_item_width_viii_dim
3693     {
3694       ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3695       / \l__enumext_columns_viii_int
3696     }
3697     \dim_add:Nn \l__enumext_columns_sep_viii_dim
3698     {
3699       \l__enumext_rightmargin_viii_dim
3700     }
3701   }
3702 }

```

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

11.41.2 Functions for join item columns

`__enumext_starred_joined_item_vii:n`
`__enumext_starred_joined_item_viii:n`

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the content passed to `\item(<columns>)` will be stored together with the value of `\itemwidthdim` for the `enumext*` environment.

```

3703 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3704 {
3705   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3706   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3707   {
3708     \msg_warning:nnee { enumext } { item-joined }
3709     { \int_use:N \l__enumext_joined_item_vii_int }
3710     { \int_use:N \l__enumext_columns_vii_int }
3711     \int_set:Nn \l__enumext_joined_item_vii_int
3712     {
3713       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3714     }
3715   }
3716   \int_compare:nNnT
3717   { \l__enumext_joined_item_vii_int }
3718   >
3719   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3720   {
3721     \msg_warning:nnee { enumext } { item-joined-columns }
3722     { \int_use:N \l__enumext_joined_item_vii_int }
3723     {
3724       \int_eval:n
3725       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3726     }
3727     \int_set:Nn \l__enumext_joined_item_vii_int
3728     {
3729       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3730     }
3731   }
3732   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3733   {
3734     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3735     \int_decr:N \l__enumext_joined_item_aux_vii_int
3736     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3737     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3738     \dim_set:Nn \l__enumext_joined_width_vii_dim
3739     {
3740       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3741       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3742         + \l__enumext_columns_sep_vii_dim
3743         ) * \l__enumext_joined_item_aux_vii_int
3744     }
3745     \dim_set_eq:NN \itemwidthdim \l__enumext_joined_width_vii_dim
3746   }
3747   {
3748     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim

```

```

3749         \dim_set_eq:NN \itemwidthdim \l__enumext_item_width_vii_dim
3750     }
3751 }

```

Same implementation for the `keyans*` environment.

```

3752 \cs_new_protected:Npn \l__enumext_starred_joined_item_viii:n #1
3753 {
3754     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3755     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3756     {
3757         \msg_warning:nnee { enumext } { item-joined }
3758         { \int_use:N \l__enumext_joined_item_viii_int }
3759         { \int_use:N \l__enumext_columns_viii_int }
3760         \int_set:Nn \l__enumext_joined_item_viii_int
3761         {
3762             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3763         }
3764     }
3765     \int_compare:nNnT
3766     { \l__enumext_joined_item_viii_int }
3767     >
3768     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3769     {
3770         \msg_warning:nnee { enumext } { item-joined-columns }
3771         { \int_use:N \l__enumext_joined_item_viii_int }
3772         {
3773             \int_eval:n
3774             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3775         }
3776         \int_set:Nn \l__enumext_joined_item_viii_int
3777         {
3778             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3779         }
3780     }
3781     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3782     {
3783         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3784         \int_decr:N \l__enumext_joined_item_aux_viii_int
3785         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3786         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3787         \dim_set:Nn \l__enumext_joined_width_viii_dim
3788         {
3789             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3790             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3791                 + \l__enumext_columns_sep_viii_dim
3792                 ) * \l__enumext_joined_item_aux_viii_int
3793         }
3794         \dim_set_eq:NN \itemwidthdim \l__enumext_joined_width_viii_dim
3795     }
3796     {
3797         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3798         \dim_set_eq:NN \itemwidthdim \l__enumext_item_width_viii_dim
3799     }
3800 }

```

(End of definition for `\l__enumext_starred_joined_item_vii:n` and `\l__enumext_starred_joined_item_viii:n`)

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

`\l__enumext_start_mini_vii:` The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `\l__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`.

```

3801 \cs_new_protected:Nn \l__enumext_start_mini_vii:
3802 {
3803     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3804     {
3805         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3806         {
3807             \linewidth
3808             - \l__enumext_minipage_right_vii_dim
3809             - \l__enumext_minipage_hsep_vii_dim

```

```

3810     }
3811     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3812     \dim_gset_eq:NN
3813         \g__enumext_minipage_right_vii_dim
3814         \l__enumext_minipage_right_vii_dim
3815     \__enumext_mini_addvspace_vii:
3816     \nointerlineskip\noindent
3817     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3818   }
3819 }

```

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:nn` to execute the `__enumext_mini_env*` on the “right side”.

```

3820 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3821 {
3822   \bool_if:NT \l__enumext_minipage_active_vii_bool
3823   {
3824     \end{__enumext_mini_env*}
3825     \hfill
3826     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3827   }
3828 }

```

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.

```

3829 \__enumext_after_env:nn {enumext*}
3830 {
3831   \bool_if:NT \g__enumext_minipage_active_vii_bool
3832   {
3833     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3834     \par\addvspace { \g__enumext_minipage_right_skip }
3835     \bool_if:NF \g__enumext_minipage_center_vii_bool
3836     {
3837       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3838       {
3839         \centering
3840       }
3841     }
3842     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3843     {
3844       \tl_use:N \g__enumext_miniright_code_vii_tl
3845     }
3846     \box_use_drop:N \l__enumext_miniright_code_vii_box
3847     \end{__enumext_mini_env*}
3848     \par\addvspace{ \g__enumext_minipage_after_skip }
3849   }
3850   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3851   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3852   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3853   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3854   \bool_gset_false:N \g__enumext_starred_bool
3855 }

```

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

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

`__enumext_stop_mini_viii:`

```

3856 \cs_new_protected:Nn \__enumext_start_mini_viii:
3857 {
3858   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3859   {
3860     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3861     {
3862       \linewidth
3863       - \l__enumext_minipage_right_viii_dim
3864       - \l__enumext_minipage_hsep_viii_dim
3865     }

```



```

3866         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3867         \dim_gset_eq:NN
3868             \g__enumext_minipage_right_viii_dim
3869             \l__enumext_minipage_right_viii_dim
3870         \__enumext_mini_addvspace_viii:
3871         \nointerlineskip\noindent
3872         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3873     }
3874 }
3875 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3876 {
3877     \bool_if:NT \l__enumext_minipage_active_viii_bool
3878     {
3879         \end{__enumext_mini_env*}
3880         \hfill
3881         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3882     }
3883 }
3884 \__enumext_after_env:nn {keyans*}
3885 {
3886     \bool_if:NT \g__enumext_minipage_active_viii_bool
3887     {
3888         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3889         \par\addvspace { \g__enumext_minipage_right_skip }
3890         \bool_if:NF \g__enumext_minipage_center_viii_bool
3891         {
3892             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3893             {
3894                 \centering
3895             }
3896         }
3897         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3898         {
3899             \tl_use:N \g__enumext_miniright_code_viii_tl
3900         }
3901         \box_use_drop:N \l__enumext_miniright_code_viii_box
3902         \end{__enumext_mini_env*}
3903         \par\addvspace{ \g__enumext_minipage_after_skip }
3904     }
3905     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3906     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3907     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3908     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3909 }

```

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

11.42 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.

```

3910 \NewDocumentEnvironment{enumext*}{ o }
3911 {
3912     \__enumext_safe_exec_vii:
3913     \__enumext_parse_keys_vii:n {#1}
3914     \__enumext_before_list_vii:
3915     \__enumext_start_store_level_vii:
3916     \__enumext_start_list:nn { }
3917     {
3918         \__enumext_list_arg_two_vii:
3919         \__enumext_before_keys_exec_vii:
3920     }
3921     \__enumext_starred_columns_set_vii:
3922     \item[] \scan_stop:
3923     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3924     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3925 }
3926 {
3927     \__enumext_stop_item_tmp_vii:
3928     \__enumext_remove_extra_parsep_vii:
3929     \__enumext_stop_list:

```

```

3930     \__enumext_stop_store_level_vii:
3931     \__enumext_after_list_vii:
3932 }

```

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

`__enumext_safe_exec_vii:` We will first call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_env*`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are not nested within `enumext`, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the “storage system” to be used.

```

3933 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3934 {
3935     \__enumext_internal_mini_page:
3936     \__enumext_is_not_nested:
3937     \int_incr:N \l__enumext_level_h_int
3938     \int_compare:nNt { \l__enumext_level_h_int } > { 1 }
3939     {
3940         \msg_error:nn { enumext } { nested }
3941     }
3942     \bool_set_true:N \l__enumext_starred_bool
3943     \__enumext_is_on_first_level:
3944 }

```

(End of definition for `__enumext_safe_exec_vii:.`)

`__enumext_parse_keys_vii:n` First we will clear the variable `\l__enumext_series_str` used by the key `series`, process the environment `[(key = val)]` and execute the function `__enumext_parse_series:n` and used by the key `series`, then we execute the function `__enumext_store_active_keys_vii:n` and reprocess the `(keys)` to pass them to the storage `(sequence)` if the key `save-key` is not active and finally we call the function `__enumext_nested_base_line_fix:` used by the key `base-fix`.

```

3945 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3946 {
3947     \tl_if_novalue:nF {#1}
3948     {
3949         \str_clear:N \l__enumext_series_str
3950         \keys_set:nn { enumext / enumext* } {#1}
3951         \__enumext_parse_series:n {#1}
3952         \__enumext_store_active_keys_vii:n {#1}
3953         \__enumext_nested_base_line_fix:
3954     }
3955 }

```

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

`__enumext_before_list_vii:` The function `__enumext_before_list_vii:` first calls the function `__enumext_vspace_above_vii:` used by the keys `above` and `above*`, then calls the function `__enumext_check_ans_active:` for the check answer mechanism and finally calls the function `__enumext_start_mini_vii:` used by the keys `mini-env`, `mini-right` and `mini-right*`.

```

3956 \cs_new_protected:Nn \__enumext_before_list_vii:
3957 {
3958     \__enumext_vspace_above_vii:
3959     \__enumext_check_ans_active:
3960     \__enumext_before_args_exec_vii:
3961     \__enumext_start_mini_vii:
3962 }

```

(End of definition for `__enumext_before_list_vii:.`)

`__enumext_after_list_vii:` The function `__enumext_after_list_vii:` first calls the function `__enumext_stop_mini_vii:` used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions `__enumext_after_stop_list_vii:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below_vii:` used by the keys `below` and `below*`. Finally set `\l__enumext_starred_bool` to false and call the `__enumext_resume_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

3963 \cs_new_protected:Nn \__enumext_after_list_vii:
3964 {
3965     \__enumext_stop_mini_vii:
3966     \__enumext_after_stop_list_vii:

```

```

3967     \__enumext_check_ans_key_hook:
3968     \__enumext_vspace_below_vii:
3969     \bool_set_false:N \__enumext_starred_bool
3970     \__enumext_resume_save_counter:
3971 }

```

(End of definition for __enumext_after_list_vii:.)

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

```

3972 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3973 {
3974     \bool_if:NT \__enumext_store_active_bool
3975     {
3976         \int_compare:nNnT { \__enumext_level_int } > { 0 }
3977         {
3978             \__enumext_store_level_open_vii:
3979         }
3980     }
3981 }
3982 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3983 {
3984     \bool_if:NT \__enumext_store_active_bool
3985     {
3986         \int_compare:nNnT { \__enumext_level_int } > { 0 }
3987         {
3988             \__enumext_store_level_close_vii:
3989         }
3990     }
3991 }

```

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

11.42.1 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 __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.

```

3992 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3993 {
3994     \__enumext_stop_item_tmp_vii:
3995     \int_incr:N \__enumext_item_column_pos_vii_int
3996     \int_gincr:N \g__enumext_item_count_all_vii_int
3997     \__enumext_item_peek_args_vii:
3998 }

```

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

```

3999 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4000 {
4001     \peek_meaning:NTF (
4002     { \__enumext_joined_item_vii:w }
4003     { \__enumext_joined_item_vii:w (1) }
4004 }

```

(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.

```

4005 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4006 {
4007     \__enumext_starred_joined_item_vii:n {#1}

```

```

4008     \peek_meaning_remove:NTF *
4009     { \__enumext_starred_item_vii:w }
4010     { \__enumext_standar_item_vii:w }
4011 }

```

(End of definition for __enumext_joined_item_vii:w.)

__enumext_standar_item_vii:w

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

```

4012 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4013 {
4014     \bool_set_false:N \l__enumext_item_starred_vii_bool
4015     \peek_meaning:NTF [
4016     {
4017         \bool_set_eq:NN
4018         \l__enumext_wrap_label_vii_bool
4019         \l__enumext_wrap_label_opt_vii_bool
4020         \__enumext_start_item_vii:w
4021     }
4022     {
4023         \bool_set_true:N \l__enumext_wrap_label_vii_bool
4024         \legacy_if_set_true:n { @noitemarg }
4025         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4026     }
4027 }

```

(End of definition for __enumext_standar_item_vii:w.)

__enumext_starred_item_vii:w

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

__enumext_starred_item_vii_aux_i:w
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

```

4028 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4029 {
4030     \bool_set_true:N \l__enumext_item_starred_vii_bool
4031     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4032     \peek_meaning:NTF [
4033     { \__enumext_starred_item_vii_aux_i:w }
4034     { \__enumext_starred_item_vii_aux_ii:w }
4035 }
4036 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4037 {
4038     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4039     \__enumext_starred_item_vii_aux_ii:w
4040 }
4041 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4042 {
4043     \peek_meaning:NTF [
4044     { \__enumext_starred_item_vii_aux_iii:w }
4045     {
4046         \dim_set_eq:NN
4047         \l__enumext_item_symbol_sep_vii_dim
4048         \l__enumext_labelsep_vii_dim
4049         \legacy_if_set_true:n { @noitemarg }
4050         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4051     }
4052 }
4053 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4054 {
4055     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4056     \legacy_if_set_true:n { @noitemarg }
4057     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4058 }

```

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

11.42.2 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.

```

4059 \\cs_new_protected_nopar:Npn \\enumext_start_item_vii:w [#1]
4060 {
4061   \\cs_set_eq:NN \\enumext_stop_item_tmp_vii: \\enumext_stop_item_vii:
4062   \\legacy_if:nT { @noitemarg }
4063   {
4064     \\legacy_if_set_false:n { @noitemarg }
4065     \\legacy_if:nT { @nmbrrlist }
4066     {
4067       \\bool_if:NT \\__enumext_hyperref_bool
4068       {
4069         \\legacy_if_set_true:n { @hyper@item }
4070       }
4071       \\refstepcounter{enumXvii}
4072       \\bool_if:NT \\__enumext_check_answers_bool
4073       {
4074         \\int_gincr:N \\g__enumext_item_number_int
4075         \\bool_set_true:N \\__enumext_item_number_bool
4076       }
4077     }
4078   }

```

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 __enumext_footnotes_key_bool is false, we will redefine the command \\footnote, followed by printing the <symbol> defined for \\item* if it is present and open a new group inside which we execute font key next to \\item and the keys wrap-label, wrap-label*, align, close the group and execute the key labelsep and then the key first. Finally we open the minipage environment and execute the listparindent key which will be equal to \\parindent, the parsep key which will be equal to \\parskip and the itemindent key.

```

4079 \\group_begin:
4080   \\lrbox{ \\__enumext_item_text_vii_box }
4081   \\bool_if:NF \\__enumext_footnotes_key_bool
4082   {
4083     \\__enumext_renew_footnote:
4084   }
4085   \\bool_if:NT \\__enumext_item_starred_vii_bool
4086   {
4087     \\tl_if_blank:VT \\g__enumext_item_symbol_aux_vii_tl
4088     {
4089       \\tl_gset_eq:NN
4090       \\g__enumext_item_symbol_aux_vii_tl \\__enumext_item_symbol_vii_tl
4091     }
4092     \\mode_leave_vertical:
4093     \\skip_horizontal:n { -\\__enumext_item_symbol_sep_vii_dim }
4094     \\makebox[ 0pt ]{ r }{ \\g__enumext_item_symbol_aux_vii_tl }
4095     \\skip_horizontal:N \\__enumext_item_symbol_sep_vii_dim
4096     \\tl_gclear:N \\g__enumext_item_symbol_aux_vii_tl
4097   }
4098   \\group_begin:
4099     \\tl_use:N \\__enumext_label_font_style_vii_tl
4100     \\bool_if:NTF \\__enumext_wrap_label_vii_bool
4101     {
4102       \\makebox[ \\__enumext_labelwidth_vii_dim ]{ \\__enumext_align_label_vii_str }
4103       { \\__enumext_wrapper_label_vii:n {#1} }
4104     }
4105     {
4106       \\makebox[ \\__enumext_labelwidth_vii_dim ]{ \\__enumext_align_label_vii_str }{ #1 }
4107     }
4108   \\group_end:
4109   \\skip_horizontal:N \\__enumext_labelsep_vii_dim
4110   \\tl_use:N \\__enumext_after_list_args_vii_tl
4111   \\__enumext_minipage:w [ t ]{ \\__enumext_joined_width_vii_dim }
4112   \\skip_set_eq:NN \\parindent \\__enumext_listparindent_vii_dim

```

```

4113         \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4114         \tl_use:N \l__enumext_fake_item_indent_vii_tl
4115     }

```

(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.

```

4116 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4117 {
4118     \__enumext_endminipage:
4119     \endlrbox
4120     \group_end:
4121     \box_set_wd:Nn \l__enumext_item_text_vii_box
4122     {
4123         \l__enumext_joined_width_vii_dim
4124         + \l__enumext_labelwidth_vii_dim
4125         + \l__enumext_labelsep_vii_dim
4126     }
4127     \int_set:Nn \hbadness { 10000 }
4128     \box_use_drop:N \l__enumext_item_text_vii_box
4129     \bool_if:NF \l__enumext_footnotes_key_bool
4130     {
4131         \__enumext_print_footnote:
4132     }
4133     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4134     {
4135         \par\noindent
4136         \int_zero:N \l__enumext_item_column_pos_vii_int
4137     }
4138     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4139 }

```

(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.

```

4140 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4141 {
4142     \int_compare:nNnT
4143     {
4144         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4145     }
4146     =
4147     { 0 }
4148     {
4149         \par
4150         \vspace{ -\l__enumext_itemsep_vii_skip }
4151         \int_gzero:N \g__enumext_item_count_all_vii_int
4152     }
4153 }

```

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`.

```

4154 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`.)

11.43 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.

```

4155 \NewDocumentEnvironment{keyans*}{ o }
4156 {
4157     \__enumext_safe_exec_viii:
4158     \__enumext_parse_keys_viii:n {#1}
4159
4160     \__enumext_before_list_viii:
4161     \__enumext_start_list:nn { }

```

```

4162     {
4163         \__enumext_list_arg_two_viii:
4164         \__enumext_before_keys_exec_viii:
4165     }
4166     \__enumext_starred_columns_set_viii:
4167     \item[] \scan_stop:
4168     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4169     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4170 }
4171 {
4172     \__enumext_stop_item_tmp_viii:
4173     \__enumext_remove_extra_parsep_viii:
4174     \__enumext_check_starred_cmd:n { item }
4175     \__enumext_stop_list:
4176     \__enumext_after_list_viii:
4177 }

```

(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.

```

4178 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4179 {
4180     \int_incr:N \l__enumext_keyans_level_h_int
4181     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4182     {
4183         \msg_error:nn { enumext } { nested }
4184     }
4185     \__enumext_keyans_start_line:
4186     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4187     \bool_set_false:N \l__enumext_store_active_bool
4188     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4189     {
4190         \msg_error:nn { enumext } { keyans-wrong-level }
4191     }
4192 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

4193 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4194 {
4195     \tl_if_novalue:nF {#1}
4196     {
4197         \keys_set:nn { enumext / keyans* } {#1}
4198     }
4199 }

```

(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`.

```

4200 \cs_new_protected:Nn \__enumext_before_list_viii:
4201 {
4202     \__enumext_vspace_above_viii:
4203     \__enumext_before_args_exec_viii:
4204     \__enumext_start_mini_viii:
4205 }

```

(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.

```

4206 \cs_new_protected:Nn \__enumext_after_list_viii:
4207 {
4208     \__enumext_stop_mini_viii:
4209     \__enumext_after_stop_list_viii:
4210     \__enumext_vspace_below_viii:
4211 }

```

(End of definition for `__enumext_after_list_viii:`)

11.43.1 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`.

```
4212 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4213 {
4214     \__enumext_stop_item_tmp_viii:
4215     \int_incr:N \l__enumext_item_column_pos_viii_int
4216     \int_gincr:N \g__enumext_item_count_all_viii_int
4217     \__enumext_item_peek_args_viii:
4218 }
```

(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).

```
4219 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4220 {
4221     \peek_meaning:NTF (
4222         { \__enumext_joined_item_viii:w }
4223         { \__enumext_joined_item_viii:w (1) }
4224 }
```

(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`.

```
4225 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4226 {
4227     \__enumext_starred_joined_item_viii:n {#1}
4228     \peek_meaning_remove:NTF *
4229     { \__enumext_starred_item_viii:w }
4230     { \__enumext_standar_item_viii:w }
4231 }
```

(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 [__enumext_label_viii_tl]`.

```
4232 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4233 {
4234     \bool_set_false:N \l__enumext_item_starred_viii_bool
4235     \peek_meaning:NTF [
4236     {
4237         \bool_set_eq:NN
4238         \l__enumext_wrap_label_viii_bool
4239         \l__enumext_wrap_label_opt_viii_bool
4240         \__enumext_start_item_viii:w
4241     }
4242     {
4243         \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

4244         \legacy_if_set_true:n { @noitemarg }
4245         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4246     }
4247 }

```

(End of definition for __enumext_standar_item_viii:w.)

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w

```

The function __enumext_starred_item_viii:w together with the specified auxiliary functions aux_i:w and aux_ii:w execute \item* and \item*[\langle content \rangle].

```

4248 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4249 {
4250     \bool_set_true:N \__enumext_item_starred_viii_bool
4251     \bool_set_true:N \__enumext_wrap_label_viii_bool
4252     \peek_meaning:NTF [
4253         { \__enumext_starred_item_viii_aux_i:w }
4254         { \__enumext_starred_item_viii_aux_ii:w }
4255     }

```

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.

```

4256 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4257 {
4258     \tl_clear:N \l__enumext_store_current_label_tl
4259     \tl_if_no_value:nF { #1 }
4260     {
4261         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4262         {
4263             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4264             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4265         }
4266         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4267     }
4268     \__enumext_starred_item_viii_aux_ii:w
4269 }
4270 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4271 {
4272     \legacy_if_set_true:n { @noitemarg }
4273     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4274 }

```

(End of definition for __enumext_starred_item_viii:w, __enumext_starred_item_viii_aux_i:w, and __enumext_starred_item_viii_aux_ii:w.)

```
\__enumext_starred_item_exec:
```

The function __enumext_starred_item_exec: will be in charge of storing the current \langle label \rangle for \item* followed by the [\langle content \rangle] for \item*[\langle content \rangle] if present in the \langle sequence \rangle and \langle prop list \rangle set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```

4275 \cs_new_protected:Nn \__enumext_starred_item_exec:
4276 {
4277     \tl_put_left:Ne \l__enumext_store_current_label_tl { \__enumext_label_viii_tl }
4278     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4279     \__enumext_keyans_store_ref:
4280     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4281     \__enumext_keyans_addto_seq_link:
4282     \int_gincr:N \g__enumext_check_starred_cmd_int
4283     \bool_if:NT \l__enumext_show_answer_bool
4284     {
4285         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4286     }
4287     \bool_if:NT \l__enumext_show_position_bool
4288     {
4289         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4290         {
4291             \group_begin:
4292             \exp_not:N \normalfont
4293             \exp_not:N \footnotesize [ \int_eval:n
4294                 {
4295                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4296                 }

```

```

4297         ]
4298     \group_end:
4299 }
4300 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
4301 }
4302 }

```

(End of definition for `__enumext_starred_item_exec:`)

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

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

```

4303 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4304 {
4305     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4306     \legacy_if:nT { @noitemarg }
4307     {
4308         \legacy_if_set_false:n { @noitemarg }
4309         \legacy_if:nT { @nmbrlist }
4310         {
4311             \bool_if:NT \l__enumext_hyperref_bool
4312             {
4313                 \legacy_if_set_true:n { @hyper@item }
4314             }
4315             \refstepcounter{enumXviii}
4316         }
4317     }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment.

```

4318     \group_begin:
4319     \lrbox{ \l__enumext_item_text_viii_box }
4320     \bool_if:NF \l__enumext_footnotes_key_bool
4321     {
4322         \__enumext_renew_footnote:
4323     }
4324     \bool_if:NT \l__enumext_item_starred_viii_bool
4325     {
4326         \__enumext_starred_item_exec:
4327     }
4328     \group_begin:
4329     \tl_use:N \l__enumext_label_font_style_viii_tl
4330     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4331     {
4332         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4333         { \__enumext_wrapper_label_viii:n {#1} }
4334     }
4335     {
4336         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4337     }
4338     \group_end:
4339     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4340     \tl_use:N \l__enumext_after_list_args_viii_tl
4341     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4342     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4343     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4344     \bool_if:NT \l__enumext_item_starred_viii_bool
4345     {
4346         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4347         \__enumext_keyans_show_item_opt:
4348         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim
4349     }
4350     {
4351         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4352     }
4353 }

```

(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.

```

4354 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:

```

```

4355 {
4356   \__enumext_endminipage:
4357   \endlrbox
4358   \group_end:
4359   \box_set_wd:Nn \l__enumext_item_text_viii_box
4360   {
4361     \l__enumext_joined_width_viii_dim
4362     + \l__enumext_labelwidth_viii_dim
4363     + \l__enumext_labelsep_viii_dim
4364   }
4365   \int_set:Nn \hbadness { 10000 }
4366   \box_use_drop:N \l__enumext_item_text_viii_box
4367   \bool_if:NF \l__enumext_footnotes_key_bool
4368   {
4369     \__enumext_print_footnote:
4370   }
4371   \int_compare:nNnTF
4372   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4373   {
4374     \par\noindent
4375     \int_zero:N \l__enumext_item_column_pos_viii_int
4376   }
4377   { \hspace{ \l__enumext_columns_sep_viii_dim } }
4378 }

```

(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.

```

4379 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4380 {
4381   \int_compare:nNnT
4382   {
4383     \int_mod:nn
4384     { \g__enumext_item_count_all_viii_int }
4385     { \l__enumext_columns_viii_int }
4386   }
4387   =
4388   { 0 }
4389   {
4390     \par
4391     \vspace{ -\l__enumext_itemsep_viii_skip }
4392     \int_gzero:N \g__enumext_item_count_all_viii_int
4393   }
4394 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

11.44 The command \getkeyans

\getkeyans

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

```

4395 \NewDocumentCommand \getkeyans { m }
4396 {
4397   \exp_args:Ne \__enumext_getkeyans_aux:n
4398   { \tl_to_str:e { \text_expand:n {#1} } }
4399 }

```

(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 $\langle \textit{argument} \rangle$ using “:”. If “:” is omitted it will return an error.

```

4400 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4401 {
4402   \str_if_in:nnTF {#1} { : }
4403   {
4404     \use:e
4405     {
4406       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4407       { {##1} {##2} }
4408     }

```

```

4409         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4410     }
4411     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4412 }

```

(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⟩*.

```

4413 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4414 {
4415     \prop_if_exist:cF { g__enumext_#1_prop }
4416     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4417     \group_begin:
4418     \prop_item:cn { g__enumext_#1_prop }{#2}
4419     \group_end:
4420 }

```

(End of definition for __enumext_getkeyans:nn.)

11.45 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 __enumext_print_keyans_X_tl.

The variable __enumext_print_keyans_starred_tl will have the default *⟨keys⟩* for \printkeyans* and will be set by \setenumext[⟨print*⟩] and the variable __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⟩]

```

4421 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4422 \keys_define:nn { enumext / print }
4423 {
4424     print* .code:n = \keys_precompile:neN { enumext / enumext* }
4425                 { \__enumext_filter_save_key:n {#1} }
4426                 \__enumext_print_keyans_starred_tl, % starred cmd
4427     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4428     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4429                 { \__enumext_filter_save_key:n {#1} }
4430                 \__enumext_print_keyans_i_tl,
4431     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4432     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4433                 { \__enumext_filter_save_key:n {#1} }
4434                 \__enumext_print_keyans_ii_tl,
4435     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4436     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4437                 { \__enumext_filter_save_key:n {#1} }
4438                 \__enumext_print_keyans_iii_tl,
4439     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4440     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4441                 { \__enumext_filter_save_key:n {#1} }
4442                 \__enumext_print_keyans_iv_tl,
4443     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4444     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4445                 { \__enumext_filter_save_key:n {#1} }
4446                 \__enumext_print_keyans_vii_tl, % starred nested
4447     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4448 }

```

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

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

```

4449 \NewDocumentCommand \printkeyans { s O{} m }
4450 {
4451     \group_begin:

```

```

4452 \tl_use:N \l__enumext_print_keyans_i_tl
4453 \tl_use:N \l__enumext_print_keyans_ii_tl
4454 \tl_use:N \l__enumext_print_keyans_iii_tl
4455 \tl_use:N \l__enumext_print_keyans_iv_tl
4456 \tl_use:N \l__enumext_print_keyans_vii_tl
4457 \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4458 \group_end:
4459 }

```

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

`__enumext_printkeyans:nnn`

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

```

4460 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4461 {
4462   \seq_if_exist:cTF { g__enumext_#3_seq }
4463   {
4464     \seq_if_empty:cF { g__enumext_#3_seq }
4465     {
4466       %%\seq_show:c { g__enumext_#3_seq }

```

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

```

4467   \bool_if:nTF {#1}
4468   {
4469     \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4470     {
4471       \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4472     }
4473     {
4474       \tl_use:N \l__enumext_print_keyans_starred_tl
4475       \begin{enumext*}[#2]
4476         \keys_set:nn { enumext / level-1 } { base-fix }
4477         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4478         \end{enumext*}
4479       }
4480     }

```

Otherwise it will open the environment `enumext` passing the optional argument to the “first level”, set the key `base-fix` and then map the `(sequence)`.

```

4481     {
4482       \begin{enumext}[#2]
4483         \keys_set:nn { enumext / enumext* } { base-fix }
4484         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4485         \end{enumext}
4486       }
4487     }
4488   }
4489   {
4490     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4491   }
4492 }

```

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

11.46 The command `\setenumext`

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

`__enumext_filter_first_level:n`
`__enumext_filter_first_level_key:n`
`__enumext_filter_first_level_pair:nn`

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

```

4493 \cs_new:Npn \__enumext_filter_first_level:n #1
4494 {
4495   \use:e
4496   {
4497     \keyval_parse:NNn
4498     \__enumext_filter_first_level_key:n
4499     \__enumext_filter_first_level_pair:nn {#1}

```

```

4500     }
4501 }

```

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

```

4502 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4503 {
4504   \str_case:nnF {#1}
4505   {
4506     { resume } {}
4507     { resume* } {}
4508   }
4509   { , { \exp_not:n {#1} } }
4510 }

```

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

```

4511 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4512 {
4513   \str_case:nnF {#1}
4514   {
4515     { series } {}
4516     { resume } {}
4517     { save-ans } {}
4518   }
4519   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
4520 }

```

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

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

```

4521 \keys_define:nn { enumext / meta-families }
4522 {
4523   enumext-1 .code:n =
4524   {
4525     \keys_set:ne { enumext / level-1 }
4526     {
4527       \__enumext_filter_first_level:n {#1}
4528     }
4529   } ,
4530   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4531   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4532   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4533   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4534   enumext* .code:n =
4535   {
4536     \keys_set:ne { enumext / enumext* }
4537     {
4538       \__enumext_filter_first_level:n {#1}
4539     }
4540   } ,
4541   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4542   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4543   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4544   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4545   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4546   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4547   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4548   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4549 }

```

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

```

4550 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4551 {
4552   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4553   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4554 }

```

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

```

4555 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4556 {
4557   \tl_if_novalue:nTF {#1}

```



```

4558     {
4559       \seq_map_inline:Nn \c__enumext_all_families_seq
4560     }
4561     {
4562       \seq_clear:N \l__enumext_setkey_tmpa_seq
4563       \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4564       \int_set:Nn \l__enumext_setkey_tmpa_int
4565       {
4566         \seq_count:N \l__enumext_setkey_tmpb_seq
4567       }
4568       \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4569       {
4570         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4571         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4572         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4573         {
4574           \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4575         }
4576       }
4577       {
4578         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4579       }
4580       \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4581       { \seq_map_inline:Nn \c__enumext_all_families_seq }
4582       { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4583     }
4584     {
4585       \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4586     }
4587   }

```

(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.

```

4588 \cs_new_protected:Npn \__enumext_set_parse:n #1
4589 {
4590   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4591   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4592   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4593   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4594   {
4595     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4596     { \tl_trim_spaces:n {#1} }
4597   }
4598   { \__enumext_set_error:nn {#1} { } }
4599 }
4600 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4601 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.47 Messages

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

```

4602 \msg_new:nnn { enumext } { package-load }
4603 {
4604   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4605 }
4606 \msg_new:nnn { enumext } { package-not-load }
4607 {
4608   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4609 }
4610 \msg_new:nnn { enumext } { package-load-foot }
4611 {
4612   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4613 }

```

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

```

4614 \msg_new:nnn { enumext } { counters }
4615 {
4616   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4617   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4618 }

```

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

```
4619 \msg_new:nnn { enumext } { unknown-choice }
4620 {
4621   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4622 }
```

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

```
4623 \msg_new:nnnn { enumext } { anskey-env-error }
4624 {
4625   The ~ '#1' ~ environment ~is~ reserved ~ by ~\
4626   'enumext' ~ package, ~ It~ is~ already~ defined.
4627 }
4628 {
4629   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4630   for ~ the ~ 'save-ans' ~ key.\
4631 }
```

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

```
4632 \msg_new:nnn { enumext } { store-prop }
4633 {
4634   * ~ Package ~ enumext: ~ Creating ~
4635   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4636 }
4637 \msg_new:nnn { enumext } { store-seq }
4638 {
4639   * ~ Package ~ enumext: ~ Creating ~
4640   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4641 }
4642 \msg_new:nnn { enumext } { store-int }
4643 {
4644   * ~ Package ~ enumext: ~ Creating ~
4645   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4646 }
4647 \msg_new:nnn { enumext } { prop-seq-int-hook }
4648 {
4649   * ~ Package ~ enumext: ~ Elements ~ in ~
4650   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\
4651   * ~ Package ~ enumext: ~ Elements ~ in ~
4652   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\
4653   * ~ Package ~ enumext: ~ Value ~ off ~
4654   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4655 }
4656 \msg_new:nnn { enumext } { item-answer-hook }
4657 {
4658   * ~ Package ~ enumext: ~ Value ~ off ~
4659   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
4660   * ~ Package ~ enumext: ~ Value ~ off ~
4661   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
4662   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4663 }
```

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

```
4664 \msg_new:nnn { enumext } { invalid-key }
4665 {
4666   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4667 }
4668 \msg_new:nnn { enumext } { unknown-key-family }
4669 {
4670   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4671 }
```

Messages used in length calculation.

```
4672 \msg_new:nnn { enumext } { width-negative }
4673 {
4674   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4675   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4676 }
4677 \msg_new:nnn { enumext } { width-zero }
4678 {
4679   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4680   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4681 }
```

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

```

4682 \msg_new:nnn { enumext } { list-lengths }
4683 {
4684     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4685     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4686     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4687     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4688     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4689     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4690     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4691     \__enumext_show_length:nnn { skip } { topsep } {#1}
4692     \__enumext_show_length:nnn { skip } { parsep } {#1}
4693     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4694     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4695     *****
4696 }

```

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

```

4697 \msg_new:nnn { enumext } { list-lengths-not-nested }
4698 {
4699     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4700     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4701     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4702     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4703     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4704     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4705     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4706     \__enumext_show_length:nnn { skip } { topsep } {#1}
4707     \__enumext_show_length:nnn { skip } { parsep } {#1}
4708     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4709     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4710     *****
4711 }

```

Messages used by `ref` key.

```

4712 \msg_new:nnn { enumext } { key-ref-empty }
4713 {
4714     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4715 }

```

Messages used by `save-ans` key.

```

4716 \msg_new:nnn { enumext } { save-ans-empty }
4717 {
4718     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4719 }
4720 \msg_new:nnn { enumext } { save-ans-log }
4721 {
4722     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4723 }
4724 \msg_new:nnn { enumext } { save-ans-log-hook }
4725 {
4726     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4727 }
4728 \msg_new:nnn { enumext } { save-ans-hook }
4729 {
4730     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4731 }

```

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

```

4732 \msg_new:nnn { enumext } { need-save-ans }
4733 {
4734     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4735 }
4736 \msg_new:nnn { enumext } { items-same-answer }
4737 {
4738     *****\\
4739     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4740     for ~ \c_left_brace_str #2 \c_right_brace_str\\
4741     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4742     'OK', ~ all ~ items ~ with ~ answer.\\
4743     *****
4744 }
4745 \msg_new:nnn { enumext } { item-greater-answer }

```

```

4746 {
4747     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4748     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4749     Items ~ > ~ Answers.
4750 }
4751 \msg_new:nnn { enumext } { item-less-answer }
4752 {
4753     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4754     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4755     Items ~ < ~ Answers.
4756 }

```

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

```

4757 \msg_new:nnn { enumext } { missing-starred }
4758 {
4759     Missing ~ '\c_backslash_str #1*' ~ #2.
4760 }
4761 \msg_new:nnn { enumext } { many-starred }
4762 {
4763     Many ~ '\c_backslash_str #1*' ~ #2.
4764 }

```

Messages used by `\printkeyans*` command.

```

4765 \msg_new:nnn { enumext } { print-starred }
4766 {
4767     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4768     #2 ~ environment ~ \msg_line_context:.
4769 }

```

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

```

4770 \msg_new:nnn { enumext } { list-too-deep }
4771 {
4772     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4773     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4774 }

```

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

```

4775 \msg_new:nnn { enumext } { anskey-unnumber-item }
4776 {
4777     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
4778 }
4779 \msg_new:nnn { enumext } { anskey-already-stored }
4780 {
4781     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
4782 }
4783 \msg_new:nnn { enumext } { anskey-empty-arg }
4784 {
4785     Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4786 }
4787 \msg_new:nnn { enumext } { anskey-wrong-place }
4788 {
4789     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4790     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4791 }
4792 \msg_new:nnn { enumext } { anskey-nested }
4793 {
4794     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4795 }
4796 \msg_new:nnn { enumext } { anskey-math-mode }
4797 {
4798     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
4799 }
4800 \msg_new:nnn { enumext } { anskey-env-wrong }
4801 {
4802     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4803 }
4804 \msg_new:nnn { enumext } { anspic-wrong-place }
4805 {
4806     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4807     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4808 }
4809 \msg_new:nnn { enumext } { command-wrong-place }
4810 {

```

```

4811     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4812     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4813 }
4814 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
4815 {
4816     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
4817     'anskey*' ~ and ~ is ~ being ~ ignored.
4818 }
4819 {
4820     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4821     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4822 }
4823 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
4824 {
4825     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
4826     'anskey*' ~ and ~ is ~ being ~ ignored.
4827 }
4828 {
4829     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4830     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4831 }
4832 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
4833 { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
4834 {
4835     The~command~'\c_backslash_str anskey'~does~not~have~a~key~called~'#1'.\\
4836     Check~that~you~have~spelled~the~key~name~correctly.
4837 }
4838 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
4839 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
4840 {
4841     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4842     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4843 }

```

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

```

4844 \msg_new:nnn { enumext } { keyans-nested }
4845 {
4846     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4847 }
4848 \msg_new:nnn { enumext } { keyans-wrong-level }
4849 {
4850     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4851     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4852 }
4853 \msg_new:nnn { enumext } { wrong-place }
4854 {
4855     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
4856     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
4857 }
4858 \msg_new:nnn { enumext } { keyanspic-nested }
4859 {
4860     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
4861 }
4862 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4863 {
4864     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4865     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4866 }

```

Messages used by [\getkeyans](#) command.

```

4867 \msg_new:nnn { enumext } { undefined-storage-anskey }
4868 {
4869     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4870 }

```

Messages used by [\miniright](#) command.

```

4871 \msg_new:nnn { enumext } { missing-miniright }
4872 {
4873     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~\\
4874     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4875 }
4876 \msg_new:nnn { enumext } { wrong-miniright-place }

```

```

4877 {
4878     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4879     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4880 }
4881 \msg_new:nnn { enumext } { wrong-miniright-use }
4882 {
4883     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4884     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4885 }

```

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

```

4886 \msg_new:nnn { enumext } { nested }
4887 {
4888     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4889 }
4890 \msg_new:nnn { enumext } { item-joined }
4891 {
4892     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4893 }
4894 \msg_new:nnn { enumext } { item-joined-columns }
4895 {
4896     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4897 }

```

11.48 Finish package

Finish package implementation.

```

4898 \file_input_stop:
4899 </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	
<code>*</code>	212
<code>\+</code>	204
<code>\-</code>	204
<code>\\</code> 220, 2632, 3545, 4616, 4625, 4630, 4650, 4652, 4659, 4661, 4674, 4679, 4684, 4699, 4738, 4740, 4742, 4747, 4748, 4753, 4754, 4772, 4789, 4806, 4811, 4820, 4829, 4835, 4841, 4850, 4855, 4864, 4873, 4878, 4883	
A	
above	<u>1445</u>
above*	<u>1445</u>
<code>\addvspace</code> 1099, 1127, 1243, 1322, 1385, 1391, 1419, 1436, 3384, 3399, 3501, 3516, 3834, 3848, 3889, 3903	
after	<u>938</u>
align	<u>493</u>
<code>\Alph</code>	36, <u>41</u>
<code>\Alpha</code>	445, 560, 605, 673, 4443
<code>\alph</code>	36, <u>41</u>
<code>\alpha</code>	446, 558, 4435
<code>\anskey</code>	12, 74, 75, <u>2450</u>
anskey*	13, <u>2560</u>
<code>\anspic</code>	15, 99, <u>3523</u>
<code>\anspic*</code>	67
<code>\arabic</code>	30, 36
<code>\arabic</code>	444, 557, 604, 4427, 4431, 4447
B	
base-fix	<u>797</u>
<code>\baselineskip</code>	49
<code>\baselineskip</code>	814, 825
before	<u>938</u>
before*	<u>938</u>
below	<u>1445</u>
below*	<u>1445</u>
bool commands:	
<code>\bool_gset_false:N</code> 320, 321, 322, 2736, 2738, 3850, 3854, 3905	
<code>\bool_gset_true:N</code> 232, 241, 1041, 1943, 1949, 3826, 3851, 3881, 3906	
<code>\bool_if:NTF</code> . 385, 397, 414, 1467, 1481, 1494, 1505, 1516, 1527, 1538, 1549, 1603, 1620, 1625, 1633, 1660, 1698, 1703, 1710, 1714, 1736, 1741, 1749, 1756, 1787, 1795, 1888, 2083, 2093, 2172, 2196, 2203, 2227, 2327, 2349, 2389, 2400, 2404, 2454, 2473, 2497, 2551, 2562, 2651, 2688, 2752, 2785, 2800, 2875, 2886, 2890, 3003, 3034, 3109, 3125, 3187, 3197, 3229, 3234, 3318, 3368, 3382, 3390, 3429, 3486, 3499, 3507, 3525, 3822, 3831, 3835, 3877, 3886, 3890, 3974, 3984, 4067, 4072, 4081, 4085, 4100, 4129, 4283, 4287, 4311, 4320, 4324, 4330, 4344, 4367	
<code>\bool_if:nTF</code> 1420, 1437, 3046, 3081, 3145, 3546, 4467	
<code>\bool_if_p:N</code> 250, 265, 810, 811, 821, 822, 1767, 1768, 1776, 1777, 1901, 1927, 1940, 1941, 1946, 1947, 2260, 2269, 2270, 2282, 2298, 2334, 2375, 2376, 2674, 2862, 2863, 2900, 2901, 3291, 3293, 3304, 3553, 3554	
<code>\bool_lazy_all:nTF</code> 248, 263, 1899, 1925, 2258, 2267, 2280, 2296, 3289, 3302	
<code>\bool_lazy_and:nnTF</code> 228, 237, 809, 820, 1766, 1775, 1939, 1945, 2333, 2340, 2374, 2515, 2527, 2673, 2679, 2861	
<code>\bool_lazy_or:nnTF</code> 1828, 1835, 2899, 3552	
<code>\bool_new:N</code> 34, 35, 36, 37, 38, 39, 40, 41, 63, 72, 93, 98, 99, 104, 105, 108, 129, 130, 138, 139, 144, 146, 147, 161, 173, 175	
<code>\bool_not_p:n</code> 229, 238, 2285, 2300, 2335, 2341, 2675, 2680, 3292, 3305	
<code>\bool_set_eq:NN</code> 3012, 3061, 4017, 4237	
<code>\bool_set_false:N</code> 394, 831, 1873, 1874, 1906, 1911, 1915, 1919, 1932, 2615, 3405, 3437, 3519, 3584, 3602, 3969, 4014, 4187, 4234	
<code>\bool_set_true:N</code> . 255, 256, 270, 271, 376, 380, 486, 846, 1451, 1456, 1723, 1845, 1846, 2115, 2123, 2616, 3006, 3008, 3037, 3039, 3057, 3069, 3209, 3266, 3298, 3311, 3337, 3434, 3461, 3811, 3866, 3942, 4023, 4030, 4031, 4075, 4243, 4250, 4251	
box commands:	
<code>\box_dp:N</code> . . 1139, 1143, 1147, 1158, 1162, 1173, 1182, 1188, 1198, 1211, 1217, 1223, 1254, 1255, 1256, 1259, 1269, 1273, 1282, 1289, 1294, 1302, 1331, 1332, 1335, 1342, 1355, 1363, 1369, 1377, 3614	
<code>\box_new:N</code> 69, 168, 174	
<code>\box_set_wd:Nn</code> 4121, 4359	
<code>\box_use_drop:N</code> 3846, 3901, 4128, 4366	
<code>\box_wd:N</code> 452	
C	
<code>\c</code>	212, 213, 710, 712, 724, 726
<code>\catcode</code>	2632
<code>\cB</code>	213
<code>\cE</code>	213
<code>\centering</code>	1422, 1439, 3635, 3839, 3894
check-ans	<u>1865</u>
Document class:	
article	42
clist commands:	
<code>\clist_const:Nn</code>	180
<code>\clist_map_function:nN</code>	3622
<code>\clist_map_inline:Nn</code> 492, 752, 937, 952, 1033, 1461	
<code>\clist_map_inline:nn</code> . 48, 59, 77, 83, 95, 107, 132, 155, 179, 520, 540, 806, 851, 872, 1047, 1567, 1812, 1879, 2062, 2080, 2112, 2255, 2794, 2961, 3174, 3177, 3204, 3216, 3219, 3239, 4591	
<code>\columnbreak</code>	74
<code>\columnbreak</code>	2337
columns	<u>1017</u>
columns-sep	<u>1017</u>
<code>\columnsep</code>	95, 98
<code>\columnsep</code>	3362, 3483
<code>\columnseprule</code>	95, 98
<code>\columnseprule</code>	3366, 3485
Commands provide by enumext :	
<code>\anskey</code> 28, 64, 69–73, 75, 76, 83, 84, 93, 108, 116, 117, 123	
<code>\anspic*</code> 28, 67, 70, 82–84, 99–101, 116, 117	
<code>\anspic</code>	71, 99–101, 123
<code>\getkeyans</code>	70, 116, 124
<code>\item*</code> . 28, 67, 70, 71, 82–84, 87, 88, 109, 114, 116, 117	
<code>\item</code>	87, 88, 103, 108–110, 113

<code>\miniright</code>	27, 47, 54, 55, 94, 96–98, 124
<code>\printkeyans*</code>	117
<code>\printkeyans</code>	28, 71, 117
<code>\setenumext</code>	28, 117, 119–121
Counters defined by <code>enumext</code> :	
<code>enumXiii</code>	26, 35
<code>enumXii</code>	26, 35
<code>enumXiv</code>	26, 35
<code>enumXi</code>	26, 35
<code>enumXviii</code>	26, 35
<code>enumXvii</code>	26, 35, 110
<code>enumXvi</code>	26, 35
<code>enumXv</code>	26, 35
cs commands:	
<code>\cs_generate_variant:Nn</code>	454, 470, 716, 732, 2164, 2169, 2245, 2568, 3164, 3624, 4421
<code>\cs_if_exist:NTF</code>	424
<code>\cs_if_free:NTF</code>	2519, 2531
<code>\cs_new:Nn</code>	198
<code>\cs_new:Npn</code>	216, 1568, 1577, 1587, 2127, 2136, 2144, 4493, 4502, 4511
<code>\cs_new_eq:NN</code>	347, 348, 349, 353, 354, 399, 400, 403, 404
<code>\cs_new_protected:Nn</code>	208, 222, 246, 279, 306, 312, 318, 324, 330, 338, 356, 371, 581, 644, 696, 807, 953, 957, 961, 965, 969, 973, 977, 981, 985, 989, 993, 997, 1001, 1005, 1009, 1013, 1048, 1060, 1084, 1101, 1112, 1129, 1204, 1228, 1245, 1307, 1324, 1346, 1381, 1387, 1462, 1476, 1490, 1501, 1512, 1523, 1534, 1545, 1631, 1734, 1747, 1764, 1785, 1813, 1818, 1843, 1884, 1894, 1937, 1952, 1959, 1968, 1973, 1978, 1983, 1992, 1997, 2002, 2170, 2194, 2201, 2225, 2232, 2246, 2471, 2490, 2506, 2569, 2605, 2636, 2671, 2713, 2734, 2742, 2783, 2798, 2826, 2859, 2895, 2907, 2915, 2966, 2970, 2989, 3042, 3077, 3093, 3103, 3119, 3259, 3287, 3316, 3323, 3346, 3376, 3388, 3427, 3451, 3469, 3494, 3505, 3542, 3586, 3600, 3620, 3625, 3641, 3672, 3801, 3820, 3856, 3875, 3933, 3956, 3963, 3972, 3982, 3999, 4140, 4178, 4200, 4206, 4219, 4275, 4379
<code>\cs_new_protected:Npn</code>	186, 190, 194, 407, 422, 439, 449, 455, 561, 606, 678, 703, 717, 1409, 1428, 1599, 1618, 1688, 1721, 1823, 2007, 2081, 2091, 2113, 2121, 2156, 2165, 2323, 2386, 2398, 2436, 2440, 2560, 2591, 2595, 2626, 2762, 2836, 2880, 2999, 3018, 3053, 3065, 3133, 3167, 3207, 3269, 3447, 3595, 3703, 3752, 3945, 4005, 4012, 4028, 4036, 4041, 4053, 4193, 4225, 4232, 4248, 4256, 4270, 4400, 4413, 4460, 4588, 4600
<code>\cs_new_protected_nopar:Nn</code>	3992, 4116, 4212, 4354
<code>\cs_new_protected_nopar:Npn</code>	4059, 4303
<code>\cs_set:Nn</code>	2391
<code>\cs_set:Npn</code>	2256, 2294, 4406
<code>\cs_set_eq:NN</code>	3923, 3924, 4061, 4168, 4169, 4305
<code>\cs_set_protected:Nn</code>	877, 893, 905, 917
<code>\cs_set_protected:Npn</code>	44, 53, 70, 78, 90, 96, 125, 151, 159, 471, 493, 525, 541, 588, 733, 753, 797, 833, 856, 929, 938, 1017, 1034, 1445, 1556, 1804, 1865, 2024, 2063, 2099, 2248, 2787, 2950, 3165, 3205
<code>\cs_to_str:N</code>	441, 464
<code>\cs_undefine:N</code>	2508, 2509, 2510, 2511
<code>\dim_add:Nn</code>	3605, 3666, 3697
<code>\dim_compare:nNnTF</code>	879, 895, 907, 919, 1411, 1430, 3135, 3140, 3146, 3152, 3154, 3156, 3328, 3351, 3455, 3473, 3597, 3643, 3659, 3674, 3690, 3803, 3858
<code>\dim_compare:nTF</code>	2359, 2701
<code>\dim_gset_eq:NN</code>	3812, 3867
<code>\dim_gzero:N</code>	2740, 3853, 3908
<code>\dim_new:N</code>	66, 73, 74, 75, 92, 134, 167, 169, 170, 176
<code>\dim_set:Nn</code>	452, 847, 3032, 3138, 3143, 3145, 3148, 3149, 3153, 3155, 3158, 3159, 3161, 3331, 3354, 3457, 3475, 3627, 3645, 3652, 3676, 3683, 3738, 3787, 3805, 3860, 4055
<code>\dim_set_eq:NN</code>	548, 595, 666, 670, 3027, 3176, 3218, 3362, 3483, 3745, 3748, 3749, 3794, 3797, 3798, 4046
<code>\dim_sub:Nn</code>	3661, 3692
<code>\dim_use:N</code>	880, 888, 1412, 1418, 2235, 2238, 2243, 3098, 3100, 3329, 3334, 3335, 3342, 3352, 3356, 3357, 3359
<code>\dim_zero:N</code>	3210, 3366, 3485, 3606, 3607, 3608
<code>\dim_zero_new:N</code>	185
<code>\c_zero_dim</code>	882, 896, 908, 920, 1412, 1430, 2361, 2703, 3135, 3140, 3146, 3153, 3329, 3352, 3455, 3473, 3643, 3659, 3674, 3690, 3803, 3858
E	
<code>\end</code>	1415, 1433, 2198, 2229, 3381, 3398, 3498, 3515, 3824, 3847, 3879, 3902, 4469, 4478, 4485
<code>\endgroup</code>	2632
<code>\endlist</code>	348
<code>\endlrbox</code>	4119, 4357
<code>\endminipage</code>	354
<code>enumext</code>	5, <u>3240</u>
enumext internal commands:	
<code>\l__enumext__check_start_line_env_tl</code>	32
<code>\l__enumext__ref_the_count_tl</code>	38
<code>\l__enumext__resume_name_tl</code>	60
<code>__enumext_add_pre_parsep:</code>	48, 1058, <u>1060</u> , 1060
<code>__enumext_after_args_exec:</code>	46, <u>953</u> , 965, 3252
<code>__enumext_after_args_exec_v:</code>	46, 47, <u>969</u> , 981, 3420
<code>__enumext_after_args_exec_vii:</code>	<u>985</u> , 1009
<code>__enumext_after_args_exec_viii:</code>	1013
<code>__enumext_after_env:nn</code>	79, 80, 82, 96, 105, 111, <u>190</u> , 190, 2646, 3408, 3829, 3884, 4154
<code>__enumext_after_hyperref:</code>	34, 369, <u>371</u> , 371
<code>__enumext_after_list:</code>	95, 112, 3257, <u>3388</u> , 3388
<code>\l__enumext_after_list_args_v_tl</code>	983
<code>\l__enumext_after_list_args_vii_tl</code>	1011, 4110
<code>\l__enumext_after_list_args_viii_tl</code>	1015, 4340
<code>__enumext_after_list_v:</code>	98, 3425, <u>3505</u> , 3505
<code>__enumext_after_list_vii:</code>	107, 3931, <u>3963</u> , 3963
<code>__enumext_after_list_viii:</code>	4176, <u>4206</u> , 4206
<code>__enumext_after_stop_list:</code>	46, 47, <u>953</u> , 961, 3403
<code>__enumext_after_stop_list_v:</code>	46, <u>969</u> , 977, 3520
<code>\l__enumext_after_stop_list_v_tl</code>	979
<code>__enumext_after_stop_list_vii:</code>	107, <u>985</u> , 1001, 3966
<code>\l__enumext_after_stop_list_vii_tl</code>	1003
<code>__enumext_after_stop_list_viii:</code>	1005, 4209
<code>\l__enumext_after_stop_list_viii_tl</code>	1007
<code>\l__enumext_align_label_vii_str</code>	4102, 4106
<code>\l__enumext_align_label_viii_str</code>	4332, 4336
<code>\l__enumext_align_label_X_str</code>	<u>159</u>

D

<code>\d</code>	204
<code>\DeclareDocumentEnvironment</code>	360
dim commands:	
<code>\dim_abs:n</code>	3138, 3143

```

\c__enumext_all_envs_clist .. 180, 492, 752, 937,
    952, 1033, 1461
\c__enumext_all_families_seq .. 119, 4550, 4559,
    4581
\l__enumext_anskey_env_bool 31, 78, 34, 256, 271,
    2562
\__enumext_anskey_env_clean_vars: . 81, 2667,
    2671, 2734
\__enumext_anskey_env_define_keys: 78, 2560,
    2569, 2640
\__enumext_anskey_env_exec: 80, 2565, 2636, 2636
\__enumext_anskey_env_make:n 64, 78, 1848, 2560,
    2560, 2568
\__enumext_anskey_env_reset_keys: 79, 80, 2605,
    2668
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2560
\__enumext_anskey_env_save_keys: .. 80, 2648,
    2671, 2671
\__enumext_anskey_env_store: .. 81, 2664, 2671,
    2713
\__enumext_anskey_env_unknown:n 79, 2588, 2591
\__enumext_anskey_env_unknown:nn . 2593, 2595
\l__enumext_anskey_level_int .. 28, 2492, 2493
\__enumext_anskey_safe_inner: . 77, 2465, 2471,
    2490
\__enumext_anskey_safe_inner:n ..... 76
\__enumext_anskey_safe_outer: . 76, 2452, 2471,
    2471
\__enumext_anskey_show_wrap_arg:n . 75, 2386,
    2386, 2402, 2417
\__enumext_anskey_show_wrap_left:n 75, 2331,
    2398, 2398
\__enumext_anskey_unknown:n 76, 2420, 2434, 2436
\__enumext_anskey_unknown:nn . 2420, 2438, 2440
\__enumext_anskey_wrapper:n ..... 2028, 2396
\__enumext_at_begin_document:n .. 33, 186, 186,
    345, 351
\l__enumext_base_line_fix_bool . 801, 811, 822,
    831
\__enumext_before_args_exec: 45, 953, 953, 3326
\__enumext_before_args_exec_v: .. 46, 969, 969,
    3454
\__enumext_before_args_exec_vii: .. 985, 985,
    3960
\__enumext_before_args_exec_viii: 989, 4203
\__enumext_before_env:nn 78, 190, 194, 2513, 2525,
    2537, 2638
\__enumext_before_keys_exec: 45, 953, 957, 3250
\__enumext_before_keys_exec_v: .. 46, 969, 973,
    3418
\__enumext_before_keys_exec_vii ..... 985
\__enumext_before_keys_exec_vii: 46, 993, 3919
\__enumext_before_keys_exec_viii: .. 46, 997,
    4164
\__enumext_before_list: ... 94, 3244, 3323, 3323
\__enumext_before_list_v: . 97, 3413, 3451, 3451
\__enumext_before_list_vii: ... 107, 3914, 3956,
    3956
\__enumext_before_list_viii: .. 112, 4160, 4200,
    4200
\l__enumext_before_no_starred_key_v_tl 975
\l__enumext_before_no_starred_key_vii_-
    tl ..... 995
\l__enumext_before_no_starred_key_viii_-
    tl ..... 999
\l__enumext_before_starred_key_v_tl ... 971
\l__enumext_before_starred_key_vii_tl . 987
\l__enumext_before_starred_key_viii_tl 991
\__enumext_calc_hspace:NNNNNNN 90, 3133, 3133,
    3164, 3169, 3211
\__enumext_check_ans_active: . 65, 94, 107, 1884,
    1884, 3327, 3959
\g__enumext_check_ans_item_tl ..... 84
\g__enumext_check_ans_key_bool 66, 67, 138, 320,
    1943, 1949, 2752
\l__enumext_check_ans_key_bool .. 66, 87, 1869,
    1874, 1940, 1946
\__enumext_check_ans_key_hook: . 66, 107, 1937,
    1937, 3402, 3967
\__enumext_check_ans_level: 65, 1884, 1890, 1894
\__enumext_check_ans_log: 66, 67, 82, 1983, 1983,
    2756
\__enumext_check_ans_log_msg_greater: 1983,
    1989, 2002
\__enumext_check_ans_log_msg_less: 1983, 1987,
    1992
\__enumext_check_ans_log_msg_same_ok: 1983,
    1988, 1997
\__enumext_check_ans_msg_greater: 1959, 1965,
    1978
\__enumext_check_ans_msg_less: 1959, 1963, 1968
\__enumext_check_ans_msg_same_ok: 1959, 1964,
    1973
\__enumext_check_ans_show: .. 66, 82, 1959, 1959,
    2754
\g__enumext_check_ans_show_bool ..... 96
\l__enumext_check_answers_bool 64, 65, 76, 138,
    1846, 1873, 1888, 2172, 2196, 2203, 2227, 2454, 2651,
    2875, 3003, 3034, 4072
\__enumext_check_starred_cmd:n 32, 67, 84, 2007,
    2007, 3423, 3581, 4174
\g__enumext_check_starred_cmd_int 138, 2010,
    2016, 2021, 3075, 3551, 4282
\l__enumext_check_start_line_env_tl 138, 285,
    292, 299, 2013, 2019, 2022
\l__enumext_columns_sep_v_dim 3473, 3475, 3483
\l__enumext_columns_sep_vii_dim .. 3643, 3645,
    3654, 3666, 3742, 4138
\l__enumext_columns_sep_viii_dim . 3674, 3676,
    3685, 3697, 3791, 4377
\l__enumext_columns_v_int 1250, 3471, 3479, 3491,
    3496
\l__enumext_columns_vii_int .. 3648, 3651, 3655,
    3664, 3706, 3710, 3713, 3719, 3725, 3729, 4133, 4144
\l__enumext_columns_viii_int . 3679, 3682, 3686,
    3695, 3755, 3759, 3762, 3768, 3774, 3778, 4372, 4385
\l__enumext_counter_i_tl ..... 44, 431
\l__enumext_counter_ii_tl ..... 44, 432
\l__enumext_counter_iii_tl ..... 44, 433
\l__enumext_counter_iv_tl ..... 44, 434
\c__enumext_counter_style_tl ..... 30, 49, 210
\g__enumext_counter_styles_tl . 27, 36, 66, 442,
    460
\l__enumext_counter_v_tl ..... 44, 435, 686
\l__enumext_counter_vi_tl ..... 44, 436
\l__enumext_counter_vii_tl ..... 44, 437, 616
\l__enumext_counter_viii_tl ..... 44, 438, 633
\l__enumext_current_widest_dim 27, 66, 466, 549,

```

596, 667, 671
 __enumext_default_item:n ... 2999, 2999, 3050
 __enumext_define_counters:Nn 26, 422, 422, 431, 432, 433, 434, 435, 436, 437, 438
 __enumext_endminipage: . 33, 351, 354, 366, 3637, 4118, 4356
 \g__enumext_envir_name_tl 31, 34, 257, 272, 328, 1816, 1821, 1831, 1971, 1976, 1981, 1995, 2000, 2005
 __enumext_execute_after_env: 32, 33, 63, 66, 67, 77, 82, 2742, 2742, 3408, 4154
 __enumext_fake_item: 877, 877, 3196
 \l__enumext_fake_item_indent_v_dim 896, 901
 \l__enumext_fake_item_indent_v_tl 898, 3058, 3062, 3070
 \l__enumext_fake_item_indent_vii_dim 908, 913
 \l__enumext_fake_item_indent_vii_tl 910, 4114
 \l__enumext_fake_item_indent_viii_dim . 920, 925, 4348
 \l__enumext_fake_item_indent_viii_tl . . 922, 4346, 4351
 \l__enumext_fake_item_indent_X_tl 96
 __enumext_fake_item_vii: 877, 905, 3228
 __enumext_fake_item_viii: 877, 917, 3233
 __enumext_filter_first_level:n . . 118, 4493, 4493, 4527, 4538
 __enumext_filter_first_level_key:n 119, 4493, 4498, 4502
 __enumext_filter_first_level_pair:nn . 119, 4493, 4499, 4511
 __enumext_filter_save_key:n . . 70, 2088, 2096, 2119, 2125, 2127, 2127, 4425, 4429, 4433, 4437, 4441, 4445
 __enumext_filter_save_key_key:n . . 70, 2127, 2132, 2136
 __enumext_filter_save_key_pair:nn 70, 2127, 2133, 2144
 __enumext_filter_series:n 58, 1568, 1568, 1611, 1623, 1628
 __enumext_filter_series_key:n 58, 1568, 1573, 1577
 __enumext_filter_series_pair:nn . . 58, 1568, 1574, 1587
 \g__enumext_footnote_arg_seq . 156, 2972, 2985, 2995
 \g__enumext_footnote_int . 156, 2979, 2982, 2984, 2986
 \g__enumext_footnote_int_seq . 156, 2973, 2986, 2991, 2994
 __enumext_footnotes_key_bool 34
 \l__enumext_footnotes_key_bool 29, 34, 110, 146, 380, 385, 394, 4081, 4129, 4320, 4367
 __enumext_footnotetext:nn . . . 2966, 2966, 2996
 __enumext_getkeyans:nn . . 117, 4409, 4413, 4413
 __enumext_getkeyans_aux:n 116, 4397, 4400, 4400
 \l__enumext_hyperref_bool 29, 34, 146, 376, 397, 414, 2376, 2863, 4067, 4311
 __enumext_hypertarget:nn 34, 371, 399, 403, 419
 __enumext_if_is_int:n 202
 __enumext_if_is_int:nTF 202, 705, 719
 __enumext_internal_mini_page: 34, 107, 356, 356, 3261, 3935
 __enumext_is_not_nested: 26, 31, 93, 107, 222, 222, 3262, 3936
 __enumext_is_on_first_level: . 26, 31, 93, 107, 222, 246, 3267, 3943
 \g__enumext_item_anskey_int 76, 84, 138, 315, 342, 343, 1956, 2325, 2877
 __enumext_item_answer_diff: . . 66, 67, 82, 1952, 1952, 2749
 \g__enumext_item_answer_diff_int . 66, 67, 138, 316, 1954, 1961, 1985
 \l__enumext_item_column_pos_vii_int 108, 3713, 3719, 3725, 3729, 3736, 3995, 4133, 4136
 \l__enumext_item_column_pos_viii_int . . 113, 3762, 3768, 3774, 3778, 3785, 4215, 4372, 4375
 \l__enumext_item_column_pos_X_int 159
 \g__enumext_item_count_all_vii_int 108, 3737, 3996, 4144, 4151
 \g__enumext_item_count_all_viii_int 113, 3786, 4216, 4384, 4392
 \g__enumext_item_count_all_X_int 159
 \g__enumext_item_number_bool 138
 \l__enumext_item_number_bool 65, 144, 1906, 1911, 1915, 1919, 1932, 2497, 2551, 3006, 3037, 4075
 \g__enumext_item_number_int 65, 66, 138, 314, 341, 343, 1905, 1910, 1914, 1918, 1931, 1956, 3005, 3036, 4074
 __enumext_item_peek_args_vii: 108, 3997, 3999, 3999
 __enumext_item_peek_args_viii: . . 113, 4217, 4219, 4219
 __enumext_item_starred: . . 89, 3093, 3093, 3111
 \l__enumext_item_starred_vii_bool 4014, 4030, 4085
 \l__enumext_item_starred_viii_bool 4234, 4250, 4324, 4344
 \l__enumext_item_starred_X_bool 159
 __enumext_item_std:w . . 33, 87, 88, 101, 345, 349, 3009, 3015, 3040, 3058, 3062, 3070, 3618
 \g__enumext_item_symbol_aux_vii_tl 4038, 4087, 4090, 4094, 4096
 \g__enumext_item_symbol_aux_X_tl 159
 \l__enumext_item_symbol_sep_vii_dim . . 4047, 4055, 4093, 4095
 \g__enumext_item_symbol_tl . . . 87, 60, 122, 3024, 3099, 3116
 \l__enumext_item_symbol_vii_tl 4090
 \l__enumext_item_text_vii_box 4080, 4121, 4128
 \l__enumext_item_text_viii_box 4319, 4359, 4366
 \l__enumext_item_text_X_box 159
 \l__enumext_item_width_vii_dim . . 3652, 3661, 3740, 3748, 3749
 \l__enumext_item_width_viii_dim . . 3683, 3692, 3789, 3797, 3798
 \l__enumext_item_width_X_dim 159
 \l__enumext_itemindent_X_dim 70
 \l__enumext_itemsep_vii_skip 4150
 \l__enumext_itemsep_viii_skip 4391
 \l__enumext_joined_item_aux_vii_int . . 3734, 3735, 3736, 3737, 3743
 \l__enumext_joined_item_aux_viii_int . 3783, 3784, 3785, 3786, 3792
 \l__enumext_joined_item_aux_X_int . . . 159
 __enumext_joined_item_vii:w . . 108, 4002, 4003, 4005, 4005
 \l__enumext_joined_item_vii_int . . 3705, 3706, 3709, 3711, 3717, 3722, 3727, 3732, 3734, 3740
 __enumext_joined_item_viii:w . 113, 4222, 4223,

[4225](#), [4225](#)
`\l__enumext_joined_item_viii_int` . [3754](#), [3755](#),
[3758](#), [3760](#), [3766](#), [3771](#), [3776](#), [3781](#), [3783](#), [3789](#)
`\l__enumext_joined_item_X_int` [159](#)
`\l__enumext_joined_width_vii_dim` . [3738](#), [3745](#),
[3748](#), [4111](#), [4123](#)
`\l__enumext_joined_width_viii_dim` [3787](#), [3794](#),
[3797](#), [4341](#), [4361](#)
`\l__enumext_joined_width_X_dim` [159](#)
`__enumext_keyans_addto_prop:n` [82](#), [2762](#), [2762](#),
[3072](#), [3548](#)
`__enumext_keyans_addto_seq:n` . [84](#), [2836](#), [2836](#),
[3074](#), [3550](#)
`__enumext_keyans_addto_seq_link:` [2836](#), [2857](#),
[2859](#), [4281](#)
`__enumext_keyans_anspic_code:nnn` . [99](#), [3539](#),
[3542](#), [3542](#)
`__enumext_keyans_default_item:n` . . [88](#), [3053](#),
[3053](#), [3089](#)
`\l__enumext_keyans_env_bool` [34](#), [3292](#), [3305](#), [3434](#),
[3519](#)
`__enumext_keyans_fake_item:` . . [877](#), [893](#), [3186](#)
`\l__enumext_keyans_level_h_int` . . [28](#), [626](#), [653](#),
[2481](#), [2543](#), [2814](#), [4180](#), [4181](#)
`\l__enumext_keyans_level_int` . . [28](#), [1403](#), [2477](#),
[2539](#), [2809](#), [3433](#), [3438](#), [3533](#)
`__enumext_keyans_make_label:` [37](#), [90](#), [3119](#), [3119](#),
[3184](#)
`__enumext_keyans_mini_addvspace:` [53](#), [97](#), [1307](#),
[1307](#), [3463](#)
`__enumext_keyans_mini_right_cmd:n` [55](#), [1405](#),
[1428](#), [1428](#)
`__enumext_keyans_mini_set_vskip:` . [52](#), [1245](#),
[1245](#), [1309](#)
`__enumext_keyans_multi_addvspace:` [98](#), [1101](#),
[1112](#), [3488](#)
`__enumext_keyans_multi_set_vskip:` [49](#), [1101](#),
[1101](#), [1114](#)
`__enumext_keyans_multicols_start:` [97](#), [3467](#),
[3469](#), [3469](#)
`__enumext_keyans_multicols_stop:` . [98](#), [1432](#),
[3494](#), [3494](#), [3518](#)
`__enumext_keyans_parse_keys:n` [3412](#), [3447](#), [3447](#)
`\l__enumext_keyans_pic_above_int` . [133](#), [3628](#),
[3629](#), [3631](#)
`\l__enumext_keyans_pic_above_skip` . [101](#), [133](#),
[3572](#), [3612](#)
`__enumext_keyans_pic_arg_two:` [101](#), [3570](#), [3600](#),
[3600](#)
`\l__enumext_keyans_pic_below_int` . [133](#), [3628](#),
[3629](#), [3632](#)
`\l__enumext_keyans_pic_body_seq` . [99](#)–[101](#), [133](#),
[3537](#), [3577](#), [3636](#)
`__enumext_keyans_pic_do:n` [101](#), [3577](#), [3579](#), [3620](#),
[3620](#), [3624](#)
`\l__enumext_keyans_pic_level_int` . . [28](#), [1395](#),
[2485](#), [2547](#), [2765](#), [2804](#), [2839](#), [2917](#), [3588](#), [3589](#)
`__enumext_keyans_pic_row:n` . . [101](#), [3622](#), [3625](#),
[3625](#)
`__enumext_keyans_pic_safe_exec:` . [100](#), [3566](#),
[3586](#), [3586](#)
`__enumext_keyans_pic_skip_abs:N` . [101](#), [3595](#),
[3595](#), [3611](#)
`\l__enumext_keyans_pic_width_dim` . [133](#), [3627](#),
[3634](#)
`__enumext_keyans_redefine_item:` . . [89](#), [3077](#),
[3077](#), [3183](#)
`__enumext_keyans_ref:` [40](#), [678](#), [696](#), [3185](#)
`__enumext_keyans_ref:n` [40](#), [675](#), [678](#), [678](#)
`__enumext_keyans_safe_exec:` . [3411](#), [3427](#), [3427](#)
`__enumext_keyans_show_ans:` . . [2880](#), [2888](#), [2907](#)
`__enumext_keyans_show_item_opt:` . [2880](#), [2895](#),
[3070](#), [3562](#), [4347](#)
`__enumext_keyans_show_left:n` . [88](#), [2880](#), [2880](#),
[3068](#), [3557](#)
`__enumext_keyans_show_pos:` . . [2880](#), [2892](#), [2915](#)
`__enumext_keyans_starred_item:n` . . [88](#), [3065](#),
[3065](#), [3085](#)
`__enumext_keyans_start_line:` . [26](#), [32](#), [279](#), [279](#),
[3435](#), [3593](#), [4185](#)
`__enumext_keyans_store_ref:` . . [83](#), [2783](#), [2783](#),
[3073](#), [3549](#), [4279](#)
`__enumext_keyans_store_ref_aux_i:` [83](#), [2783](#),
[2795](#), [2798](#)
`__enumext_keyans_store_ref_aux_ii:` [83](#), [2783](#),
[2824](#), [2826](#)
`__enumext_keyans_wrapper_opt:n` . . [2031](#), [2903](#)
`\l__enumext_label_copy_i_tl` . . [2290](#), [2802](#), [2807](#),
[2812](#), [2817](#)
`\l__enumext_label_copy_v_tl` [2812](#)
`\l__enumext_label_copy_vi_tl` [2807](#)
`\l__enumext_label_copy_vii_tl` [2265](#), [2276](#), [2307](#),
[2802](#)
`\l__enumext_label_copy_viii_tl` [2817](#)
`\l__enumext_label_copy_X_tl` [148](#)
`\l__enumext_label_fill_left_v_tl` [3123](#)
`\l__enumext_label_fill_left_X_tl` [96](#)
`\l__enumext_label_fill_right_v_tl` [3130](#)
`\l__enumext_label_fill_right_X_tl` [96](#)
`\l__enumext_label_font_style_v_tl` [3124](#), [3561](#)
`\l__enumext_label_font_style_vii_tl` . . [4099](#)
`\l__enumext_label_font_style_viii_tl` . . [4329](#)
`\l__enumext_label_i_tl` [541](#)
`\l__enumext_label_ii_tl` [541](#)
`\l__enumext_label_iii_tl` [541](#)
`\l__enumext_label_iv_tl` [541](#)
`__enumext_label_style:Nnn` [26](#), [36](#), [455](#), [455](#), [470](#),
[546](#), [593](#), [664](#), [668](#)
`\l__enumext_label_v_tl` . . [82](#), [84](#), [661](#), [2770](#), [2844](#),
[2909](#), [2944](#), [3067](#), [3071](#), [3415](#), [3556](#), [3558](#)
`\l__enumext_label_vi_tl` . [82](#), [84](#), [661](#), [2767](#), [2841](#),
[3556](#), [3558](#), [3562](#)
`\l__enumext_label_vii_tl` . [588](#), [4025](#), [4050](#), [4057](#)
`\l__enumext_label_viii_tl` [588](#), [4245](#), [4273](#), [4277](#)
`\l__enumext_label_width_by_box` . . [66](#), [451](#), [452](#)
`__enumext_label_width_by_box:Nn` [36](#), [449](#), [449](#),
[454](#), [466](#), [729](#)
`\l__enumext_labelsep_i_dim` . . [2912](#), [2947](#), [4285](#),
[4300](#)
`\l__enumext_labelsep_v_dim` [3478](#)
`\l__enumext_labelsep_vii_dim` . [3647](#), [3657](#), [3741](#),
[4048](#), [4109](#), [4125](#)
`\l__enumext_labelsep_viii_dim` [3678](#), [3688](#), [3790](#),
[4339](#), [4348](#), [4363](#)
`\l__enumext_labelwidth_i_dim` . [2912](#), [2947](#), [4285](#),
[4300](#)
`\l__enumext_labelwidth_v_dim` [3478](#)
`\l__enumext_labelwidth_vii_dim` . . [3647](#), [3656](#),


```

    3741, 4102, 4106, 4124
\l__enumext_labelwidth_viii_dim .. 3678, 3687,
    3790, 4332, 4336, 4362
\l__enumext_leftmargin_tmp_v_bool . 101, 3602
\l__enumext_leftmargin_tmp_X_bool ..... 70
\l__enumext_leftmargin_tmp_X_dim ..... 70
\l__enumext_leftmargin_X_dim ..... 70
\__enumext_level: 198, 198, 570, 573, 574, 583, 585,
    880, 884, 888, 955, 959, 963, 967, 1050, 1052, 1054,
    1056, 1089, 1091, 1093, 1095, 1099, 1132, 1135, 1154,
    1163, 1169, 1174, 1178, 1189, 1193, 1194, 1199, 1235,
    1239, 1412, 1418, 1465, 1467, 1469, 1472, 1479, 1481,
    1483, 1486, 2083, 2085, 2087, 2115, 2116, 2118, 2174,
    2182, 2186, 2190, 2391, 2394, 2395, 3008, 3009, 3013,
    3014, 3015, 3022, 3024, 3028, 3029, 3032, 3039, 3040,
    3095, 3098, 3100, 3107, 3108, 3109, 3112, 3115, 3247,
    3249, 3298, 3311, 3318, 3329, 3331, 3334, 3335, 3337,
    3342, 3349, 3352, 3354, 3356, 3357, 3358, 3359, 3362,
    3368, 3373, 3379, 3382, 3384, 3390
\l__enumext_level_h_int 107, 28, 230, 252, 266, 609,
    646, 1902, 1922, 2284, 2301, 2517, 2529, 3306, 3937,
    3938
\l__enumext_level_int . 93, 28, 200, 239, 251, 267,
    358, 1062, 1206, 1399, 1896, 1928, 2261, 2271, 2277,
    2283, 2291, 2299, 2306, 2516, 2528, 2744, 3199, 3263,
    3264, 3274, 3282, 3296, 3309, 3364, 3442, 3529, 3976,
    3986, 4188
\__enumext_list_arg_two_i: ..... 3165
\__enumext_list_arg_two_ii: ..... 3165
\__enumext_list_arg_two_iii: ..... 3165
\__enumext_list_arg_two_iv: ..... 3165
\__enumext_list_arg_two_v: . 89, 3165, 3417, 3603
\__enumext_list_arg_two_vii: ..... 3205, 3918
\__enumext_list_arg_two_viii: .... 3205, 4163
\l__enumext_listoffset_v_dim ..... 3480
\l__enumext_listparindent_vii_dim .... 4112
\l__enumext_listparindent_viii_dim ... 4342
\__enumext_log_answer_vars: . 33, 330, 338, 2751
\__enumext_log_global_vars: . 33, 330, 330, 2750
\__enumext_make_label: 37, 87, 89, 3103, 3103, 3194
\l__enumext_mark_answer_sym_tl 72, 2037, 2240,
    2406, 2919, 2932, 4289
\l__enumext_mark_position_str 122, 2041, 2042,
    2068, 2069, 2238
\l__enumext_mark_ref_sym_tl .. 2054, 2381, 2871
\__enumext_mini_addvspace: .. 52, 94, 1228, 1228,
    3339
\__enumext_mini_addvspace_vii: 54, 1381, 1381,
    3815
\__enumext_mini_addvspace_viii: 54, 1381, 1387,
    3870
__enumext_mini_env* ..... 356
\__enumext_mini_right_cmd:n 55, 1407, 1409, 1409
\__enumext_mini_set_vskip: . 50, 1129, 1129, 1230
\__enumext_mini_set_vskip_vii: 53, 1324, 1324,
    1383
\__enumext_mini_set_vskip_viii: 53, 1324, 1346,
    1389
\__enumext_minipage:w 33, 351, 353, 362, 3634, 4111,
    4341
\l__enumext_minipage_active_v_bool .. 97, 98,
    3461, 3486, 3499, 3507
\g__enumext_minipage_active_vii_bool .. 105,
    3826, 3831, 3850
\l__enumext_minipage_active_vii_bool . 3811,
    3822
\g__enumext_minipage_active_viii_bool 3881,
    3886, 3905
\l__enumext_minipage_active_viii_bool 3866,
    3877
\g__enumext_minipage_active_X_bool ... 159
\l__enumext_minipage_active_X_bool .... 84
\g__enumext_minipage_after_skip 84, 1328, 1340,
    3848, 3903
\l__enumext_minipage_after_skip 50, 51, 96, 98,
    84, 1145, 1160, 1180, 1196, 1211, 1217, 1223, 1237,
    1247, 1256, 1259, 1271, 1289, 1300, 1316, 1348, 1361,
    1375, 3399, 3516
\g__enumext_minipage_center_vii_bool . 3835,
    3851
\g__enumext_minipage_center_viii_bool 3890,
    3906
\g__enumext_minipage_center_X_bool ... 159
\l__enumext_minipage_hsep_v_dim ... 97, 3459
\l__enumext_minipage_hsep_vii_dim .... 3809
\l__enumext_minipage_hsep_viii_dim ... 3864
\l__enumext_minipage_left_skip 50, 97, 84, 1137,
    1152, 1171, 1186, 1233, 1243, 1248, 1254, 1263, 1280,
    1292, 1312, 1322, 1326, 1331, 1335, 1349, 1353, 1367,
    1385, 1391
\l__enumext_minipage_left_v_dim 97, 3457, 3465
\l__enumext_minipage_left_vii_dim 3805, 3817
\l__enumext_minipage_left_viii_dim 3860, 3872
\l__enumext_minipage_left_X_dim ..... 84
\g__enumext_minipage_right_skip 84, 1327, 1332,
    1336, 3834, 3889
\l__enumext_minipage_right_skip . 50, 84, 1141,
    1156, 1176, 1191, 1249, 1255, 1267, 1285, 1296, 1350,
    1357, 1371, 1419, 1436
\l__enumext_minipage_right_v_dim .. 97, 1430,
    1435, 3455, 3459
\g__enumext_minipage_right_vii_dim 104, 3813,
    3833, 3853
\l__enumext_minipage_right_vii_dim 104, 3803,
    3808, 3814
\g__enumext_minipage_right_viii_dim .. 3868,
    3888, 3908
\l__enumext_minipage_right_viii_dim .. 3858,
    3863, 3869
\g__enumext_minipage_right_X_dim ..... 159
\g__enumext_minipage_right_X_skip .... 159
\g__enumext_minipage_stat_int . 94, 97, 84, 1424,
    1441, 3338, 3392, 3397, 3462, 3509, 3514
\l__enumext_miniright_code_vii_box 3842, 3846
\g__enumext_miniright_code_vii_tl 105, 3837,
    3844, 3852
\l__enumext_miniright_code_viii_box .. 3897,
    3901
\g__enumext_miniright_code_viii_tl 3892, 3899,
    3907
\l__enumext_miniright_code_X_box ..... 159
\__enumext_multi_addvspace: . 49, 95, 1084, 1084,
    3370
\__enumext_multi_set_vskip: 48, 1048, 1048, 1086
\l__enumext_multicols_above_ii_skip ... 1067
\l__enumext_multicols_above_iii_skip .. 1073
\l__enumext_multicols_above_iv_skip ... 1079
\l__enumext_multicols_above_v_skip 1103, 1117,
    1127

```

`\l__enumext_multicols_above_X_skip` 78
`\l__enumext_multicols_below_v_skip` 1107, 1121, 3501
`\l__enumext_multicols_below_X_skip` 78
`__enumext_multicols_start:` . 94, 95, 3344, 3346, 3346
`__enumext_multicols_stop:` 95, 1414, 3376, 3376, 3401
`__enumext_nested_base_line_fix:` 43, 107, 797, 807, 3278, 3953
`__enumext_newlabel:nn` 29, 35, 73, 407, 407, 2317, 2830
`\l__enumext_newlabel_arg_one_tl` 29, 35, 73, 83, 148, 2310, 2318, 2380, 2819, 2831, 2869
`\l__enumext_newlabel_arg_two_tl` 29, 35, 72, 148, 2264, 2274, 2288, 2304, 2319, 2806, 2811, 2816, 2832
`__enumext_parse_keys:n` 43, 59, 3243, 3269, 3269
`__enumext_parse_keys_vii:n` . 43, 59, 3913, 3945, 3945
`__enumext_parse_keys_viii:n` . 4158, 4193, 4193
`__enumext_parse_save_key:n` 69, 2108, 2113, 2113
`__enumext_parse_save_key_vii:n` 69, 2103, 2113, 2121
`__enumext_parse_series:n` 59, 93, 107, 1599, 1599, 3277, 3951
`__enumext_parse_store_keys:n` 93
`\l__enumext_parsep_i_skip` 1065, 1067, 1209, 1257
`\l__enumext_parsep_ii_skip` . . 1071, 1073, 1215
`\l__enumext_parsep_iii_skip` . . 1077, 1079, 1221
`\l__enumext_parsep_vii_skip` 4113
`\l__enumext_parsep_viii_skip` 4343
`\l__enumext_partopsep_v_skip` . 1119, 1123, 1283, 1287, 1294, 1298, 1314, 1318
`\l__enumext_partopsep_viii_skip` 1359
`__enumext_phantomsection:` 34, 371, 400, 404, 420
`__enumext_print_footnote:` . . 2966, 2989, 4131, 4369
`__enumext_print_keyans_box:NN` 72, 2232, 2232, 2245, 2393, 2911, 2946, 4285, 4300
`\l__enumext_print_keyans_i_tl` 4430, 4452
`\l__enumext_print_keyans_ii_tl` . . 4434, 4453
`\l__enumext_print_keyans_iii_tl` . . 4438, 4454
`\l__enumext_print_keyans_iv_tl` . . 4442, 4455
`\l__enumext_print_keyans_starred_tl` 117, 118, 122, 4426, 4474
`\l__enumext_print_keyans_vii_tl` 117, 4446, 4456
`\l__enumext_print_keyans_X_tl` 122
`__enumext_printkeyans:nnn` 117, 118, 4457, 4460, 4460
`__enumext_redefine_item:` . 88, 3042, 3042, 3193
`\l__enumext_ref_key_arg_tl` 38, 49, 213, 563, 564, 577, 608, 611, 622, 628, 639, 680, 681, 692
`\l__enumext_ref_the_count_tl` . 38, 49, 570, 573, 576, 616, 618, 621, 633, 635, 638, 686, 688, 691
`__enumext_regex_counter_style:` . . 30, 38, 208, 208, 571, 617, 634, 687
`__enumext_register_counter_style:Nn` . . 439, 439, 444, 445, 446, 447, 448
`__enumext_remove_extra_parsep_vii:` . . 3928, 4140, 4140
`__enumext_remove_extra_parsep_viii:` . 4173, 4379, 4379
`__enumext_renew_footnote:` . . 2966, 2970, 4083, 4322
`\l__enumext_renew_the_count_v_tl` 689, 698, 700
`\l__enumext_renew_the_count_vii_tl` 619, 648, 650
`\l__enumext_renew_the_count_viii_tl` 636, 655, 657
`\l__enumext_renew_the_count_X_tl` 49
`__enumext_rescan_anskey_env:n` . . 79, 81, 2626, 2721, 2729
`__enumext_reset_global_bool:` . . 306, 309, 318
`__enumext_reset_global_int:` . . 306, 308, 312
`__enumext_reset_global_tl:` 306, 310, 324
`__enumext_reset_global_vars:` . 32, 82, 306, 306, 2759
`\l__enumext_resume_active_bool` 59, 61, 60, 1603, 1723
`__enumext_resume_counter:` . . 60, 61, 1721, 1727, 1734
`__enumext_resume_counter:n` . 59, 61, 1692, 1697, 1721, 1721, 1791, 1799
`__enumext_resume_counter_save_ans:` . . 61, 62, 1721, 1732, 1764
`__enumext_resume_counter_series:` 61, 62, 1721, 1730, 1747
`\g__enumext_resume_int` . . 60, 1644, 1738, 1739
`__enumext_resume_last:n` . . 59, 1599, 1605, 1618
`\l__enumext_resume_name_tl` 60, 1640, 1648, 1651, 1667, 1675, 1678, 1724, 1725, 1753, 1760
`__enumext_resume_save_counter:` 59, 107, 1631, 1631, 3406, 3970
`__enumext_resume_series:n` . 60, 1562, 1688, 1688
`__enumext_resume_starred:` . 62, 1563, 1785, 1785
`\g__enumext_resume_vii_int` 60, 1671, 1743, 1744
`\l__enumext_rightmargin_vii_dim` . . 3659, 3663, 3668
`\l__enumext_rightmargin_viii_dim` . 3690, 3694, 3699
`__enumext_safe_exec:` . . 34, 93, 3242, 3259, 3259
`__enumext_safe_exec_vii:` . 34, 3912, 3933, 3933
`__enumext_safe_exec_viii:` . . 4157, 4178, 4178
`\l__enumext_series_name_tl` 61
`\l__enumext_series_str` . . 60, 93, 107, 1560, 1601, 1609, 1610, 1612, 1614, 1635, 1638, 1642, 1662, 1665, 1669, 3273, 3949
`__enumext_set_error:nn` 4588, 4598, 4600
`__enumext_set_parse:n` 4571, 4588, 4588
`\l__enumext_setkey_tmpa_int` . . 117, 4564, 4568
`\l__enumext_setkey_tmpa_seq` . . 117, 4562, 4572, 4578, 4580, 4582, 4595
`\l__enumext_setkey_tmpa_tl` 117, 4570, 4574
`\l__enumext_setkey_tmpb_seq` . . 117, 4563, 4566, 4570, 4571
`\l__enumext_setkey_tmpb_tl` 117, 4590, 4592, 4593
`\l__enumext_show_answer_bool` . 2048, 2072, 2400, 2886, 2900, 3553, 4283
`__enumext_show_length:nnn` . . 45, 216, 216, 4685, 4686, 4687, 4688, 4689, 4690, 4691, 4692, 4693, 4694, 4700, 4701, 4702, 4703, 4704, 4705, 4706, 4707, 4708, 4709
`\l__enumext_show_position_bool` . . 2051, 2075, 2404, 2890, 2901, 3554, 4287
`\g__enumext_standar_bool` 31, 93, 34, 229, 232, 250, 321, 1633, 1698, 1710, 1736, 1749, 1787, 1927, 1941, 3293
`\l__enumext_standar_bool` . 93, 96, 34, 2269, 2282, 2298, 3266, 3405

```

\l__enumext_standar_first_bool 31, 93, 34, 255,
810, 1620, 1767, 1829, 1836
\__enumext_standar_item_vii:w . 108, 109, 4010,
4012, 4012
\__enumext_standar_item_viii:w 113, 4230, 4232,
4232
\__enumext_standar_ref: . . . . 38, 561, 581, 3195
\__enumext_standar_ref:n . . . . 38, 553, 561, 561
\g__enumext_standar_series_tl . 60, 1622, 1623,
1789, 1792
\g__enumext_starred_bool 31, 107, 34, 238, 241, 265,
322, 1660, 1703, 1714, 1741, 1756, 1795, 1901, 1947,
2260, 2270, 2300, 2800, 3854
\l__enumext_starred_bool . . 107, 34, 2285, 2335,
2341, 2389, 2675, 2680, 3942, 3969
\__enumext_starred_columns_set_vii: . . 3641,
3641, 3921
\__enumext_starred_columns_set_viii: . 3641,
3672, 4166
\l__enumext_starred_first_bool 31, 107, 34, 270,
821, 1625, 1776, 1829, 1836
\__enumext_starred_item:nn . . 3018, 3018, 3048
\__enumext_starred_item_exec: 114, 4275, 4275,
4326
\__enumext_starred_item_vii:w . 108, 109, 4009,
4028, 4028
\__enumext_starred_item_vii_aux_i:w . . 4028,
4033, 4036
\__enumext_starred_item_vii_aux_ii:w . 4028,
4034, 4039, 4041
\__enumext_starred_item_vii_aux_iii:w 4028,
4044, 4053
\__enumext_starred_item_viii:w 113, 114, 4229,
4248, 4248
\__enumext_starred_item_viii_aux_i:w . . 114,
4248, 4253, 4256
\__enumext_starred_item_viii_aux_ii:w . 114,
4248, 4254, 4268, 4270
\__enumext_starred_joined_item_vii:n 103, 108,
3703, 3703, 4007
\__enumext_starred_joined_item_viii:n . 103,
113, 3703, 3752, 4227
\__enumext_starred_ref: . . . . 39, 606, 644, 3225
\__enumext_starred_ref:n . . . . 39, 600, 606, 606
\g__enumext_starred_series_tl . 60, 1627, 1628,
1797, 1800
\__enumext_start_from:NNn 41, 703, 703, 716, 738
\l__enumext_start_i_int . . . . 1739, 1751, 1770
\__enumext_start_item_tmp_vii: 106, 3924, 3992,
3992
\__enumext_start_item_tmp_viii: . . 111, 4169,
4212, 4212
\__enumext_start_item_vii:w 109, 110, 4020, 4025,
4050, 4057, 4059, 4059
\__enumext_start_item_viii:w . . 113, 4240, 4245,
4273, 4303, 4303
\g__enumext_start_line_tl 31, 34, 258, 273, 327,
1971, 1976, 1981, 1995, 2000, 2005
\__enumext_start_list:nn . . 33, 90, 101, 345, 347,
3246, 3414, 3567, 3916, 4161
\__enumext_start_mini_vii: 107, 3801, 3801, 3961
\__enumext_start_mini_viii: . . 112, 3856, 3856,
4204
\__enumext_start_save_ans_msg: 63, 1813, 1813,
1838
\__enumext_start_store_level: . 93, 3245, 3287,
3287
\__enumext_start_store_level_vii: 108, 3915,
3972, 3972
\l__enumext_start_vii_int . . . 1744, 1758, 1779
\l__enumext_start_X_int . . . . . 96, 733
\__enumext_stop_item_tmp_vii: . . 106, 108, 110,
3923, 3927, 3994, 4061
\__enumext_stop_item_tmp_viii: 111, 113, 4168,
4172, 4214, 4305
\__enumext_stop_item_vii: 110, 111, 4061, 4116,
4116
\__enumext_stop_item_viii: 115, 4305, 4354, 4354
\__enumext_stop_list: . . 33, 345, 348, 3255, 3424,
3580, 3929, 4175
\__enumext_stop_mini_vii: 105, 107, 3801, 3820,
3965
\__enumext_stop_mini_viii: 112, 3856, 3875, 4208
\__enumext_stop_save_ans_msg: . 63, 1813, 1818,
2748
\__enumext_stop_store_level: . . 93, 3256, 3287,
3316
\__enumext_stop_store_level_vii: . 108, 3930,
3972, 3982
\l__enumext_store_active_bool . 28, 64, 93, 108,
1768, 1777, 1845, 2473, 3291, 3304, 3429, 3437, 3525,
3584, 3974, 3984, 4187
\__enumext_store_active_keys:n 69, 2081, 2081,
3284
\__enumext_store_active_keys_vii:n . 69, 107,
2081, 2091, 3952
\__enumext_store_addto_prop:n 70, 82, 2156, 2156,
2164, 2326, 2781, 4278
\__enumext_store_addto_seq:n 71, 84, 2165, 2165,
2169, 2176, 2190, 2198, 2207, 2221, 2229, 2384, 2874
\l__enumext_store_anskey_arg_tl . . 28, 74, 108,
2332, 2337, 2339, 2344, 2351, 2354, 2364, 2369, 2372,
2378, 2384
\__enumext_store_anskey_code:n 74, 76, 81, 2323,
2323, 2466, 2719, 2727
\l__enumext_store_anskey_env_tl . . 28, 80, 108,
2649, 2653, 2659, 2721, 2729
\l__enumext_store_anskey_opt_tl 28, 80, 81, 108,
2650, 2677, 2683, 2690, 2696, 2706, 2716, 2725
\__enumext_store_anskey_safe_outer: . . . . 76
\g__enumext_store_columns_break_bool . 2573,
2674, 2736
\l__enumext_store_columns_break_bool . 2334,
2422
\l__enumext_store_current_label_tl 28, 82, 84,
114, 108, 2764, 2767, 2770, 2777, 2779, 2781, 2838,
2841, 2844, 2850, 2855, 2865, 2874, 4258, 4263, 4264,
4277, 4278, 4280
\l__enumext_store_current_label_tmp_tl . 28,
108, 3067, 3071
\l__enumext_store_current_opt_arg_tl 28, 114,
108, 2884, 2897, 2903, 4266
\__enumext_store_internal_ref: . . 72, 74, 2246,
2246, 2329
\g__enumext_store_item_join_int . . 2576, 2681,
2685, 2737
\l__enumext_store_item_join_int . . 2342, 2346,
2425
\g__enumext_store_item_star_bool . 2578, 2688,

```


2738
 \l__enumext_store_item_star_bool . 2349, 2427
 \g__enumext_store_item_symbol_sep_dim 2583,
 2703, 2708, 2740
 \l__enumext_store_item_symbol_sep_dim 2361,
 2366, 2432
 \g__enumext_store_item_symbol_tl . 2581, 2694,
 2698, 2739
 \l__enumext_store_item_symbol_tl . 2352, 2356,
 2430
 \l__enumext_store_keyans_item_opt_sep_-
 tl 2034, 2775, 2777, 2848, 2852, 4261, 4263
 __enumext_store_level_close: . 71, 2170, 2194,
 3320
 __enumext_store_level_close_vii: . 71, 2201,
 2225, 3988
 __enumext_store_level_open: . 71, 2170, 2170,
 3299, 3312
 __enumext_store_level_open_vii: . 71, 2201,
 2201, 3978
 \g__enumext_store_name_tl . 28, 64, 96, 108, 326,
 333, 334, 335, 336, 1821, 1847, 1970, 1975, 1980, 1994,
 1999, 2004, 2746
 \l__enumext_store_name_tl 28, 63, 65, 108, 1654,
 1657, 1681, 1684, 1772, 1781, 1816, 1825, 1826, 1847,
 1848, 1849, 1851, 1852, 1854, 1856, 1857, 1859, 1861,
 1862, 1886, 2158, 2160, 2167, 2312, 2313, 2412, 2655,
 2821, 2822, 2925, 2938, 4295
 \l__enumext_store_ref_key_bool 74, 2057, 2327,
 2375, 2785, 2862
 \l__enumext_store_save_key_vii_bool . 2093,
 2123
 \l__enumext_store_save_key_vii_tl 2095, 2096,
 2124, 2125, 2205, 2213, 2217, 2221
 \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, 64, 78, 1823, 1839,
 1843
 __enumext_storing_set:n . 63, 1808, 1823, 1823
 \l__enumext_the_counter_v_tl 688
 \l__enumext_the_counter_vii_tl 618
 \l__enumext_the_counter_viii_tl 635
 \l__enumext_the_counter_X_tl 49
 __enumext_tmp:n 44, 48, 53, 59, 70, 77, 78, 83, 90, 95,
 96, 107, 125, 132, 151, 155, 159, 179, 797, 806, 1556,
 1567, 1804, 1812, 1865, 1883, 2024, 2062, 2063, 2080,
 2099, 2112, 2248, 2255, 2256, 2277, 2291, 2294, 2306,
 2787, 2794, 3165, 3204, 3205, 3239
 __enumext_tmp:nn 471, 492, 493, 524, 525, 540, 733,
 752, 833, 855, 856, 876, 929, 937, 938, 952, 1017, 1033,
 1034, 1047, 1445, 1461, 2950, 2965
 __enumext_tmp:nnn 541, 557, 558, 559, 560, 588, 604,
 605
 __enumext_tmp:nnnnn 753, 778, 781, 784, 786, 788,
 791, 794
 __enumext_tmp:w 4406, 4409
 \l__enumext_tmpa_vii_int 3651, 3654, 3663, 3694
 \l__enumext_tmpa_viii_int 3682, 3685
 \l__enumext_tmpa_X_dim 159
 \l__enumext_tmpa_X_int 159
 \l__enumext_topsep_v_skip 1105, 1109, 1252, 1265,
 1273, 1278, 1298, 1302, 3583, 3615
 \l__enumext_topsep_vii_skip . 1329, 1338, 1342
 \l__enumext_topsep_viii_skip . 1351, 1373, 1377
 __enumext_undefine_anskey_env: . 77, 82, 2506,
 2506, 2757
 \l__enumext_vspace_a_star_v_bool 1494
 \l__enumext_vspace_a_star_vii_bool . . 1516
 \l__enumext_vspace_a_star_viii_bool . . 1527
 \l__enumext_vspace_a_star_X_bool 96
 __enumext_vspace_above: . 56, 1462, 1462, 3325
 __enumext_vspace_above_v: . 57, 1490, 1490, 3453
 \l__enumext_vspace_above_v_skip . 1492, 1496,
 1498
 __enumext_vspace_above_vii: 57, 107, 1512, 1512,
 3958
 \l__enumext_vspace_above_vii_skip 1514, 1518,
 1520
 __enumext_vspace_above_viii: . 57, 1512, 1523,
 4202
 \l__enumext_vspace_above_viii_skip 1525, 1529,
 1531
 \l__enumext_vspace_b_star_v_bool 1505
 \l__enumext_vspace_b_star_vii_bool . . 1538
 \l__enumext_vspace_b_star_viii_bool . . 1549
 \l__enumext_vspace_b_star_X_bool 96
 __enumext_vspace_below: . 56, 1476, 1476, 3404
 __enumext_vspace_below_v: . 57, 1501, 1501, 3521
 \l__enumext_vspace_below_v_skip . 1503, 1507,
 1509
 __enumext_vspace_below_vii: 57, 107, 1534, 1534,
 3968
 \l__enumext_vspace_below_vii_skip 1536, 1540,
 1542
 __enumext_vspace_below_viii: . 57, 1534, 1545,
 4210
 \l__enumext_vspace_below_viii_skip 1547, 1551,
 1553
 __enumext_widest_from:nnn . 41, 717, 717, 732,
 744
 \g__enumext_widest_label_tl 27, 36, 66, 459, 463,
 467
 \l__enumext_wrap_label_opt_v_bool . . . 3061
 \l__enumext_wrap_label_opt_vii_bool 109, 4019
 \l__enumext_wrap_label_opt_viii_bool . 113,
 4239
 \l__enumext_wrap_label_opt_X_bool 96
 \l__enumext_wrap_label_v_bool 3057, 3061, 3069,
 3125
 \l__enumext_wrap_label_vii_bool . 109, 4018,
 4023, 4031, 4100
 \l__enumext_wrap_label_viii_bool . 113, 4238,
 4243, 4251, 4330
 \l__enumext_wrap_label_X_bool 96
 __enumext_wrapper_label_v:n 3127, 3562
 __enumext_wrapper_label_vii:n 4103
 __enumext_wrapper_label_viii:n 4333
 \l__enumext_write_aux_file_tl . 29, 73, 83, 148,
 2315, 2321, 2828, 2834
 __enumext_zero_parsep: . . 51, 1149, 1204, 1204
 enumext* 5, 3910
 enumXi 431
 enumXii 431
 enumXiii 431
 enumXiv 431
 enumXv 431
 enumXvi 431
 enumXvii 431

enumXviii 431
 Environments provide by **enumext**:
 anskey* 28, 64, 72, 73, 75, 77, 78, 80, 82, 93, 108, 117, 121, 123
 enumext* 25, 26, 29–31, 34, 35, 39, 41–47, 53, 54, 57–60, 62, 63, 65, 66, 68–77, 80, 82, 83, 86, 92–94, 102, 103, 105, 108, 110, 111, 113, 115, 117, 118, 122, 125
 enumext 25, 26, 30, 31, 34–43, 45–53, 55, 56, 58–60, 62, 63, 65, 66, 68–77, 80, 82, 83, 86–94, 96, 100, 102, 104, 107, 108, 117, 118, 122, 123
 keyans* 25, 26, 28–32, 35, 39–47, 53, 54, 57, 64, 67, 68, 70, 78, 83, 86, 92, 102, 104, 112, 122, 125
 keyanspic 25, 26, 28, 32, 35, 37, 40, 55, 64, 67, 70, 71, 78, 82–84, 99–101, 124
 keyans 25, 26, 28, 31, 32, 35–37, 40–43, 45–47, 49, 52, 53, 55–57, 64, 67, 68, 70, 71, 78, 82–84, 89–91, 96, 97, 99, 101, 104, 113, 122, 124

Environments:

list 30, 33, 90, 92
 lrbox 102, 110, 111, 115
 minipage . 30, 33, 34, 47, 49, 50, 99–102, 110, 111, 115
 multicols 48–50, 55, 94–98
 scontents 78, 80

exp commands:

\exp_after:wN 4409
 \exp_args:Ne 2718, 2726, 3281, 4397
 \exp_args:NV 2438, 2593
 \exp_not:N . 57, 462, 576, 621, 638, 691, 886, 900, 901, 912, 913, 924, 925, 2380, 2409, 2410, 2867, 2922, 2923, 2935, 2936, 4292, 4293, 4406
 \exp_not:n 260, 275, 287, 294, 301, 515, 535, 576, 577, 621, 622, 638, 639, 691, 692, 887, 1585, 1597, 2045, 2142, 2154, 2318, 2346, 2356, 2366, 2380, 2381, 2685, 2698, 2708, 2831, 2869, 2871, 4509, 4519

F

\fbox 2029
 file commands:
 \file_input_stop: 4898
 first 938
 font 471
 \footnote 86
 \footnote 86, 2974
 \footnotemark 2984
 \footnotesize 2410, 2923, 2936, 4293
 \footnotetext 2968

G

\getkeyans 16, 116, 4395
 group commands:
 \group_begin: . . 2408, 2453, 2628, 2715, 2921, 2934, 4079, 4098, 4291, 4318, 4328, 4417, 4451
 \group_end: 2415, 2469, 2732, 2928, 2941, 4108, 4120, 4298, 4338, 4358, 4419, 4458

H

\hbadness 4127, 4365
 hbox commands:
 \hbox_set:Nn 451
 \hfill 501, 505, 510, 511, 1416, 1434, 2380, 2867, 3825, 3880
 hook commands:
 \hook_gput_code:nnn 9, 188, 192, 196, 369
 \hook_gremove_code:nn 80, 2644
 \hook_gset_rule:nnnn 370
 \hook_if_empty:nTF 2642
 \hspace 4138, 4377

\hyperlink 74, 84
 \hyperlink 2380, 2867
 \hypertarget 34
 \hypertarget 399

I

\IfHyperBoolean 377
 \IfPackageLoadedTF 11, 19, 373, 387
 \ignorespaces 889
 \inputlineno 260, 275, 287, 294, 301
 int commands:
 \int_add:Nn 3736, 3785
 \int_case:nn . . . 1062, 1206, 1896, 1922, 1961, 1985
 \int_compare:nNnTF . . 358, 609, 626, 646, 653, 1131, 1250, 1395, 1399, 1403, 2009, 2015, 2477, 2481, 2485, 2493, 2539, 2543, 2547, 2744, 2765, 2804, 2809, 2814, 2839, 2917, 3264, 3274, 3296, 3309, 3348, 3364, 3378, 3392, 3438, 3442, 3471, 3496, 3509, 3529, 3533, 3589, 3706, 3716, 3732, 3755, 3765, 3781, 3938, 3976, 3986, 4133, 4142, 4181, 4188, 4371, 4381, 4568
 \int_compare_p:nNn . . . 230, 239, 251, 252, 266, 267, 1902, 1928, 2261, 2271, 2283, 2284, 2299, 2301, 2342, 2516, 2517, 2528, 2529, 2681, 3306
 \int_decr:N 3735, 3784
 \int_eval:n . 343, 2160, 2313, 2410, 2822, 2923, 2936, 3180, 3224, 3724, 3773, 4293
 \int_from_alph:n 711, 725
 \int_from_roman:n 713, 727
 \int_gadd:Nn 3737, 3786
 \int_gdecr:N 1905, 1910, 1914, 1918, 1931
 \int_gincr:N 1738, 1743, 2325, 2877, 3005, 3036, 3075, 3338, 3462, 3551, 3996, 4074, 4216, 4282
 \int_gset:Nn 1954, 2982
 \int_gset_eq:NN 1637, 1644, 1650, 1656, 1664, 1671, 1677, 1683, 2979
 \int_gzero:N . 314, 315, 316, 1424, 1441, 2021, 2737, 3397, 3514, 4151, 4392
 \int_if_exist:NNTF 1612, 1648, 1654, 1675, 1681, 1859
 \int_incr:N 2492, 3263, 3433, 3588, 3937, 3995, 4180, 4215
 \int_mod:nn 4144, 4383
 \int_new:N . 28, 29, 30, 31, 32, 33, 60, 61, 84, 100, 119, 135, 136, 141, 142, 143, 145, 156, 162, 163, 164, 165, 166, 1614, 1862
 \int_set:Nn 707, 711, 713, 1751, 1758, 1770, 1779, 2629, 3628, 3629, 3651, 3682, 3705, 3711, 3727, 3754, 3760, 3776, 4127, 4365, 4564
 \int_set_eq:NN 1739, 1744, 3734, 3783
 \int_sign:n 1956
 \int_step_function:nnN 2277, 2291, 2306
 \int_step_inline:nnn 3630
 \int_to_roman:n 200, 2257, 2295
 \int_use:N 336, 341, 342, 1132, 1753, 1760, 1772, 1781, 3180, 3199, 3224, 3282, 3349, 3358, 3373, 3379, 3709, 3710, 3722, 3758, 3759, 3771
 \int_zero:N 4136, 4375
 \item . 86–88, 108, 110, 113, 115, 349, 2178, 2184, 2209, 2215, 2339, 2841, 2844, 3044, 3079, 3922, 3924, 4167, 4169, 4280
 \item* 5, 14, 67, 3077
 item-pos* 2950
 item-sym* 2950
 \itemindent 91
 \itemindent 90
 itemindent 833

```
\itemsep ..... 100, 101
\itemsep ..... 3604, 3610
\itemwidthdim ..... 185, 3745, 3749, 3794, 3798
```

K

keyans	14, 3409
keyans*	14, 4155
keyanspic	15, 3564

Keys for command provide by `enumext`:

break-col	74, 75, 78–80
item-join	74, 75, 78–80
item-pos*	74, 75, 78, 79, 81
item-star	74, 75, 78, 79, 81
item-sym*	74, 75, 78, 79, 81

Keys for environments provide by `enumext`:

above*	27, 56, 57, 107
above	27, 56, 57, 94, 97, 107, 112
after	45–47, 96, 99, 107, 112
align	27, 37, 89, 110, 121
base-fix	42, 58, 70, 107, 118
before*	45, 46, 94, 112
before	45, 46, 97
below*	27, 56, 57, 107
below	27, 56, 57, 96, 99, 107, 112
check-ans	28, 30, 31, 63–67, 70, 82, 84, 87, 88, 94, 96, 107, 111, 122
columns-sep	47, 95, 98
columns	27, 47, 50, 56, 95, 97
first	45–47, 110
font	36, 89, 110
item-pos*	86, 87
item-sym*	28, 86, 87
itemindent	27, 43, 44, 89, 110
itemsep	42, 92
labelsep	36, 87, 91, 110
labelwidth	36, 38–41, 91
label	26, 27, 36, 38, 41, 102
lisparindent	92
list-indent	27, 43, 44, 101
list-offset	43, 44
listparindent	43, 110
mark-ans	68, 70, 75
mark-pos	68, 121
mark-ref	68, 70, 72, 74
mini-env	27, 34, 47, 55, 56, 70, 86, 94, 97, 104, 105, 107, 112
mini-right*	27, 30, 47, 70, 105, 107
mini-right	27, 30, 47, 54, 70, 105, 107
mini-sep	27, 47, 70, 94, 97
no-store	28, 63–65, 70, 76
noitemsep	42, 51
nosep	42, 51
parindent	92
parsep	42, 92, 110
partopsep	42
ref	26, 30, 38–40, 122
resume*	26, 58, 59, 62–64, 70, 96, 107, 119
resume	26, 33, 58–64, 70, 96, 107, 119
rightmargin	43, 102
save-ans	28, 33, 58–63, 65, 66, 69–71, 76–78, 82, 84, 88, 93, 96, 99, 113, 114, 116, 117, 119, 122
save-key	28, 58, 69, 93, 107
save-pos	70
save-ref	29, 35, 68, 70, 72, 74, 83, 84, 88, 114
save-sep	68, 70, 114

series	26, 58–62, 70, 93, 96, 107, 119
show-ans	68, 70, 72, 74, 75, 88, 114
show-length	31, 45, 122
show-pos	28, 68, 72, 74, 75, 84, 88, 114
start	27, 30, 41, 58
store-key	69
topsep	42
widest	27, 30, 41
wrap-ans	30, 68, 70, 72, 75
wrap-label*	36, 87, 89, 109, 110, 113
wrap-label	36, 89, 109, 110, 113
wrap-opt	68, 70

keys commands:

```

\keys_define:nn 473, 495, 527, 543, 590, 661, 735, 755,
    799, 835, 858, 931, 940, 1019, 1036, 1447, 1558, 1806,
    1867, 2026, 2065, 2101, 2106, 2420, 2571, 2607, 2952,
    4422, 4521
\l_keys_key_str . . . . . 76, 79, 2438, 2593, 4670
\keys_precompile:nnN . 117, 4421, 4424, 4428, 4432,
    4436, 4440, 4444
\keys_set:nn . 487, 815, 826, 1042, 1452, 1457, 1700,
    1705, 1792, 1800, 2458, 3276, 3281, 3449, 3950, 4197,
    4476, 4483, 4525, 4530, 4531, 4532, 4533, 4536, 4541,
    4542, 4543, 4544, 4545, 4546, 4547, 4585
\keys_set_known:nn . . . . . 2725
keyval commands:
\keyval_parse:NNn . . . . . 1572, 2131, 4497

```

L

Label	<u>541</u> , <u>588</u> , <u>661</u>
Labels provide by enumext :	
\Alph*	36
\Roman*	36
\alph*	36
\arabic*	30, 36
\roman*	36
\labelsep	101
\labelsep	3605, 3608
labelsep	<u>471</u>
\labelwidth	36, 101
\labelwidth	3605, 3606
labelwidth	<u>471</u>
\leftmargin	91
\leftmargin	90, 3605

legacy commands:

```

\legacy_if:nTF ..... 4062, 4065, 4306, 4309
\legacy_if_gset_false:n ..... 363
\legacy_if_set_false:n ..... 4064, 4308
\legacy_if_set_true:n 4024, 4049, 4056, 4069, 4244,
    4272, 4313
\linewidth ..... 94, 97
\linewidth .... 3333, 3459, 3627, 3654, 3685, 3807, 3862
\list ..... 347
list-indent ..... 833
list-offset ..... 833
\listparindent ..... 3607
listparindent ..... 833
\lrbox ..... 4080, 4319

```

M

<code>\makebox</code>	102
<code>\makebox</code>	.. 2236, 2238, 3099, 4094, 4102, 4106, 4332, 4336	
<code>\makeLabel</code>	86, 89, 90, 102
<code>\makeLabel</code>	89, 90, 3105, 3121
<code>\makesavenoteenv</code>	393

mark-ans 2024

mark-pos 2024, 2063

mark-ref 2024

mini-env 1017

mini-sep 1017

\minipage 353

\miniright 10, 54, 1393, 3395, 3512

mode commands:

 \mode_if_math:TF 2501, 2555

 \mode_if_vertical:TF 1087, 1115, 1231, 1310

 \mode_leave_vertical: 813, 824, 886, 900, 912, 924, 2234, 3097, 4092

msg commands:

 \msg_error:nn .. 2462, 2495, 2499, 2553, 2661, 3440, 3444, 3531, 3591, 3940, 4183, 4190, 4548

 \msg_error:nnn 566, 613, 630, 683, 1397, 1401, 1426, 1443, 1712, 1716, 1831, 2444, 2503, 2521, 2533, 2541, 2545, 2549, 2557, 2599, 4411, 4416, 4490, 4601

 \msg_error:nnnn 2447, 2475, 2479, 2483, 2487, 2602, 3431, 3527, 3535, 4471

 \msg_error:nnnnn 514, 534, 2044

 \msg_fatal:nn 3265

 \msg_fatal:nnn 425

 \msg_info:nnn 13, 16, 21, 24, 375, 389

 \msg_line_context: .. 4635, 4640, 4645, 4674, 4679, 4684, 4699, 4714, 4718, 4722, 4726, 4730, 4734, 4741, 4748, 4754, 4768, 4772, 4777, 4781, 4785, 4789, 4794, 4798, 4802, 4806, 4811, 4846, 4850, 4855, 4860, 4864, 4869, 4873, 4878, 4883, 4888, 4892, 4896

 \msg_log:nnn 1851, 1856, 1861

 \msg_log:nnnnn 340, 1994, 1999, 2004

 \msg_log:nnnnnn 332

 \msg_new:nnn 4602, 4606, 4610, 4614, 4619, 4632, 4637, 4642, 4647, 4656, 4664, 4668, 4672, 4677, 4682, 4697, 4712, 4716, 4720, 4724, 4728, 4732, 4736, 4745, 4751, 4757, 4761, 4765, 4770, 4775, 4779, 4783, 4787, 4792, 4796, 4800, 4804, 4809, 4844, 4848, 4853, 4858, 4862, 4867, 4871, 4876, 4881, 4886, 4890, 4894

 \msg_new:nnnn 4623, 4814, 4823, 4832, 4838

 \msg_term:nnnn . 1815, 1820, 3189, 3199, 3230, 3235

 \msg_term:nnnnn 1975

 \msg_warning:nn 3394, 3511

 \msg_warning:nnnn 2012, 2018, 3137, 3142, 3708, 3721, 3757, 3770

 \msg_warning:nnnnn 1970, 1980

\multicolsep 95, 98

\multicolsep 3363, 3484

N

\NeedsTeXFormat 3

\newcounter 428

\NewDocumentCommand 1393, 2450, 3523, 4395, 4449, 4555

\NewDocumentEnvironment . 3240, 3409, 3564, 3910, 4155

\newenvsc 2564

\newlabel 35

\newlabel 411

no-store 1865

\noindent . 3340, 3464, 3816, 3871, 3923, 4135, 4168, 4374

\nointerlineskip 3340, 3464, 3816, 3871

noitemsep 753

\nopagebreak 1098, 1126, 1242, 1321, 1384, 1390

\normalfont 2409, 2922, 2935, 4292

nosep 753

P

Packages:

 caption 105

 enumext 25, 30, 38, 63, 90, 99, 120, 121

 enumitem 35, 36

 expl3 102

 footnotehyper 34

 hyperref 29, 30, 34, 35, 74, 84, 110, 120

 lua-visual-debug 50

 multicol 25, 120

 scontents 25, 77, 78

 shortlst 102

\par .. 1098, 1126, 1242, 1321, 1384, 1390, 1419, 1436, 2388, 3384, 3399, 3501, 3516, 3639, 3834, 3848, 3889, 3903, 4135, 4149, 4374, 4390

\parindent 4112, 4342

\parsep 48, 51, 100, 101

\parsep 3221, 3604, 3611, 3616

parsep 753

\parskip 4113, 4343

\partopsep 101

\partopsep 3222, 3609

partopsep 753

peek commands:

 \peek_meaning:NTF 4001, 4015, 4032, 4043, 4221, 4235, 4252

 \peek_meaning_remove:NTF 4008, 4228

 \peek_remove_spaces:n 3083

\phantomsection 34

\phantomsection 400

prg commands:

 \prg_do_nothing: 404

 \prg_new_protected_conditional:Npnn ... 202

 \prg_replicate:nn 219

 \prg_return_false: 206

 \prg_return_true: 205

\printkeyans 16, 117, 4449

prop commands:

 \prop_count:N 334, 2160, 2313, 2412, 2822, 2925, 2938, 4295

 \prop_gput_if_not_in:Nnn 2158

 \prop_if_exist:NTF 1849, 4415

 \prop_item:Nn 4418

 \prop_new:N 1852

\ProvidesExplPackage 4

R

\raggedcolumns 3372, 3490

\ref 72, 83

ref 541, 588, 661

\refstepcounter 4071, 4315

regex commands:

 \regex_match:nnTF .. 204, 710, 712, 724, 726, 2657

 \regex_replace_once:nnN 212

\renewcommand 576, 621, 638, 691

\RenewDocumentCommand ... 2974, 3044, 3079, 3105, 3121

\RequirePackage 17, 25

resume 1556

resume* 1556

rightmargin 833

\Roman 36, 41

\Roman 447

\roman 36, 41

\roman 448, 559, 4439

S

<code>\s</code>	2658
<code>save-ans</code>	1804
<code>save-key</code>	2099
<code>save-ref</code>	2024
<code>save-sep</code>	2024
scan commands:	
<code>\scan_stop:</code>	101, 3618, 3922, 4167, 4406, 4409
scontents internal commands:	
<code>\l_scontents_fname_out_tl</code>	2617
<code>_scontents_parse_environment_keys:n</code>	2623
<code>_scontents_rescan_tokens:n</code>	2630
<code>\l_scontents_storing_bool</code>	2615
<code>\l_scontents_writing_bool</code>	2616
seq commands:	
<code>\seq_clear:N</code>	4562
<code>\seq_const_from_clist:Nn</code>	4550
<code>\seq_count:N</code>	335, 3577, 4566
<code>\seq_gclear:N</code>	2972, 2973
<code>\seq_gput_right:Nn</code>	2167, 2985, 2986
<code>\seq_if_empty:NTF</code>	2991, 4464, 4580
<code>\seq_if_exist:NTF</code>	1854, 4462
<code>\seq_if_in:NnTF</code>	4469
<code>\seq_item:Nn</code>	2655, 3636
<code>\seq_map_function:NN</code>	4571
<code>\seq_map_inline:Nn</code>	4477, 4484, 4559, 4581, 4582
<code>\seq_map_pairwise_function:NNN</code>	2993
<code>\seq_new:N</code>	120, 121, 133, 157, 158, 1857
<code>\seq_pop_left:NN</code>	4570
<code>\seq_put_right:Nn</code>	3537, 4578, 4595
<code>\seq_set_from_clist:Nn</code>	4563
<code>\seq_set_map_e:NNn</code>	4572
<code>\seq_show:N</code>	4466
series	1556
<code>\setcounter</code>	721, 725, 727, 3180, 3224, 3582
<code>\setenumext</code>	6, 118, 4555
<code>show-ans</code>	2024, 2063
<code>show-length</code>	929
<code>show-pos</code>	2063
skip commands:	
<code>\skip_add:Nn</code>	1067, 1073, 1079, 1089, 1093, 1117, 1121, 1211, 1217, 1223, 1233, 1237, 1259, 1312, 1316, 3604
<code>\skip_gset:Nn</code>	1332, 1336, 1340
<code>\skip_gzero_new:N</code>	1327, 1328
<code>\skip_horizontal:N</code>	901, 913, 925, 4095, 4109, 4339
<code>\skip_horizontal:n</code>	887, 2235, 2243, 3098, 3100, 4093, 4348
<code>\skip_if_eq:nnTF</code>	1065, 1071, 1077, 1134, 1168, 1209, 1215, 1221, 1252, 1257, 1278, 1329, 1351, 1464, 1478, 1492, 1503, 1514, 1525, 1536, 1547
<code>\skip_new:N</code>	80, 81, 85, 86, 87, 88, 89, 137, 177
<code>\skip_set:Nn</code>	1050, 1054, 1103, 1107, 1137, 1141, 1145, 1152, 1156, 1160, 1171, 1176, 1180, 1186, 1191, 1196, 1254, 1255, 1256, 1263, 1267, 1271, 1280, 1285, 1289, 1292, 1296, 1300, 1331, 1335, 1353, 1357, 1361, 1367, 1371, 1375, 3598, 3612
<code>\skip_set_eq:NN</code>	3178, 3220, 3221, 4112, 4113, 4342, 4343
<code>\skip_use:N</code>	1052, 1056, 1091, 1095, 1099, 1119, 1123, 1135, 1154, 1163, 1169, 1174, 1178, 1189, 1193, 1194, 1199, 1235, 1239, 1265, 1465, 1469, 1472, 1479, 1483, 1486, 3384
<code>\skip_zero:N</code>	3222, 3363, 3484, 3609, 3610
<code>\skip_zero_new:N</code>	1247, 1248, 1249, 1326, 1348, 1349, 1350

<code>\c_zero_skip</code>	1065, 1071, 1077, 1135, 1169, 1209, 1215, 1221, 1252, 1257, 1278, 1329, 1351, 1465, 1479, 1492, 1503, 1514, 1525, 1536, 1547
<code>\small</code>	4427, 4431, 4435, 4439, 4443, 4447
<code>\star</code>	2956
<code>start</code>	733
<code>\stepcounter</code>	2978, 3544
str commands:	
<code>\c_backslash_str</code>	2503, 4635, 4640, 4645, 4650, 4652, 4654, 4659, 4661, 4759, 4763, 4767, 4777, 4781, 4789, 4790, 4794, 4806, 4807, 4811, 4812, 4833, 4835, 4839, 4841, 4873, 4874, 4878, 4883, 4884
<code>\c_colon_str</code>	2312, 2821, 4406
<code>\c_left_brace_str</code>	4740, 4747, 4753
<code>\c_right_brace_str</code>	4740, 4747, 4753
<code>\str_case:nn</code>	224, 281
<code>\str_case:nnTF</code>	1579, 1589, 2138, 2146, 4504, 4513
<code>\str_clear:N</code>	3273, 3949
<code>\str_count:n</code>	219
<code>\str_if_empty:NTF</code>	1601, 1642, 1669
<code>\str_if_eq:nnTF</code>	3181, 3226
<code>\str_if_in:nnTF</code>	4402
<code>\str_new:N</code>	123, 172
<code>\str_set:Nn</code>	530, 531, 532, 2041, 2042, 2068, 2069
<code>\string</code>	393
<code>\strutbox</code>	1139, 1143, 1147, 1158, 1162, 1173, 1182, 1188, 1198, 1211, 1217, 1223, 1254, 1255, 1256, 1259, 1269, 1273, 1282, 1289, 1294, 1302, 1331, 1332, 1335, 1342, 1355, 1363, 1369, 1377, 3614

T

\TeX and \LaTeX commands:

<code>\@auxout</code>	409
<code>\@currenvir</code>	224, 281
<code>\protected@write</code>	409
tex commands:	
<code>\tex_newlinechar:D</code>	2629
text commands:	
<code>\text_expand:n</code>	4398
<code>\textasteriskcentered</code>	2038, 2055
<code>\thepage</code>	415
tl commands:	
<code>\c_space_tl</code>	2903, 4684, 4699, 4722, 4726
<code>\tl_clear:N</code>	500, 506, 2022, 2085, 2095, 2116, 2124, 2332, 2649, 2650, 2764, 2838, 4258
<code>\tl_clear_new:N</code>	457
<code>\tl_const:Nn</code>	49, 441
<code>\tl_gclear:N</code>	326, 327, 328, 1622, 1627, 2739, 3116, 3852, 3907, 4096
<code>\tl_gclear_new:N</code>	1609
<code>\tl_gput_right:Nn</code>	442
<code>\tl_greplace_all:Nnn</code>	463
<code>\tl_gset:Nn</code>	257, 258, 272, 273, 1610, 1623, 1628, 1847, 2653, 4038
<code>\tl_gset_eq:NN</code>	459, 3024, 4089
<code>\tl_if_blank:nTF</code>	2442, 2460, 2597, 4087
<code>\tl_if_empty:NTF</code>	564, 583, 611, 628, 648, 655, 681, 698, 1635, 1640, 1662, 1667, 1725, 1789, 1797, 1826, 1886, 2174, 2205, 2352, 2694, 2716, 2746, 2775, 2848, 2897, 3095, 4261, 4593
<code>\tl_if_empty:nTF</code>	1690
<code>\tl_if_exist:NTF</code>	1695
<code>\tl_if_novalue:nTF</code>	2456, 2772, 2846, 2882, 2976, 3001, 3020, 3025, 3055, 3271, 3575, 3947, 4195, 4259, 4557

