

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-14^{*}

©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	Public dimension	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>multicol</code>	3	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	Support for <code>minipage</code>	4	5.3	The environment <code>anskey*</code>	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	5.4	The environment <code>keyans</code>	13
1.3.6	Support for <code>\footnote</code>	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	14
2	The environments provided	4	5.5	The environment <code>keyanspic</code>	14
2.1	The environment <code>enumext</code>	4	5.5.1	The command <code>\anspic</code>	15
2.2	The environment <code>enumext*</code>	5	5.6	Printing stored content	15
2.3	The command <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.3.1	Keys for <code>\item*</code>	5	5.6.2	The command <code>\printkeyans</code>	16
2.4	The command <code>\item</code> in <code>enumext*</code>	5	6	Full examples	17
3	The command <code>\setenumext</code>	6	7	The way of non-enumerated lists	19
4	The <code>keyval</code> system	6	8	References	21
4.1	Keys for label and ref	6	9	Change history	22
4.2	Keys for spaces	7	10	Index of Documentation	23
4.2.1	Vertical spaces	7	11	Implementation	25
4.2.2	Horizontal spaces	8	12	Index of Implementation	128
4.3	Keys for add code	8			
4.4	Keys for start, series and resume	9			
4.5	Keys for <code>multicols</code>	9			
4.6	Keys for <code>minipage</code>	9			
4.6.1	The command <code>\miniright</code>	10			
4.6.2	The key <code>mini-right</code>	10			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-06-14.

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

1 Introduction

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

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

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

3. True False

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

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

*

$(x - 1)^2$

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

*

$3(x + y + z)$

3. True False

(a) $\alpha > \delta$

*

False

(b) L^AT_EXze is cool?

*

Very True!

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

*

Yes

(b) Usually uses the package manager?

*

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

*

doesn’t exist for now :(

ii. xsim

*

very good

iii. exsheets

*

obsolete

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

* (c) i. doesn’t exist for now :(

ii. very good

iii. obsolete

*

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

★ 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

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

A

A)

B

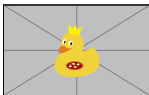
B)

A

C)

A

D)



E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

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

2 / 141

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 Public dimension

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

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

1.3.3 Support for multicol

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

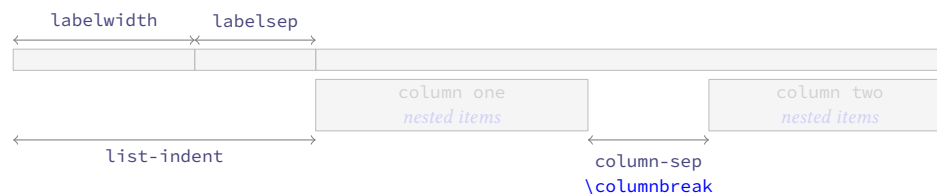


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

1.3.4 Support for minipage

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



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

The `minipage` environments on “left side” and “right side” is always used with “aligned on top” `[t]`. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the “left” side `minipage` environment to the “right” side one must use the command `\miniright` (see §4.6).

1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem[6]` package to customize the references which is activated by the `ref` key (§4.1), the standard \LaTeX `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§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.

1.3.6 Support for \footnote

This package provides an internal implementation for the `\footnote` command which is compatible with the `hyperref` package for the `enumext*` and `keyans*` environments, but will not produce the expected links, and if the `mini-env` key is used in `enumext` or `keyans` environments the output will look like the classic way they are displayed in the environment `minipage`.
The best way to solve this is to use Jean-François Burnol `footnotehyper`[9] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

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

2 The environments provided

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

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

2.1 The environment enumext

The `enumext` is an environment that works in the same way as the standard `enumerate` environment provided by 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 `\itemwidth` that includes `labelwidth`, `labelsep` plus the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.

Example with columns=2

1. This text is in the first level.
2. This text is in the first level.
- X This text is in the first level.
- ★ 3. This text is in the first level.

2.3 The command \item*

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

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

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

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

2.3.1 Keys for `\item*`

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

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

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

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

```
\begin{enumext*}[widest=10,columns=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item The seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
  \item The ninth
  \item[Z] 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 8. The eighth item is way too long for this and needs two columns
8. The ninth
- Z 9. The tenth

3 The command `\setenumext`

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

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

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

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

4 The keyval system

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

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

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

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

4.1 Keys for label and ref

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

Sets the $\langle label \rangle$ that will be printed at the *current level*. The default value for the first level of the environments `enumext` and `enumext*` are `\arabic*`, for second level are `(\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 $\langle label \rangle$ will be displayed, and the form in which it is used by standard “*label and ref*” and the “*internal reference*” system with the `save-ref` key. You cannot use commands with $\langle label \rangle$ as an argument, for example `\emph{⟨ \backslash alph*⟩}` will return an error. For full customization of how $\langle label \rangle$ is displayed use the `font` or `wrap-label` keys.

`ref = {⟨code ⟨ \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash 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{⟨ \backslash 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 $\langle label \rangle$ defined by `label` key and the text of an item on the first line. Internally sets the value of `\labelsep` for the current level.

`labelwidth = {⟨rigid length⟩}` default: *by label*

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

`widest = {⟨integer | string⟩}` default: *empty*

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

`font = {⟨font commands⟩}` default: *empty*

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

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

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

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

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

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

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

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

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

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] \langle item's before \rangle \item \miniright \langle content \rangle \end{enumext}
\begin{enumext}[mini-env=\langle rigid length \rangle] \langle item's before \rangle \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 §1.3.6 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*` = $\langle \text{content} \rangle$ 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}]]
  \item Text \anskey{answer}
  \item Text
    \begin{keyans}
      ...
    \end{keyans}
\end{enumext}

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

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

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

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

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

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

5.1.1 Keys for label and ref

`save-ref = { \langle true | false \rangle }` default: *false*
Activates the “*internal label and ref*” mechanism for referencing “*stored content*” in $\langle store name \rangle$ set by `save-ans` key. To reference the location of the “*stored content*” within the environment you must use `\ref{\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+\parbox{#1}`
 Wraps the *argument* passed to the `\anskey` and the *body* in `anskey*` environment referenced by `{#1}` when using the `show-ans` or `show-pos` keys. The `{code}` must be passed between braces and only affects the *argument* or *body* and NOT the “stored content” in the *sequence* and *prop list* `{store name}` set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double `{##1}`.
- `wrap-opt` = {`{code {#1} more code}`} default: `[{#1}]`
 Wraps the *optional argument* passed to the `\item*` and `\anspic*` referenced by `{#1}` in the `keyans`, `keyans*` and `keyanspic` environments when using the `show-ans` or `show-pos` keys. The `{code}` must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* `{store name}` set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double `{##1}`.
- `show-ans` = {`{true | false}`} default: `false`
 Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the `{label}` for `\item*` and `\anspic*` at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.
- `mark-ans` = {`{symbol}`} default: `\textasteriskcentered`
 Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.
- `mark-pos` = {`{left | right}`} default: `left`
 Sets the *aligned* of the symbol defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

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*` $\langle ast \rangle$ $\langle content \rangle$.

`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. <li style="margin-left: 20px;">* <input type="text" value="first answer"/> 2. Text containing our instructions or questions. <li style="margin-left: 20px;">(a) Question. <li style="margin-left: 40px;">* <input type="text" value="second answer"/> | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. <li style="margin-left: 20px;">* <input type="text" value="third answer"/> 4. Text containing our instructions or questions. <li style="margin-left: 20px;">* <input type="text" value="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*}
  \fourth answer
\end{anskey*}
\end{enumext}

```

★ 5. Text containing our instructions or questions.

[5] First answer with verbatim

6. Text containing our instructions or questions.

(a) Question.

[6] second answer

7. Text containing our instructions or questions.

[7] third answer

8. Text containing our instructions or questions.

[8] fourth answer

5.4 The environments `keyans` and `keyans*`

```

keyans \begin{keyans}[\key = val] \item \item[\custom] \item* \item*[\content] \end{keyans}
keyans* \begin{keyans*}[\key = val] \item \item[\custom] \item* \item*[\content] \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[\custom]` work in the usual and the command `\item[\content]` is available for the `keyans*` environment.

```

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

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

```

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

5.4.1 The `\item*` in `keyans` and `keyans*`

```

\item* \item*
\item*[\content]

```

The `\item*` and `\item*[\content]` command “store” the current `\label` set by `label` key next to the `\content` (if it is present) in *sequence* and *prop list* `{\store name}` 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*[\note] 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.

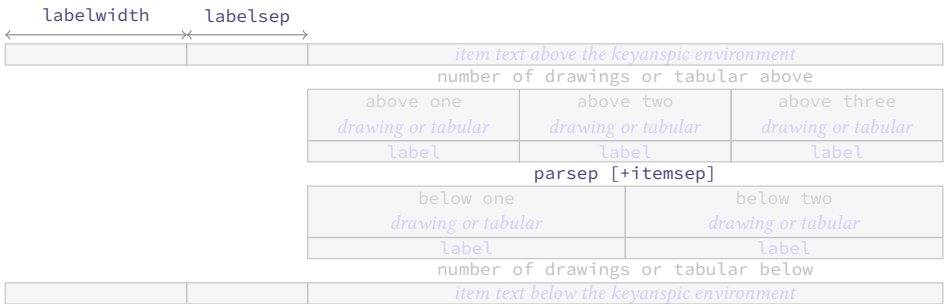


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

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) You use linux?
- [3]

(b) Rate the following package and class

i.

[4]

ii.

[5]

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

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:

```
$ pdfdetach -saveall enumext.pdf
```

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

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

2. A

3. B

4. A

Example 2

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

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

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

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.

long Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing

vitaie, felis. Curabitur dictum gravida mauris.

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 2024-06-14 – First public release.

10 Index of Documentation

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

C

Document class:

article

book

exam

letter

report

\columnbreak

\columnsep

Commands provide by enumext:

\anskey

\anspic

\getkeyans

\item*

\item

\miniright

\printkeyans

\setenumext

Counters defined by enumext:

enumXiii

enumXii

enumXiv

enumXi

enumXviii

enumXvii

enumXvi

enumXv

E

Environments provide by enumext:

anskey*

enumext*

enumext

keyans*

keyanspic

keyans

Environments:

enumerate

figure

list

minipage

multicols

table

task

F

\footnote

I

\itemsep

K

Keys for command provide by enumext:

break-col

item-join

item-pos*

item-star

item-sym*

Keys for environments provide by enumext:

above*

above

after

align

base-fix

before*

before

below*

below

check-ans

columns-sep

columns

first

font

item-pos*

item-sym*

itemindent

itemsep

labelsep

labelwidth

labelwith

label

list-indent

list-offset

listparindent

mark-ans

mark-pos

mark-ref

mini-env

mini-right*

mini-right

mini-sep

no-store

noitemsep

nosep

parsep

partopsep

ref

resume*

resume

rightmargin

save-ans

save-key

save-ref

save-sep

series

show-ans

show-length

show-pos

start

topsep

widest

wrap-ans

wrap-label*

wrap-label

wrap-opt

L

\label

Labels provide by enumext:

\Alph*

\Roman*

\alph*

\arabic*

©2024 by Pablo González L

23 / 141

<code>\roman*</code>	7	<code>l3seq</code>	1, 21
<code>\labelsep</code>	3, 7	<code>multicol</code>	1, 2, 4, 21
<code>\labelwidth</code>	3, 7	<code>scontents</code>	1, 2, 13
<code>\linewidth</code>	10	<code>task</code>	5, 6
<code>\listparindent</code>	9	<code>xsim</code>	2
P		<code>\parsep</code>	8
Packages:		<code>\partopsep</code>	8
<code>enumerate</code>	21	R	
<code>enumext</code>	1–6, 15, 21	<code>\raggedcolumns</code>	4
<code>enumitem</code>	3–5, 9, 20, 21	<code>\ref</code>	4
<code>footnotehyper</code>	5	<code>\rightmargin</code>	9
<code>hyperref</code>	4, 5, 11–13, 21	T	
<code>l3keys</code>	6	<code>\topsep</code>	8
<code>l3prop</code>	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   {2024-06-14}
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
\l__enumext_envir_name_tl
```

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

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

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

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

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

The initial values of these variables are set by the function `__enumext_define_counters:Nn` (§11.10) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.13).

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

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

```
\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl
```

Internal variables used by `ref` key (§11.13).

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

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

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

Internal variables used by `resume`, `resume*` and `series` keys (§11.24).

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

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


```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default `<label style>` and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§11.14) and `label` (§11.12) keys.

```

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

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§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).

```

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

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

```

Internal variables used by `columns` key §11.21).

```

79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \skip_new:c { l__enumext_multicols_above_#1_skip }
82   \skip_new:c { l__enumext_multicols_below_#1_skip }
83 }
84 \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).

```

85 \int_new:N \g__enumext_minipage_stat_int
86 \skip_new:N \l__enumext_minipage_left_skip
87 \skip_new:N \l__enumext_minipage_right_skip
88 \skip_new:N \l__enumext_minipage_after_skip
89 \skip_new:N \g__enumext_minipage_right_skip
90 \skip_new:N \g__enumext_minipage_after_skip
91 \cs_set_protected:Npn \__enumext_tmp:n #1
92 {
93   \dim_new:c { l__enumext_minipage_left_#1_dim }
94   \bool_new:c { l__enumext_minipage_active_#1_bool }
95 }
96 \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).

```

97 \cs_set_protected:Npn \__enumext_tmp:n #1
98 {
99   \bool_new:c { l__enumext_wrap_label_#1_bool }
100   \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
101   \int_new:c { l__enumext_start_#1_int }
102   \tl_new:c { l__enumext_fake_item_indent_#1_tl }
103   \tl_new:c { l__enumext_label_fill_left_#1_tl }
104   \tl_new:c { l__enumext_label_fill_right_#1_tl }
105   \bool_new:c { l__enumext_vspace_a_star_#1_bool }
106   \bool_new:c { l__enumext_vspace_b_star_#1_bool }
107 }
108 \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*` (§11.36) 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.

```
109 \bool_new:N \l__enumext_store_active_bool
110 \tl_new:N \l__enumext_store_name_tl
111 \tl_new:N \g__enumext_store_name_tl
112 \tl_new:N \l__enumext_store_anskey_arg_tl
113 \tl_new:N \l__enumext_store_anskey_env_tl
114 \tl_new:N \l__enumext_store_anskey_opt_tl
115 \tl_new:N \l__enumext_store_current_label_tl
116 \tl_new:N \l__enumext_store_current_opt_arg_tl
117 \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` (§11.47).

```
118 \tl_new:N \l__enumext_setkey_tmpa_tl
119 \tl_new:N \l__enumext_setkey_tmpb_tl
120 \int_new:N \l__enumext_setkey_tmpa_int
121 \seq_new:N \l__enumext_setkey_tmpa_seq
122 \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_aux_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool
\l__enumext_store_upper_level_X_bool
```

Internal variables used by command `\printkeyans` (§11.46), `show-pos` key (§11.26), `item-sym*` key (§11.34), `save-key` key (§11.26.2) and “*storage level system*”.

```
123 \tl_new:N \l__enumext_print_keyans_starred_tl
124 \str_new:N \l__enumext_mark_position_str
125 \tl_new:N \g__enumext_item_symbol_aux_tl
126 \cs_set_protected:Npn \__enumext_tmp:n #1
127 {
128   \tl_new:c { \l__enumext_print_keyans_#1_tl }
129   \tl_new:c { \l__enumext_store_save_key_#1_tl }
130   \bool_new:c { \l__enumext_store_save_key_#1_bool }
131   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
132 }
133 \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).

```
134 \seq_new:N \l__enumext_keyans_pic_body_seq
135 \dim_new:N \l__enumext_keyans_pic_width_dim
136 \int_new:N \l__enumext_keyans_pic_above_int
137 \int_new:N \l__enumext_keyans_pic_below_int
138 \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.

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

```

144 \int_new:N \g__enumext_item_number_int
145 \bool_new:N \l__enumext_item_number_bool
146 \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 (§11.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

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

```

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

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

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

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

```

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

```

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

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

Internal variables used for redefinition of `\footnote` (§11.42).

```

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

```

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

```

\l__enumext_item_starred_X_bool
\l__enumext_item_column_pos_X_int
\g__enumext_item_count_all_X_int
\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_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.

```

160 \cs_set_protected:Npn \__enumext_tmp:n #1
161 {
162   \bool_new:c { l__enumext_item_starred_#1_bool }
163   \int_new:c { l__enumext_item_column_pos_#1_int }
164   \int_new:c { g__enumext_item_count_all_#1_int }
165   \int_new:c { l__enumext_joined_item_#1_int }
166   \int_new:c { l__enumext_joined_item_aux_#1_int }
167   \int_new:c { l__enumext_tmpa_#1_int }
168   \dim_new:c { l__enumext_tmpa_#1_dim }
169   \box_new:c { l__enumext_item_text_#1_box }
170   \dim_new:c { l__enumext_joined_width_#1_dim }
171   \dim_new:c { l__enumext_item_width_#1_dim }
172   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
173   \str_new:c { l__enumext_align_label_#1_str }
174   \bool_new:c { g__enumext_minipage_active_#1_bool }
175   \box_new:c { l__enumext_miniright_code_#1_box }
176   \bool_new:c { g__enumext_minipage_center_#1_bool }
177   \dim_new:c { g__enumext_minipage_right_#1_dim }
178   \skip_new:c { g__enumext_minipage_right_#1_skip }
179 }
180 \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`.

```

181 \clist_const:Nn \c__enumext_all_envs_clist
182 {
183   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
184   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
185 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

11.5 Some utility functions

`\keys_precompile:neN` Non-standard kernel variant used by the `\printkeyans` command (§11.46).

```
186 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
```

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

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

```
187 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
188 {
189   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
190 }
```

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

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

```
191 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
192 {
193   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
194 }
195 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
196 {
197   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
198 }
```

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

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

```
199 \cs_new:Nn \__enumext_level:
200 {
201   \int_to_roman:n { \__enumext_level_int }
202 }
```

(End of definition for `__enumext_level:.`)

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

```
203 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
204 {
205   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
206   { \prg_return_true: }
207   { \prg_return_false: }
208 }
```

(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.

```
209 \cs_new_protected:Nn \__enumext_regex_counter_style:
210 {
211   \tl_map_inline:Nn \c__enumext_counter_style_tl
212   {
213     \regex_replace_once:nnN { \c{##1}\* }
214     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \__enumext_ref_key_arg_tl
215   }
216 }
```

(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.

```
217 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
218 {
219   * ~ #2
220   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
221   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
222 }
```

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

11.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`
`__enumext_is_on_first_level:`

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

```

223 \cs_new_protected:Nn \__enumext_is_not_nested:
224 {
225   \str_case:en { \@currenvir }
226   {
227     {enumext}
228     {
229       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
230       \bool_lazy_and:nnT
231       { \bool_not_p:n { \g__enumext_standar_bool } }
232       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
233       {
234         \bool_gset_true:N \g__enumext_standar_bool
235       }
236     }
237     {enumext*}
238     {
239       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
240       \bool_lazy_and:nnT
241       { \bool_not_p:n { \g__enumext_starred_bool } }
242       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
243       {
244         \bool_gset_true:N \g__enumext_starred_bool
245       }
246     }
247   }
248 }

```

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.

```

249 \cs_new_protected:Nn \__enumext_is_on_first_level:
250 {
251   \bool_lazy_all:nT
252   {
253     { \bool_if_p:N \g__enumext_standar_bool }
254     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
255     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
256   }
257   {
258     \bool_set_true:N \l__enumext_standar_first_bool
259     \bool_set_true:N \l__enumext_anskey_env_bool
260     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
261     \tl_gset:Nn \g__enumext_start_line_tl
262     {
263       on ~ line ~ \exp_not:V \inputlineno
264     }
265   }
266   \bool_lazy_all:nT
267   {
268     { \bool_if_p:N \g__enumext_starred_bool }
269     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
270     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
271   }
272   {
273     \bool_set_true:N \l__enumext_starred_first_bool
274     \bool_set_true:N \l__enumext_anskey_env_bool
275     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
276     \tl_gset:Nn \g__enumext_start_line_tl
277     {
278       on ~ line ~ \exp_not:V \inputlineno
279     }
280   }
281 }

```

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

`__enumext_keyans_name_and_start:`

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

```

282 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
283 {
284   \str_case:en { \@currentenvir }
285   {
286     {keyans}
287     {
288       \tl_set:Nn \__enumext_envir_name_tl { keyans }
289       \tl_set:Nc \__enumext_check_start_line_env_tl
290       {
291         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
292       }
293     }
294     {keyans*}
295     {
296       \tl_set:Nn \__enumext_envir_name_tl { keyans* }
297       \tl_set:Nc \__enumext_check_start_line_env_tl
298       {
299         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
300       }
301     }
302     {keyanspic}
303     {
304       \tl_set:Nn \__enumext_envir_name_tl { keyanspic }
305       \tl_set:Nc \__enumext_check_start_line_env_tl
306       {
307         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
308       }
309     }
310   }
311 }

```

(End of definition for `__enumext_keyans_name_and_start:`)

11.5.2 Utilities for log and terminal

`__enumext_reset_global_vars:`

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

`__enumext_reset_global_int:`

`__enumext_reset_global_bool:`

`__enumext_reset_global_tl:`

```

312 \cs_new_protected:Nn \__enumext_reset_global_vars:
313 {
314   \__enumext_reset_global_int:
315   \__enumext_reset_global_bool:
316   \__enumext_reset_global_tl:
317 }
318 \cs_new_protected:Nn \__enumext_reset_global_int:
319 {
320   \int_gzero:N \__enumext_item_number_int
321   \int_gzero:N \__enumext_item_anskey_int
322   \int_gzero:N \__enumext_item_answer_diff_int
323 }
324 \cs_new_protected:Nn \__enumext_reset_global_bool:
325 {
326   \bool_gset_false:N \__enumext_check_ans_key_bool
327   \bool_gset_false:N \__enumext_standar_bool
328   \bool_gset_false:N \__enumext_starred_bool
329 }
330 \cs_new_protected:Nn \__enumext_reset_global_tl:
331 {
332   \tl_gclear:N \__enumext_store_name_tl
333   \tl_gclear:N \__enumext_start_line_tl
334   \tl_gclear:N \__enumext_envir_name_tl
335 }

```

(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.

```

336 \cs_new_protected:Nn \__enumext_log_global_vars:
337 {
338   \msg_log:nneeee { enumext } { prop-seq-int-hook }
339   { \g__enumext_store_name_tl }
340   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
341   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
342   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
343 }

```

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.

```

344 \cs_new_protected:Nn \__enumext_log_answer_vars:
345 {
346   \msg_log:nneeee { enumext } { item-answer-hook }
347   { \int_use:N \g__enumext_item_number_int }
348   { \int_use:N \g__enumext_item_anskey_int }
349   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
350 }

```

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

11.6 Copying list and minipage environments

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

```

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

```

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

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

```

351 \__enumext_at_begin_document:n
352 {
353   \cs_new_eq:NN \__enumext_start_list:nn \list
354   \cs_new_eq:NN \__enumext_stop_list: \endlist
355   \cs_new_eq:NN \__enumext_item_std:w \item
356 }

```

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

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

```

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

```

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

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

```

357 \__enumext_at_begin_document:n
358 {
359   \cs_new_eq:NN \__enumext_minipage:w \minipage
360   \cs_new_eq:NN \__enumext_endminipage: \endminipage
361 }

```

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

11.7 The internal minipage environment

`__enumext_internal_mini_page:`
`__enumext_mini_env*`

The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` environment (*custom version of minipage*) setting the `\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 (§11.43)

```

362 \cs_new_protected:Nn \__enumext_internal_mini_page:
363 {
364   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
365   {
366     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
367     {
368       \__enumext_minipage:w [ t ] { ##1 }
369       \legacy_if_gset_false:n { @minipage }
370       \vspace { 0pt }
371     }
372     { \__enumext_endminipage: }
373   }
374 }
```

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

11.8 Compatibility with hyperref and footnotehyper

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

```

375 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
376 \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”.

```

377 \cs_new_protected:Nn \__enumext_after_hyperref:
378 {
379   \IfPackageLoadedTF { hyperref }
380   {
381     \msg_info:nnn { enumext } { package-load } { hyperref }
382     \bool_set_true:N \l__enumext_hyperref_bool
383     \IfHyperBoolean{hyperfootnotes}
384     {
385       \typeout{hyperfootnotes=true}
386       \bool_set_true:N \l__enumext_footnotes_key_bool
387     }
388     { \typeout{hyperfootnotes=false} }
389   }
390   { }
```

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`.

```

391 \bool_if:NT \l__enumext_footnotes_key_bool
392 {
393   \IfPackageLoadedTF { footnotehyper }
394   {
395     \msg_info:nnn { enumext } { package-load } { footnotehyper }
396   }
397   {
398     \typeout{No ~ footnotehyper ~ load}
399     \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
400     \bool_set_false:N \l__enumext_footnotes_key_bool
401   }
402 }
```

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.

```

403   \bool_if:NTF \__enumext_hyperref_bool
404   {
405     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
406     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
407   }
408   {
409     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
410     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
411   }
412 }

```

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

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

#1: `__enumext_newlabel_arg_one_tl`

#2: `__enumext_newlabel_arg_two_tl`

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

```

413 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
414 {
415   \protected@write \@auxout { }
416   {
417     \token_to_str:N \newlabel {#1}
418     {
419       {#2}
420       \bool_if:NT \__enumext_hyperref_bool
421       { { \thepage } {#2} {#1} }
422       { }
423     }
424   }
425   \__enumext_hypertarget:nn {#1} { }
426   \__enumext_phantomsection:
427 }

```

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

11.9 Definition of public dimension

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

```

428 \dim_zero_new:N \itemwidth

```

11.10 Definition of counters

`__enumext_define_counters:Nn`
`__enumext_define_counters:cn`

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

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

#2: The counter’s name.

```

429 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
430 {
431   \cs_if_exist:cTF { c@ #2 }
432   { \msg_fatal:nnn { enumext } { counters } { #2 } }
433   {
434     \tl_set:Nn #1 { #2 }
435     \newcounter { #2 }
436   }
437 }

```

(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   438 \__enumext_define_counters:Nn \__enumext_counter_i_tl   { enumXi   }
enumXii  439 \__enumext_define_counters:Nn \__enumext_counter_ii_tl  { enumXii  }
enumXiii 440 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii 441 \__enumext_define_counters:Nn \__enumext_counter_iv_tl  { enumXiv   }
enumXviii

```

```

442 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
443 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
444 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
445 \__enumext_define_counters:Nn \__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.

```

446 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
447 {
448   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
449   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
450 }
451 \__enumext_register_counter_style:Nn \arabic { 0 }
452 \__enumext_register_counter_style:Nn \Alph { M }
453 \__enumext_register_counter_style:Nn \alph { m }
454 \__enumext_register_counter_style:Nn \Roman { VIII }
455 \__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.

```

456 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
457 {
458   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
459   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
460 }
461 \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.

```

462 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
463 {
464   \tl_clear_new:N #1
465   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
466   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
467   \tl_map_inline:Nn \g__enumext_counter_styles_tl
468   {
469     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
470     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
471     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
472   }
473   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
474   { \tl_use:N \g__enumext_widest_label_tl }
475   \tl_set_eq:cN { the #2 } #1
476 }
477 \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 Definition of keys `font`, `labelsep`, `labelwidth`, `wrap-label` and `wrap-label*` keys for `enumext` and `keyans` environments.

```

478 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
479 {
480   \keys_define:nn { enumext / #1 }
481   {
482     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
483     font .value_required:n = true,
484     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
485     labelsep .initial:n = {0.3333em},
486     labelsep .value_required:n = true,
487     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
488     labelwidth .value_required:n = true,
489     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
490     wrap-label .initial:n = {##1},
491     wrap-label .value_required:n = true,
492     wrap-label* .code:n = {
493       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
494       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
495     },
496     wrap-label* .value_required:n = true,
497   }
498 }
499 \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.

```

500 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
501 {
502   \keys_define:nn { enumext / #1 }
503   {
504     align .choice:,
505     align / left .code:n =
506       {
507         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
508         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
509       },
510     align / right .code:n =
511       {
512         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
513         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
514       },
515     align / center .code:n =
516       {
517         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
518         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
519       },
520     align / unknown .code:n =
521       \msg_error:nnee { enumext } { unknown-choice }
522       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
523     align .initial:n = left,
524     align .value_required:n = true,
525   }
526 }
527 \clist_map_inline:nn
528 {
529   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
530 }
531 { \__enumext_tmp:nn #1 }

532 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
533 {
534   \keys_define:nn { enumext / #1 }
535   {
536     align .choice:,

```

```

537 align / left .code:n = \str_set:cn { l__enumext_align_label#2_str } { l },
538 align / right .code:n = \str_set:cn { l__enumext_align_label#2_str } { r },
539 align / center .code:n = \str_set:cn { l__enumext_align_label#2_str } { c },
540 align / unknown .code:n =
541     \msg_error:nneee { enumext } { unknown-choice }
542     { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
543 align .initial:n = left,
544 align .value_required:n = true,
545 }
546 }
547 \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 `<label>`, 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 *<labels>* of the *four levels* of `enumext` environment, along with the default value for `labelwidth` key and `ref` key.

```

label \__enumext_label_i_tl 548 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
ref   \__enumext_label_ii_tl 549 {
\__enumext_label_iii_tl 550     \keys_define:nn { enumext / #1 }
\__enumext_label_iv_tl 551     {
552         label .code:n = {
553             \__enumext_label_style:cnv { l__enumext_label#2_tl }
554             { l__enumext_counter#2_tl } {##1}
555             \dim_set_eq:cN { l__enumext_labelwidth#2_dim }
556             \__enumext_current_widest_dim
557         },
558         label .initial:n = #3,
559         label .value_required:n = true,
560         ref .code:n = \__enumext_standar_ref:n {##1},
561         ref .value_required:n = true,
562     }
563 }
564 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
565 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
566 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
567 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

(End of definition for label and others.)

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

```

568 \cs_new_protected:Npn \__enumext_standar_ref:n #1
569 {
570     \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
571     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
572     {
573         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
574     }
575     {
576         \tl_set_eq:Nc
577         \__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
578         \__enumext_regex_counter_style:
579         \tl_set_eq:Nc
580         \__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
581         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
582         {
583             \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
584             { \exp_not:V \__enumext_ref_key_arg_tl }
585         }
586     }
587 }

```


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

```

588 \cs_new_protected:Nn \__enumext_standar_ref:
589 {
590   \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
591   {
592     \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
593   }
594 }

```

(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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
595 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
596 {
597   \keys_define:nn { enumext / #1 }
598   {
599     label .code:n = {
600       \__enumext_label_style:cnv { \l__enumext_label_#2_tl }
601       { \__enumext_counter_#2_tl } {##1}
602       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
603       \l__enumext_current_widest_dim
604     },
605     label .initial:n = #3,
606     label .value_required:n = true,
607     ref .code:n = \__enumext_starred_ref:n {##1},
608     ref .value_required:n = true,
609   }
610 }
611 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
612 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

613 \cs_new_protected:Npn \__enumext_starred_ref:n #1
614 {
615   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
616   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
617   {
618     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
619     {
620       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
621     }
622     {
623       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
624       \__enumext_regex_counter_style:
625       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
626       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
627       {
628         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
629         { \exp_not:V \l__enumext_ref_key_arg_tl }
630       }
631     }
632   }
633   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
634   {
635     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
636     {
637       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
638     }
639     {
640       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
641       \__enumext_regex_counter_style:
642       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
643       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
644       {
645         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }

```

```

646             { \exp_not:V \l__enumext_ref_key_arg_tl }
647         }
648     }
649 }
650 }

```

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.

```

651 \cs_new_protected:Nn \__enumext_starred_ref:
652 {
653     \int_compare:nNtT { \l__enumext_level_h_int } = { 1 }
654     {
655         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
656         {
657             \tl_use:N \l__enumext_renew_the_count_vii_tl
658         }
659     }
660     \int_compare:nNtT { \l__enumext_keyans_level_h_int } = { 1 }
661     {
662         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
663         {
664             \tl_use:N \l__enumext_renew_the_count_viii_tl
665         }
666     }
667 }

```

(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.

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
668 \keys_define:nn { enumext / keyans }
669 {
670     label .code:n = {
671         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
672         { \l__enumext_counter_v_tl } {#1}
673         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
674         \l__enumext_current_widest_dim
675         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
676         { \l__enumext_counter_vi_tl } {#1}
677         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
678         \l__enumext_current_widest_dim
679     },
680     label .initial:n = \Alph*,
681     label .value_required:n = true,
682     ref .code:n = \__enumext_keyans_ref:n {#1},
683     ref .value_required:n = true,
684 }

```

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

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

```

685 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
686 {
687     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
688     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
689     {
690         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
691     }
692     {
693         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
694         \__enumext_regex_counter_style:
695         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
696         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
697         {
698             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
699             { \exp_not:V \l__enumext_ref_key_arg_tl }
700         }
701     }
702 }

```

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

```

703 \cs_new_protected:Nn \__enumext_keyans_ref:
704 {
705     \tl_if_empty:NF \__enumext_renew_the_count_v_tl
706     {
707         \tl_use:N \__enumext_renew_the_count_v_tl
708     }
709 }

```

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

11.14 Setting start and widest keys

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.

```

710 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
711 {
712     \__enumext_if_is_int:nTF { #3 }
713     {
714         \int_set:Nn #2 {#3}
715     }
716     {
717         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
718         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
719         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
720         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
721     }
722 }
723 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

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.

```

724 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
725 {
726     \__enumext_if_is_int:nTF {#4}
727     {
728         \setcounter{enumX#1} { #4 }
729     }
730     {
731         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
732         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
733         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
734         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
735     }
736     \__enumext_label_width_by_box:cv
737     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
738 }
739 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

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

```

740 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
741 {
742     \keys_define:nn { enumext / #1 }

```

```

743     {
744         start .code:n = {
745             \__enumext_start_from:ccn
746             { l__enumext_label_#2_tl }
747             { l__enumext_start_#2_int } {##1}
748         },
749         start .initial:n = 1,
750         widest .code:n = {
751             \__enumext_widest_from:nccn {#2}
752             { l__enumext_label_#2_tl }
753             { l__enumext_labelwidth_#2_dim } {##1}
754         },
755         widest .value_required:n = true,
756         start .value_required:n = true,
757     }
758 }
759 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.15 Setting keys for vertical spaces

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

```

760 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
761 {
762     \keys_define:nn { enumext / #1 }
763     {
764         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
765         topsep .initial:n = {#3},
766         topsep .value_required:n = true,
767         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
768         partopsep .initial:n = {#4},
769         partopsep .value_required:n = true,
770         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
771         parsep .initial:n = {#5},
772         parsep .value_required:n = true,
773         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
774         itemsep .initial:n = {#6},
775         itemsep .value_required:n = true,
776         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
777         noitemsep .value_forbidden:n = true,
778         nosepp .meta:n = {
779             itemsep = 0pt, parsep = 0pt,
780             topsep = 0pt, partopsep = 0pt,
781         },
782         nosepp .value_forbidden:n = true,
783     }
784 }

```

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

```

785 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
786 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
787 { 4.0pt plus 2.0pt minus 1.0pt }
788 \__enumext_tmp:nnnnnn { level-2 } { ii } { 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 { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
792 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
793 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
794 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
795 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
796 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
797 { 2.0pt plus 1.0pt minus 1.0pt }
798 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
799 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
800 { 4.0pt plus 2.0pt minus 1.0pt }
801 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
802 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
803 { 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 `\vspace{-\baselineskip}` and set `topsep=0pt` for the “first level” of the nested `enumext` or `enumext*` environments.

```
base-fix
\__enumext_nested_base_line_fix:
804 \cs_set_protected:Npn \__enumext_tmp:n #1
805 {
806   \keys_define:nn { enumext / #1 }
807   {
808     base-fix .bool_set:N = \__enumext_base_line_fix_bool,
809     base-fix .initial:n = false,
810     base-fix .value_forbidden:n = true,
811   }
812 }
813 \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 (§11.43)

```
814 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
815 {
816   \bool_lazy_and:nnT
817   { \bool_if_p:N \__enumext_standar_first_bool }
818   { \bool_if_p:N \__enumext_base_line_fix_bool }
819   {
820     \mode_leave_vertical:
821     \vspace { -\baselineskip }
822     \keys_set:nn { enumext / level-1 }
823     {
824       topsep = 0pt, above = 0pt, above* = 0pt,
825     }
826   }
827   \bool_lazy_and:nnT
828   { \bool_if_p:N \__enumext_starred_first_bool }
829   { \bool_if_p:N \__enumext_base_line_fix_bool }
830   {
831     \mode_leave_vertical:
832     \vspace { -\baselineskip }
833     \keys_set:nn { enumext / enumext* }
834     {
835       topsep = 0pt, above = 0pt, above* = 0pt,
836     }
837   }
838   \bool_set_false:N \__enumext_base_line_fix_bool
839 }
```

🔑 This key is enabled by default in the command `\printkeyans` (§11.46).

(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.

```
840 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
841 {
842   \keys_define:nn { enumext / #1 }
843   {
844     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
845     itemindent .value_required:n = true,
846     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
847     rightmargin .value_required:n = true,
848     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
849     listparindent .value_required:n = true,
850     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
851     list-offset .value_required:n = true,
852     list-indent .code:n =
853       \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
854       \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {#1},
```

```

855         list-indent .value_required:n = true,
856     }
857 }
858 \clist_map_inline:nn
859 {
860     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
861 }
862 { \__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.

```

863 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
864 {
865     \keys_define:nn { enumext / #1 }
866     {
867         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
868         itemindent .value_required:n = true,
869         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
870         rightmargin .value_required:n = true,
871         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
872         listparindent .value_required:n = true,
873         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
874         list-offset .value_required:n = true,
875         list-indent .meta:n = { list-offset = ##1 },
876         list-indent .value_required:n = true,
877     }
878 }
879 \clist_map_inline:nn
880 {
881     {enumext*}{vii}, {keyans*}{viii}
882 }
883 { \__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.

```

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

```



```

915     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
916     {
917         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
918         {
919             \exp_not:N \mode_leave_vertical:
920             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
921         }
922     }
923 }
924 \cs_set_protected:Nn \__enumext_fake_item_viii:
925 {
926     \dim_compare:nNt
927     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
928     {
929         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
930         {
931             \exp_not:N \mode_leave_vertical:
932             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
933         }
934     }
935 }

```

(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.

```

936 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
937 {
938     \keys_define:nn { enumext / #1 }
939     {
940         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
941         show-length .initial:n = false,
942     }
943 }
944 \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.

```

945 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
946 {
947     \keys_define:nn { enumext / #1 }
948     {
949         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
950         before .value_required:n = true,
951         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
952         before* .value_required:n = true,
953         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
954         after .value_required:n = true,
955         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
956         first .value_required:n = true,
957     }
958 }
959 \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 `{\arg two}` of the list: `{\code}\list{\arg one}{\arg two}`.

```

960 \cs_new_protected:Nn \__enumext_before_args_exec:
961 {
962     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
963 }

```

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

```

964 \cs_new_protected:Nn \__enumext_before_keys_exec:
965 {
966   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
967 }

```

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

```

968 \cs_new_protected:Nn \__enumext_after_stop_list:
969 {
970   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
971 }

```

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

```

972 \cs_new_protected:Nn \__enumext_after_args_exec:
973 {
974   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
975 }

```

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

11.19.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
976 \cs_new_protected:Nn \__enumext_before_args_exec_v:
977 {
978   \tl_use:N \l__enumext_before_starred_key_v_tl
979 }
980 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
981 {
982   \tl_use:N \l__enumext_before_no_starred_key_v_tl
983 }
984 \cs_new_protected:Nn \__enumext_after_stop_list_v:
985 {
986   \tl_use:N \l__enumext_after_stop_list_v_tl
987 }
988 \cs_new_protected:Nn \__enumext_after_args_exec_v:
989 {
990   \tl_use:N \l__enumext_after_list_args_v_tl
991 }

```

(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*

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

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii:
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
992 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
993 {
994   \tl_use:N \l__enumext_before_starred_key_vii_tl
995 }
996 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
997 {
998   \tl_use:N \l__enumext_before_starred_key_viii_tl
999 }
1000 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1001 {
1002   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1003 }
1004 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1005 {
1006   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1007 }
1008 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1009 {
1010   \tl_use:N \l__enumext_after_stop_list_vii_tl
1011 }
1012 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1013 {
1014   \tl_use:N \l__enumext_after_stop_list_viii_tl

```

```

1015 }
1016 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1017 {
1018   \tl_use:N \l__enumext_after_list_args_vii_tl
1019 }
1020 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1021 {
1022   \tl_use:N \l__enumext_after_list_args_viii_tl
1023 }

```

(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.

```

1024 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1025 {
1026   \keys_define:nn { enumext / #1 }
1027   {
1028     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
1029     mini-env .value_required:n = true,
1030     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
1031     mini-sep .initial:n = 0.3333em,
1032     mini-sep .value_required:n = true,
1033     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
1034     columns-sep .value_required:n = true,
1035     columns .int_set:c = { \l__enumext_columns_#2_int },
1036     columns .initial:n = 1,
1037     columns .value_required:n = true,
1038   }
1039 }
1040 \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.

```

1041 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1042 {
1043   \keys_define:nn { enumext / #1 }
1044   {
1045     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1046     mini-right .value_required:n = true,
1047     mini-right* .code:n = {
1048       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1049       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1050     },
1051     mini-right* .value_required:n = true,
1052   }
1053 }
1054 \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 multicols

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



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

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

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

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 `multicols` environment in `enumext`.

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

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

```

1055 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1056 {
1057   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: } _skip {
1058     {
1059       \skip_use:c { \l__enumext_topsep_ \__enumext_level: } _skip {
1060       }
1061     }
1062   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: } _skip {
1063     {
1064       \skip_use:c { \l__enumext_topsep_ \__enumext_level: } _skip {
1065       }
1066     }
1067   }
1068 }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

1067 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1068 {
1069   \int_case:nn { \l__enumext_level_int }
1070   {
1071     { 2 }{
1072       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip } {
1073         {
1074           \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1075         }
1076       }
1077     { 3 }{
1078       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip } {
1079         {
1080           \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1081         }
1082       }
1083     { 4 }{
1084       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip } {
1085         {
1086           \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1087         }
1088       }
1089     }
1090   }
1091 }

```

(End of definition for `__enumext_add_pre_parsep:`)

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

```

1091 \cs_new_protected:Nn \__enumext_multi_addvspace:
1092 {
1093   \__enumext_multi_set_vskip:
1094   \mode_if_vertical:T
1095   {
1096     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: } _skip {
1097       {
1098         \skip_use:c { \l__enumext_partopsep_ \__enumext_level: } _skip {

```

```

1099     }
1100     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: \_skip }
1101     {
1102         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: \_skip }
1103     }
1104 }
1105 \par\nopagebreak
1106 \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: \_skip } }
1107 }

```

(End of definition for `\l__enumext_multi_addvspace:`)

11.21.2 Adjustment of vertical spaces for multicols in keyans

`\l__enumext_keyans_multi_set_vskip:`
`\l__enumext_keyans_multi_addvspace:`

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

```

1108 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
1109 {
1110     \skip_set:Nn \l__enumext_multicols_above_v_skip
1111     {
1112         \l__enumext_topsep_v_skip
1113     }
1114     \skip_set:Nn \l__enumext_multicols_below_v_skip
1115     {
1116         \l__enumext_topsep_v_skip
1117     }
1118 }
1119 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
1120 {
1121     \l__enumext_keyans_multi_set_vskip:
1122     \mode_if_vertical:T
1123     {
1124         \skip_add:Nn \l__enumext_multicols_above_v_skip
1125         {
1126             \skip_use:N \l__enumext_partopsep_v_skip
1127         }
1128         \skip_add:Nn \l__enumext_multicols_below_v_skip
1129         {
1130             \skip_use:N \l__enumext_partopsep_v_skip
1131         }
1132     }
1133     \par\nopagebreak
1134     \addvspace{ \l__enumext_multicols_above_v_skip }
1135 }

```

(End of definition for `\l__enumext_keyans_multi_set_vskip:` and `\l__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.

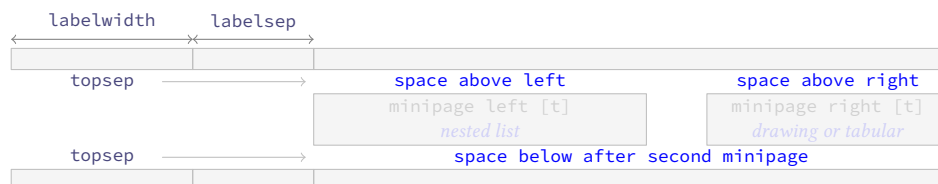


Figure 8: Representation of the `\minipage` spacing adjustment for a nested level.

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

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

- Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (glue) produced by the `\minipage` environment is quite complicated, even more if `\multicols` 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 *(horizontal mode)*, then we will make the settings for the *(vertical mode)* 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`.

```

1136 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1137 {
1138   \int_compare:nNnTF
1139     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1140     {

```

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.

```

1141   \skip_if_eq:nnTF
1142     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1143   {
1144     \skip_set:Nn \l__enumext_minipage_left_skip
1145       {
1146         -0.150\box_dp:N \strutbox
1147       }
1148     \skip_set:Nn \l__enumext_minipage_right_skip
1149       {
1150         0.695\box_dp:N \strutbox
1151       }
1152     \skip_set:Nn \l__enumext_minipage_after_skip
1153       {
1154         \box_dp:N \strutbox
1155       }
1156     \__enumext_zero_parsep:
1157   }
1158   {
1159     \skip_set:Nn \l__enumext_minipage_left_skip
1160       {
1161         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1162       }
1163     \skip_set:Nn \l__enumext_minipage_right_skip
1164       {
1165         0.695\box_dp:N \strutbox
1166       }
1167     \skip_set:Nn \l__enumext_minipage_after_skip
1168       {
1169         1.85\box_dp:N \strutbox
1170         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1171       }
1172   }
1173 }
1174 {

```

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.

```

1175   \skip_if_eq:nnTF
1176     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1177   {
1178     \skip_set:Nn \l__enumext_minipage_left_skip
1179       {
1180         0.5\box_dp:N \strutbox
1181         - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1182       }
1183     \skip_set:Nn \l__enumext_minipage_right_skip
1184       {
1185         \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1186       }
1187     \skip_set:Nn \l__enumext_minipage_after_skip

```



```

1188         {
1189             1.6\box_dp:N \strutbox
1190         }
1191     }
1192     {
1193         \skip_set:Nn \l__enumext_minipage_left_skip
1194         {
1195             0.5875\box_dp:N \strutbox
1196             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1197         }
1198         \skip_set:Nn \l__enumext_minipage_right_skip
1199         {
1200             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1201             + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1202         }
1203         \skip_set:Nn \l__enumext_minipage_after_skip
1204         {
1205             0.325\box_dp:N \strutbox
1206             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1207         }
1208     }
1209 }
1210 }

```

(End of definition for `__enumext_mini_set_vskip:`.)

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

```

1211 \cs_new_protected:Nn \__enumext_zero_parsep:
1212 {
1213     \int_case:nn { \l__enumext_level_int }
1214     {
1215         { 2 }{
1216             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1217             {
1218                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1219             }
1220         }
1221         { 3 }{
1222             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1223             {
1224                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1225             }
1226         }
1227         { 4 }{
1228             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1229             {
1230                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1231             }
1232         }
1233     }
1234 }

```

(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*.

```

1235 \cs_new_protected:Nn \__enumext_mini_addvspace:
1236 {
1237     \__enumext_mini_set_vskip:
1238     \mode_if_vertical:T
1239     {
1240         \skip_add:Nn \l__enumext_minipage_left_skip
1241         {
1242             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1243         }
1244         \skip_add:Nn \l__enumext_minipage_after_skip
1245         {

```

```

1246         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: \skip }
1247     }
1248 }
1249 \par\nopagebreak
1250 \addvspace { \l__enumext_minipage_left_skip }
1251 }

```

(End of definition for `\l__enumext_mini_addvspace:`.)

11.22.2 Adjustment of vertical spaces for minipage in keyans

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

```

1252 \cs_new_protected:Nn \l__enumext_keyans_mini_set_vskip:
1253 {
1254     \skip_zero_new:N \l__enumext_minipage_after_skip
1255     \skip_zero_new:N \l__enumext_minipage_left_skip
1256     \skip_zero_new:N \l__enumext_minipage_right_skip
1257     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1258     {
1259         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1260         {
1261             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1262             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1263             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1264             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1265             {
1266                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1267             }
1268         }
1269         {
1270             \skip_set:Nn \l__enumext_minipage_left_skip
1271             {
1272                 \skip_use:N \l__enumext_topsep_v_skip
1273             }
1274             \skip_set:Nn \l__enumext_minipage_right_skip
1275             {
1276                 0.705\box_dp:N \strutbox
1277             }
1278             \skip_set:Nn \l__enumext_minipage_after_skip
1279             {
1280                 1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1281             }
1282         }
1283     }
1284     {
1285         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1286         {
1287             \skip_set:Nn \l__enumext_minipage_left_skip
1288             {
1289                 0.5\box_dp:N \strutbox
1290                 + \l__enumext_partopsep_v_skip
1291             }
1292             \skip_set:Nn \l__enumext_minipage_right_skip
1293             {
1294                 \l__enumext_partopsep_v_skip
1295             }
1296             \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1297         }
1298         {
1299             \skip_set:Nn \l__enumext_minipage_left_skip
1300             {
1301                 0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1302             }
1303             \skip_set:Nn \l__enumext_minipage_right_skip
1304             {
1305                 \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1306             }
1307             \skip_set:Nn \l__enumext_minipage_after_skip
1308             {
1309                 0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip

```

```

1310         }
1311     }
1312 }
1313 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

The function `__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `__enumext_mini_env*` environment in `keyans`, taking into account whether \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`.

```

1314 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1315 {
1316     \__enumext_keyans_mini_set_vskip:
1317     \mode_if_vertical:T
1318     {
1319         \skip_add:Nn \l__enumext_minipage_left_skip
1320         {
1321             \l__enumext_partopsep_v_skip
1322         }
1323         \skip_add:Nn \l__enumext_minipage_after_skip
1324         {
1325             \l__enumext_partopsep_v_skip
1326         }
1327     }
1328     \par\nopagebreak
1329     \addvspace { \l__enumext_minipage_left_skip }
1330 }

```

(End of definition for `__enumext_keyans_mini_addvspace:`)

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

`__enumext_mini_set_vskip_vii:`

`__enumext_mini_set_vskip_viii:`

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

```

1331 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1332 {
1333     \skip_zero_new:N \l__enumext_minipage_left_skip
1334     \skip_gzero_new:N \g__enumext_minipage_right_skip
1335     \skip_gzero_new:N \g__enumext_minipage_after_skip
1336     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1337     {
1338         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1339         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1340     }
1341     {
1342         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1343         \skip_gset:Nn \g__enumext_minipage_right_skip
1344         {
1345             \l__enumext_topsep_vii_skip
1346         }
1347         \skip_gset:Nn \g__enumext_minipage_after_skip
1348         {
1349             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1350         }
1351     }
1352 }
1353 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1354 {
1355     \skip_zero_new:N \l__enumext_minipage_after_skip
1356     \skip_zero_new:N \l__enumext_minipage_left_skip
1357     \skip_zero_new:N \l__enumext_minipage_right_skip
1358     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1359     {
1360         \skip_set:Nn \l__enumext_minipage_left_skip
1361         {
1362             0.5\box_dp:N \strutbox
1363         }
1364         \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1365     {
1366         \l__enumext_partopsep_viii_skip
1367     }
1368     \skip_set:Nn \l__enumext_minipage_after_skip
1369     {
1370         1.6\box_dp:N \strutbox
1371     }
1372 }
1373 {
1374     \skip_set:Nn \l__enumext_minipage_left_skip
1375     {
1376         0.5875\box_dp:N \strutbox
1377     }
1378     \skip_set:Nn \l__enumext_minipage_right_skip
1379     {
1380         \l__enumext_topsep_viii_skip
1381     }
1382     \skip_set:Nn \l__enumext_minipage_after_skip
1383     {
1384         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1385     }
1386 }
1387 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext_mini_env*` environment on the *left side* when the `mini-right` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether T_EX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to 0pt in both environments.

```

1388 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1389 {
1390     \__enumext_mini_set_vskip_vii:
1391     \par\nopagebreak
1392     \addvspace { \l__enumext_minipage_left_skip }
1393 }
1394 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1395 {
1396     \__enumext_mini_set_vskip_viii:
1397     \par\nopagebreak
1398     \addvspace { \l__enumext_minipage_left_skip }
1399 }

```

(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 L^AT_EX justification is maintained in the `__enumext_mini_env*` on the “right side”.

`\miniright`

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

```

1400 \NewDocumentCommand \miniright { s }
1401 {
1402     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1403     {
1404         \msg_error:nnn { enumext } { wrong-miniright-place }
1405     }
1406     % outside
1407     \bool_lazy_and:nnT
1408     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1409     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1410     {
1411         \msg_error:nnn { enumext } { wrong-miniright-place }
1412     }
1413     % starred env

```

```

1414 \bool_if:NT \l__enumext_starred_bool
1415 {
1416   \msg_error:nnn { enumext } { wrong-miniright-starred }
1417 }
1418 \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1419 {
1420   \__enumext_keyans_mini_right_cmd:n {#1}
1421 }
1422 { \__enumext_mini_right_cmd:n {#1} }
1423 }

```

(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.

```

1424 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1425 {
1426   \dim_compare:nNnTF
1427   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1428   {
1429     \__enumext_multicols_stop:
1430     \end{__enumext_mini_env*}
1431     \hfill
1432     \begin{__enumext_mini_env*}
1433     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1434     \par\addvspace { \l__enumext_minipage_right_skip }
1435     \bool_if:nF {#1}
1436     {
1437       \centering
1438     }
1439     \int_gzero:N \g__enumext_minipage_stat_int
1440   }
1441   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1442 }

```

(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.

```

1443 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1444 {
1445   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1446   {
1447     \__enumext_keyans_multicols_stop:
1448     \end{__enumext_mini_env*}
1449     \hfill
1450     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1451     \par\addvspace { \l__enumext_minipage_right_skip }
1452     \bool_if:nF {#1}
1453     {
1454       \centering
1455     }
1456     \int_gzero:N \g__enumext_minipage_stat_int
1457   }
1458   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1459 }

```

(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.

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

```

above* 1460 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1461 {
below* 1462   \keys_define:nn { enumext / #1 }
1463   {
1464     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1465     above .value_required:n = true,
1466     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1467               \keys_set:nn { enumext / #1 } { above = {##1} },
1468     above* .value_required:n = true,
1469     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1470     below .value_required:n = true,
1471     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1472               \keys_set:nn { enumext / #1 } { below = {##1} },
1473     below* .value_required:n = true,
1474   }
1475 }
1476 \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.

```

1477 \cs_new_protected:Nn \__enumext_vspace_above:
1478 {
1479   \skip_if_eq:nnF
1480   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1481   {
1482     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1483     {
1484       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1485     }
1486     {
1487       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1488     }
1489   }
1490 }

```

(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.

```

1491 \cs_new_protected:Nn \__enumext_vspace_below:
1492 {
1493   \skip_if_eq:nnF
1494   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1495   {
1496     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1497     {
1498       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1499     }
1500     {
1501       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1502     }
1503   }
1504 }

```

(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.

```

1505 \cs_new_protected:Nn \__enumext_vspace_above_v:
1506 {
1507   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1508   {
1509     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1510     {
1511       \vspace*{ \l__enumext_vspace_above_v_skip }

```



```

1512     }
1513     { \vspace { \l__enumext_vspace_above_v_skip } }
1514 }
1515 }

```

(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.

```

1516 \cs_new_protected:Nn \__enumext_vspace_below_v:
1517 {
1518     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1519     {
1520         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1521         {
1522             \vspace*{ \l__enumext_vspace_below_v_skip }
1523         }
1524         { \vspace { \l__enumext_vspace_below_v_skip } }
1525     }
1526 }

```

(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:`

```

1527 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1528 {
1529     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1530     {
1531         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1532         {
1533             \vspace*{ \l__enumext_vspace_above_vii_skip }
1534         }
1535         { \vspace { \l__enumext_vspace_above_vii_skip } }
1536     }
1537 }
1538 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1539 {
1540     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1541     {
1542         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1543         {
1544             \vspace*{ \l__enumext_vspace_above_viii_skip }
1545         }
1546         { \vspace { \l__enumext_vspace_above_viii_skip } }
1547     }
1548 }

```

(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:`

```

1549 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1550 {
1551     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1552     {
1553         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1554         {
1555             \vspace*{ \l__enumext_vspace_below_vii_skip }
1556         }
1557         { \vspace { \l__enumext_vspace_below_vii_skip } }
1558     }
1559 }
1560 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1561 {
1562     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1563     {
1564         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1565         {
1566             \vspace*{ \l__enumext_vspace_below_viii_skip }

```

```

1567     }
1568     { \vspace { \l__enumext_vspace_below_viii_skip } }
1569   }
1570 }

```

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

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1571 \cs_set_protected:Npn \__enumext_tmp:n #1
1572 {
1573   \keys_define:nn { enumext / #1 }
1574   {
1575     series .str_set:N = \l__enumext_series_str,
1576     series .value_required:n = true,
1577     resume .code:n = \__enumext_resume_series:n {##1},
1578     resume* .code:n = \__enumext_resume_starred:,
1579     resume* .value_forbidden:n = true,
1580   }
1581 }
1582 \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

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

```

1583 \cs_new:Npn \__enumext_filter_series:n #1
1584 {
1585   \use:e
1586   {
1587     \keyval_parse:NNn
1588       \__enumext_filter_series_key:n
1589       \__enumext_filter_series_pair:nn {#1}
1590   }
1591 }

```

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

```

1592 \cs_new:Npn \__enumext_filter_series_key:n #1
1593 {
1594   \str_case:nnF {#1}
1595   {
1596     { resume } {} { resume* } {} { base-fix } {}
1597   }
1598   { , { \exp_not:n {#1} } }
1599 }

```

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

```

1600 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1601 {
1602   \str_case:nnF {#1}
1603   {
1604     { series } {} { resume } {} { start } {} { save-ans } {} { save-key } {}
1605   }
1606   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1607 }

```

(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 $\langle keys \rangle$ 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 (§11.43).

```

1608 \cs_new_protected:Npn \__enumext_parse_series:n #1
1609 {
1610   \str_if_empty:NTF \l__enumext_series_str
1611   {
1612     \bool_if:NF \l__enumext_resume_active_bool
1613     {
1614       \__enumext_resume_last:n {#1}
1615     }
1616   }
1617   {
1618     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1619     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1620     { \__enumext_filter_series:n {#1} }
1621     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1622     {
1623       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1624     }
1625   }
1626 }

```

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.

```

1627 \cs_new_protected:Npn \__enumext_resume_last:n #1
1628 {
1629   \bool_if:NT \l__enumext_standar_first_bool
1630   {
1631     \tl_gclear:N \g__enumext_standar_series_tl
1632     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1633   }
1634   \bool_if:NT \l__enumext_starred_first_bool
1635   {
1636     \tl_gclear:N \g__enumext_starred_series_tl
1637     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1638   }
1639 }

```

(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 `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{<series name>}` but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={<series name>}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={<series name>}`. This function is passed to the `enumext` environment definition (§11.38) and the `enumext*` environment definition (§11.43).

```

1640 \cs_new_protected:Nn \__enumext_resume_save_counter:
1641 {
1642   \bool_if:NT \g__enumext_standar_bool
1643   {
1644     \tl_if_empty:NF \l__enumext_series_str
1645     {
1646       \int_gset_eq:cN
1647       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1648     }
1649     \tl_if_empty:NTF \l__enumext_resume_name_tl
1650     {
1651       \str_if_empty:NT \l__enumext_series_str

```

```

1652         {
1653             \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1654         }
1655     }
1656     {
1657         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1658         {
1659             \int_gset_eq:cN
1660             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1661         }
1662     }
1663     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1664     {
1665         \int_gset_eq:cN
1666         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1667     }
1668 }
1669 \bool_if:NT \g__enumext_starred_bool
1670 {
1671     \tl_if_empty:NF \l__enumext_series_str
1672     {
1673         \int_gset_eq:cN
1674         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1675     }
1676     \tl_if_empty:NTF \l__enumext_resume_name_tl
1677     {
1678         \str_if_empty:NT \l__enumext_series_str
1679         {
1680             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1681         }
1682     }
1683     {
1684         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1685         {
1686             \int_gset_eq:cN
1687             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1688         }
1689     }
1690     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1691     {
1692         \int_gset_eq:cN
1693         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1694     }
1695 }
1696 }

```

(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.

```

1697 \cs_new_protected:Npn \__enumext_resume_series:n #1
1698 {
1699     \tl_if_empty:nTF {#1}
1700     {
1701         \__enumext_resume_counter:n { }
1702     }
1703     {
1704         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1705         {
1706             \__enumext_resume_counter:n {#1}
1707             \bool_if:NT \g__enumext_standar_bool
1708             {
1709                 \keys_set:nv { enumext / level-1 }
1710                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1711             }

```

```

1712         \bool_if:NT \g__enumext_starred_bool
1713         {
1714             \keys_set:nv { enumext / enumext* }
1715             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1716         }
1717     }
1718     {
1719         \bool_if:NT \g__enumext_standar_bool
1720         {
1721             \msg_error:nnn { enumext } { unknown-series } {#1}
1722         }
1723         \bool_if:NT \g__enumext_starred_bool
1724         {
1725             \msg_error:nnn { enumext } { unknown-series } {#1}
1726         }
1727     }
1728 }
1729 }

```

(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`.

```

1730 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1731 {
1732     \bool_set_true:N \l__enumext_resume_active_bool
1733     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1734     \tl_if_empty:NTF \l__enumext_resume_name_tl
1735     {
1736         \__enumext_resume_counter:
1737     }
1738     {
1739         \__enumext_resume_counter_series:
1740     }
1741     \__enumext_resume_counter_save_ans:
1742 }

```

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.

```

1743 \cs_new_protected:Nn \__enumext_resume_counter:
1744 {
1745     \bool_if:NT \g__enumext_standar_bool
1746     {
1747         \int_gincr:N \g__enumext_resume_int
1748         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1749     }
1750     \bool_if:NT \g__enumext_starred_bool
1751     {
1752         \int_gincr:N \g__enumext_resume_vii_int
1753         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1754     }
1755 }

```

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.

```

1756 \cs_new_protected:Nn \__enumext_resume_counter_series:
1757 {
1758     \bool_if:NT \g__enumext_standar_bool
1759     {
1760         \int_set:Nn \l__enumext_start_i_int
1761         {
1762             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1763         }
1764     }
1765     \bool_if:NT \g__enumext_starred_bool

```

```

1766     {
1767         \int_set:Nn \l__enumext_start_vii_int
1768         {
1769             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1770         }
1771     }
1772 }

```

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.

```

1773 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1774 {
1775     \bool_lazy_and:nnT
1776     { \bool_if_p:N \l__enumext_standar_first_bool }
1777     { \bool_if_p:N \l__enumext_store_active_bool }
1778     {
1779         \int_set:Nn \l__enumext_start_i_int
1780         {
1781             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1782         }
1783     }
1784     \bool_lazy_and:nnT
1785     { \bool_if_p:N \l__enumext_starred_first_bool }
1786     { \bool_if_p:N \l__enumext_store_active_bool }
1787     {
1788         \int_set:Nn \l__enumext_start_vii_int
1789         {
1790             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1791         }
1792     }
1793 }

```

(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.

```

1794 \cs_new_protected:Nn \__enumext_resume_starred:
1795 {
1796     \bool_if:NT \g__enumext_standar_bool
1797     {
1798         \tl_if_empty:NF \g__enumext_standar_series_tl
1799         {
1800             \__enumext_resume_counter:n { }
1801             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1802         }
1803     }
1804     \bool_if:NT \g__enumext_starred_bool
1805     {
1806         \tl_if_empty:NF \g__enumext_starred_series_tl
1807         {
1808             \__enumext_resume_counter:n { }
1809             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1810         }
1811     }
1812 }

```

(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*`.

```

1813 \cs_set_protected:Npn \__enumext_tmp:n #1
1814 {
1815     \keys_define:nn { enumext / #1 }

```

```

1816     {
1817         save-ans .code:n = \__enumext_storing_set:n {##1},
1818         save-ans .value_required:n = true,
1819     }
1820 }
1821 \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

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

```

1822 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1823 {
1824     \msg_term:nnVV { enumext } { save-ans-log }
1825     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1826 }
1827 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1828 {
1829     \msg_term:nnVV { enumext } { save-ans-log-hook }
1830     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1831 }

```

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

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.

```

1832 \cs_new_protected:Npn \__enumext_storing_set:n #1
1833 {
1834     \tl_set:Nx \l__enumext_store_name_tl {#1}
1835     \tl_if_empty:NTF \l__enumext_store_name_tl
1836     {
1837         \bool_lazy_or:nnT
1838         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1839         {
1840             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1841         }
1842     }
1843     {
1844         \bool_lazy_or:nnT
1845         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1846         {
1847             \__enumext_start_save_ans_msg:
1848             \__enumext_storing_exec:
1849         }
1850     }
1851 }

```

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*`.

```

1852 \cs_new_protected:Nn \__enumext_storing_exec:
1853 {
1854     \bool_set_true:N \l__enumext_store_active_bool
1855     \bool_set_true:N \l__enumext_check_answers_bool
1856     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1857     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1858     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }

```



```

1859     {
1860         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1861         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1862     }
1863     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1864     {
1865         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1866         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1867     }
1868     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1869     {
1870         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1871         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1872     }
1873 }

```

(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 no-store 1874 \cs_set_protected:Npn \__enumext_tmp:n #1
1875 {
1876     \keys_define:nn { enumext / #1 }
1877     {
1878         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1879         check-ans .initial:n = false,
1880         check-ans .value_required:n = true,
1881         no-store .code:n = {
1882             \bool_set_false:N \l__enumext_check_answers_bool
1883             \bool_set_false:N \l__enumext_check_ans_key_bool
1884         },
1885         no-store .value_forbidden:n = true,
1886     }
1887 }
1888 \clist_map_inline:nn
1889 {
1890     level-1, level-2, level-3, level-4, enumext*
1891 }
1892 { \__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 `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `__enumext_check_ans_level`: only if “*true*”, i.e. the key `no-store` is not active.

```

1893 \cs_new_protected:Nn \__enumext_check_ans_active:
1894 {
1895   \tl_if_empty:NF \l__enumext_store_name_tl
1896   {
1897     \bool_if:NT \l__enumext_check_answers_bool
1898     {
1899       \__enumext_check_ans_level:
1900     }
1901   }
1902 }

```

The function `__enumext_check_ans_level`: will decrement by “*one*” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment `enumext`, taking into account whether it is nested within `enumext*` or the opposite and set `\l__enumext_item_number_bool` to “*false*”.

```

1903 \cs_new_protected:Nn \__enumext_check_ans_level:
1904 {
1905   \int_case:nn { \l__enumext_level_int }
1906   {
1907     { 1 }{
1908       \bool_lazy_all:nT
1909       {
1910         { \bool_if_p:N \g__enumext_starred_bool }
1911         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1912       }
1913       {
1914         \int_gdecr:N \g__enumext_item_number_int
1915         \bool_set_false:N \l__enumext_item_number_bool
1916       }
1917     }
1918     { 2 }{
1919       \int_gdecr:N \g__enumext_item_number_int
1920       \bool_set_false:N \l__enumext_item_number_bool
1921     }
1922     { 3 }{
1923       \int_gdecr:N \g__enumext_item_number_int
1924       \bool_set_false:N \l__enumext_item_number_bool
1925     }
1926     { 4 }{
1927       \int_gdecr:N \g__enumext_item_number_int
1928       \bool_set_false:N \l__enumext_item_number_bool
1929     }
1930   }

```

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.

```

1931   \int_case:nn { \l__enumext_level_h_int }
1932   {
1933     { 1 }{
1934       \bool_lazy_all:nT
1935       {
1936         { \bool_if_p:N \g__enumext_standar_bool }
1937         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1938       }
1939       {
1940         \int_gdecr:N \g__enumext_item_number_int
1941         \bool_set_false:N \l__enumext_item_number_bool
1942       }
1943     }
1944   }
1945 }

```

(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.

```

1946 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1947 {
1948   \bool_lazy_and:nnT
1949     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1950     { \bool_if_p:N \g__enumext_standar_bool }
1951   {
1952     \bool_gset_true:N \g__enumext_check_ans_key_bool
1953   }
1954   \bool_lazy_and:nnT
1955     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1956     { \bool_if_p:N \g__enumext_starred_bool }
1957   {
1958     \bool_gset_true:N \g__enumext_check_ans_key_bool
1959   }
1960 }

```

(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:.

```

1961 \cs_new_protected:Nn \__enumext_item_answer_diff:
1962 {
1963   \int_gset:Nn \g__enumext_item_answer_diff_int
1964   {
1965     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1966   }
1967 }

```

(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:.

```

1968 \cs_new_protected:Nn \__enumext_check_ans_show:
1969 {
1970   \int_case:nn { \g__enumext_item_answer_diff_int }
1971   {
1972     { -1 } { \__enumext_check_ans_msg_less: }
1973     { 0 } { \__enumext_check_ans_msg_same_ok: }
1974     { 1 } { \__enumext_check_ans_msg_greater: }
1975   }
1976 }
1977 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1978 {
1979   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1980   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1981 }
1982 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1983 {
1984   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1985   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1986 }
1987 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1988 {
1989   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1990   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1991 }

```

(End of definition for __enumext_check_ans_show: and others.)

__enumext_check_ans_log:

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

__enumext_check_ans_log_msg_less:

__enumext_check_ans_log_msg_same_ok:

__enumext_check_ans_log_msg_greater:

```

1992 \cs_new_protected:Nn \__enumext_check_ans_log:
1993 {
1994   \int_case:nn { \g__enumext_item_answer_diff_int }
1995   {
1996     { -1 } { \__enumext_check_ans_log_msg_less: }
1997     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1998     { 1 } { \__enumext_check_ans_log_msg_greater: }
1999   }
2000 }
2001 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2002 {
2003   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2004   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2005 }
2006 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2007 {
2008   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2009   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2010 }
2011 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2012 {
2013   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2014   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2015 }

```

(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.

```

2016 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2017 {
2018   \int_compare:nNnT
2019   { \g__enumext_check_starred_cmd_int } = { 0 }
2020   {
2021     \msg_warning:nnnV
2022     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2023   }
2024   \int_compare:nNnT
2025   { \g__enumext_check_starred_cmd_int } > { 1 }
2026   {
2027     \msg_warning:nnnV
2028     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2029   }
2030   \int_gzero:N \g__enumext_check_starred_cmd_int
2031   \tl_clear:N \l__enumext_check_start_line_env_tl
2032 }

```

(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*`.

```

2033 \cs_set_protected:Npn \__enumext_tmp:n #1
2034 {
2035   \keys_define:nn { enumext / #1 }
2036   {
2037     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2038     wrap-ans .initial:n =
2039       {
2040         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2041       },
2042     wrap-ans .value_required:n = true,
2043     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2044     wrap-opt .initial:n = [{##1}],
2045     wrap-opt .value_required:n = true,
2046     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2047     save-sep .initial:n = {, ~ },

```

```

2048     save-sep      .value_required:n = true,
2049     mark-ans      .tl_set:N = \l__enumext_mark_answer_sym_tl,
2050     mark-ans      .initial:n = \textasteriskcentered,
2051     mark-ans      .value_required:n = true,
2052     mark-pos      .choice:,
2053     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2054     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2055     mark-pos / unknown .code:n =
2056         \msg_error:nnee { enumext } { unknown-choice }
2057         { mark-pos } { left, ~ right } { \exp_not:n {#1} },
2058     mark-pos      .initial:n = right,
2059     mark-pos      .value_required:n = true,
2060     show-ans      .bool_set:N = \l__enumext_show_answer_bool,
2061     show-ans      .initial:n = false,
2062     show-ans      .value_required:n = true,
2063     show-pos      .bool_set:N = \l__enumext_show_position_bool,
2064     show-pos      .initial:n = false,
2065     show-pos      .value_required:n = true,
2066     mark-ref      .tl_set:N = \l__enumext_mark_ref_sym_tl,
2067     mark-ref      .initial:n = \textasteriskcentered,
2068     mark-ref      .value_required:n = true,
2069     save-ref      .bool_set:N = \l__enumext_store_ref_key_bool,
2070     save-ref      .initial:n = false,
2071     save-ref      .value_required:n = true,
2072   }
2073 }
2074 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {#1} }

```

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

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

```

2075 \cs_set_protected:Npn \l__enumext_tmp:n #1
2076 {
2077   \keys_define:nn { enumext / #1 }
2078   {
2079     mark-pos .choice:,
2080     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2081     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2082     mark-pos .initial:n = right,
2083     mark-pos .value_required:n = true,
2084     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2085     show-ans .initial:n = false,
2086     show-ans .value_required:n = true,
2087     show-pos .bool_set:N = \l__enumext_show_position_bool,
2088     show-pos .initial:n = false,
2089     show-pos .value_required:n = true,
2090   }
2091 }
2092 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

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

11.26.1 Store optional arguments of the environments

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

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

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

```

2093 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2094 {
2095   \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2096   {
2097     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2098     \tl_set:ce

```

```

2099         { \__enumext_store_save_key_ \__enumext_level: _tl }
2100         { \__enumext_filter_save_key:n {#1} }
2101     }
2102 }
2103 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2104 {
2105     \bool_if:NF \__enumext_store_save_key_vii_bool
2106     {
2107         \tl_clear:N \__enumext_store_save_key_vii_tl
2108         \tl_set:Nx \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2109     }
2110 }

```

(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.

```

2111 \cs_set_protected:Npn \__enumext_tmp:n #1
2112 {
2113     \keys_define:nn { enumext / enumext* }
2114     {
2115         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2116         save-key .value_required:n = true,
2117     }
2118     \keys_define:nn { enumext / #1 }
2119     {
2120         save-key .code:n = \__enumext_parse_save_key:n {##1},
2121         save-key .value_required:n = true,
2122     }
2123 }
2124 \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*`.

```

2125 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2126 {
2127     \bool_set_true:c { \__enumext_store_save_key_ \__enumext_level: _bool }
2128     \tl_clear:c { \__enumext_store_save_key_ \__enumext_level: _tl }
2129     \tl_set:ce
2130     { \__enumext_store_save_key_ \__enumext_level: _tl }
2131     { \__enumext_filter_save_key:n {#1} }
2132 }
2133 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2134 {
2135     \bool_set_true:N \__enumext_store_save_key_vii_bool
2136     \tl_clear:N \__enumext_store_save_key_vii_tl
2137     \tl_set:Nx \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2138 }

```

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

11.26.3 Internal functions to store optional arguments

```

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

```

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

```

2139 \cs_new:Npn \__enumext_filter_save_key:n #1
2140 {
2141     \use:e
2142     {
2143         \keyval_parse:NNn
2144         \__enumext_filter_save_key_key:n

```

```

2145         \__enumext_filter_save_key_pair:nn {#1}
2146     }
2147 }

```

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.

```

2148 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2149 {
2150     \str_case:nnF {#1}
2151     {
2152         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2153     }
2154     { , { \exp_not:n {#1} } }
2155 }

```

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.

```

2156 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2157 {
2158     \str_case:nnF {#1}
2159     {
2160         { series } {} { resume } {} { save-ans } {} { save-ref } {}
2161         { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2162         { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2163         { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2164         { mini-right* } {}
2165     }
2166     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2167 }

```

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

11.26.4 Function for storing content in prop list

`__enumext_store_addto_prop:n` 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.

```

2168 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2169 {
2170     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2171     {
2172         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2173     }
2174     { #1 }
2175 }
2176 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }

```

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

11.26.5 Function for storing content in sequence

`__enumext_store_addto_seq:n` 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`.

`__enumext_store_addto_seq:v` 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.

```

2177 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2178 {
2179     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2180 }
2181 \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.

```

2182 \cs_new_protected:Nn \__enumext_store_level_open:
2183 {
2184   \bool_if:NT \l__enumext_check_answers_bool
2185   {
2186     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2187     {
2188       \__enumext_store_addto_seq:n
2189       {
2190         \item \begin{enumext}
2191       }
2192     }
2193     {
2194       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2195       {
2196         \item \begin{enumext} [
2197       }
2198       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2199       {
2200         ]
2201       }
2202       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2203     }
2204   }
2205 }
2206 \cs_new_protected:Nn \__enumext_store_level_close:
2207 {
2208   \bool_if:NT \l__enumext_check_answers_bool
2209   {
2210     \__enumext_store_addto_seq:n { \end{enumext} }
2211   }
2212 }

```

(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.

```

2213 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2214 {
2215   \bool_if:NT \l__enumext_check_answers_bool
2216   {
2217     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2218     {
2219       \__enumext_store_addto_seq:n
2220       {
2221         \item \begin{enumext*}
2222       }
2223     }
2224     {
2225       \tl_put_left:Nn l__enumext_store_save_key_vii_tl
2226       {
2227         \item \begin{enumext*}[
2228       }
2229       \tl_put_right:Nn l__enumext_store_save_key_vii_tl
2230       {
2231         ]
2232       }
2233       \__enumext_store_addto_seq:V l__enumext_store_save_key_vii_tl
2234     }
2235   }
2236 }
2237 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2238 {
2239   \bool_if:NT \l__enumext_check_answers_bool
2240   {
2241     \__enumext_store_addto_seq:n { \end{enumext*} }
2242   }
2243 }

```

(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

2244 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2245 {
2246   \mode_leave_vertical:
2247   \skip_horizontal:n { -\dim_use:N #2 }
2248   \makebox[0pt][ r ]
2249   {
2250     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2251     {
2252       \tl_use:N \l__enumext_mark_answer_sym_tl
2253     }
2254   }
2255   \skip_horizontal:n { \dim_use:N #2 }
2256 }
2257 \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`.

```
2258 \cs_new_protected:Nn \__enumext_store_internal_ref:
2259 {
2260   \cs_set_protected:Npn \__enumext_tmp:n ##1
2261   {
2262     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2263     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2264     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2265     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2266   }
2267   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2268   \cs_set:Npn \__enumext_tmp:n ##1
2269   { . \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.

```
2270 \bool_lazy_all:nT
2271 {
2272   { \bool_if_p:N \g__enumext_starred_bool }
2273   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2274 }
2275 {
2276   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2277   { \tl_use:N \l__enumext_label_copy_vii_tl }
2278 }
2279 \bool_lazy_all:nT
2280 {
2281   { \bool_if_p:N \l__enumext_standar_bool }
2282   { \bool_if_p:N \g__enumext_starred_bool }
2283   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2284 }
2285 {
2286   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2287   {
2288     \tl_use:N \l__enumext_label_copy_vii_tl
2289     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2290   }
2291 }
```

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.

```

2292 \bool_lazy_all:nT
2293 {
2294   { \bool_if_p:N \l__enumext_standar_bool }
2295   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2296   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2297   { \bool_not_p:n { \l__enumext_starred_bool } }
2298 }
2299 {
2300   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2301   {
2302     \tl_use:N \l__enumext_label_copy_i_tl
2303     \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2304   }
2305 }
2306 \cs_set:Npn \l__enumext_tmp:n ##1
2307 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2308 \bool_lazy_all:nT
2309 {
2310   { \bool_if_p:N \l__enumext_standar_bool }
2311   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2312   { \bool_not_p:n { \g__enumext_starred_bool } }
2313   { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
2314 }
2315 {
2316   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2317   {
2318     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2319     . \tl_use:N \l__enumext_label_copy_vii_tl
2320   }
2321 }

```

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

```

2322 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2323 {
2324   \l__enumext_store_name_tl \c_colon_str
2325   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2326 }

```

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

```

2327 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2328 {
2329   \l__enumext_newlabel:nn
2330   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2331   { \l__enumext_newlabel_arg_two_tl }
2332 }
2333 \l__enumext_write_aux_file_tl
2334 }

```

(End of definition for `\l__enumext_store_internal_ref:.`)

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

`\l__enumext_store_anskey_code:n`

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

```

2335 \cs_new_protected:Npn \l__enumext_store_anskey_code:n #1
2336 {
2337   \int_gincr:N \g__enumext_item_anskey_int
2338   \l__enumext_store_addto_prop:n {#1}
2339   \bool_if:NT \l__enumext_store_ref_key_bool
2340   {
2341     \l__enumext_store_internal_ref:
2342   }
2343   \l__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the $[\langle \textit{key} = \textit{val} \rangle]$ passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “*store*” in the $\langle \textit{sequence} \rangle$. First we clear the variable

`\l__enumext_store_anskey_arg_tl` and process the $\langle keys \rangle$, if the `break-col` key is present and the command is running under `enumext` (not in `enumext*`) we will add `\columnbreak` and then `\item`.

```

2344 \tl_clear:N \l__enumext_store_anskey_arg_tl
2345 \bool_lazy_and:nnT
2346 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2347 { \bool_not_p:n { \l__enumext_starred_bool } }
2348 {
2349   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2350 }
2351 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2352 \bool_lazy_and:nnT
2353 { \bool_not_p:n { \l__enumext_starred_bool } }
2354 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2355 {
2356   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2357   {
2358     ( \exp_not:V \l__enumext_store_item_join_int )
2359   }
2360 }

```

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

```

2361 \bool_if:NTF \l__enumext_store_item_star_bool
2362 {
2363   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2364   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2365   {
2366     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2367     {
2368       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2369     }
2370   }
2371   \dim_compare:nT
2372   {
2373     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2374   }
2375   {
2376     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2377     {
2378       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2379     }
2380   }
2381   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2382 }
2383 {
2384   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2385 }

```

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 $\langle sequence \rangle$.

```

2386 \bool_lazy_and:nnT
2387 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2388 { \bool_if_p:N \l__enumext_hyperref_bool }
2389 {
2390   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2391   {
2392     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2393     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2394   }
2395 }
2396 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2397 }

```

(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.

```

2398 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2399 {

```

```

2400 \par
2401 \bool_if:NTF \l__enumext_starred_bool
2402 {
2403   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2404 }
2405 {
2406   \__enumext_print_keyans_box:cc
2407   { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2408   { \l__enumext_labelsep_ \l__enumext_level: _dim }
2409 }
2410 \__enumext_anskey_wrapper:n { #1 }
2411 }

```

(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-anskey` or the “*position*” of the content stored in the *⟨prop list⟩* when using the `show-pos` key on the left margin next to the “*wraps*” *⟨argument⟩* passed to `\anskey` and the *⟨body⟩* in `anskey*` on the right side when using the `show-anskey` key.

```

2412 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2413 {
2414   \bool_if:NT \l__enumext_show_answer_bool
2415   {
2416     \__enumext_anskey_show_wrap_arg:n { #1 }
2417   }
2418   \bool_if:NT \l__enumext_show_position_bool
2419   {
2420     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2421     {
2422       \group_begin:
2423       \exp_not:N \normalfont
2424       \exp_not:N \footnotesize [ \int_eval:n
2425       {
2426         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2427       }
2428       ]
2429       \group_end:
2430     }
2431     \__enumext_anskey_show_wrap_arg:n { #1 }
2432   }
2433 }

```

(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 *⟨sequences⟩* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing.

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

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

```

2434 \keys_define:nn { enumext / anskey }
2435 {
2436   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2437   break-col .default:n = true,
2438   break-col .value_forbidden:n = true,
2439   item-join .int_set:N = \l__enumext_store_item_join_int,
2440   item-join .value_required:n = true,
2441   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2442   item-star .default:n = true,
2443   item-star .value_forbidden:n = true,
2444   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2445   item-sym* .value_required:n = true,
2446   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2447   item-pos* .value_required:n = true,
2448   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2449 }

```

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`.

```

2450 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2451 {
2452   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2453 }
2454 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2455 {
2456   \tl_if_blank:nTF {#2}
2457   {
2458     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2459   }
2460   {
2461     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2462   }
2463 }

```

(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`.

```

2464 \NewDocumentCommand \anskey { o +m }
2465 {
2466   \__enumext_anskey_safe_outer:
2467   \group_begin:
2468     \bool_if:NT \l__enumext_check_answers_bool
2469     {
2470       \tl_if_novalue:nF {#1}
2471       {
2472         \keys_set:nn { enumext / anskey } {#1}
2473       }
2474       \tl_if_blank:nTF {#2}
2475       {
2476         \msg_error:nn { enumext } { anskey-empty-arg }
2477       }
2478       {
2479         \__enumext_anskey_safe_inner:
2480         \__enumext_store_anskey_code:n {#2}
2481       }
2482     }
2483   \group_end:
2484 }

```

(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.

`__enumext_anskey_safe_inner:`

```

2485 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2486 {
2487   \bool_if:NF \l__enumext_store_active_bool
2488   {
2489     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2490   }
2491   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2492   {
2493     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2494   }
2495   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2496   {
2497     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2498   }
2499   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2500   {

```

```

2501     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2502   }
2503 }

```

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.

```

2504 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2505 {
2506   \int_incr:N \__enumext_anskey_level_int
2507   \int_compare:nNt { \__enumext_anskey_level_int } > { 1 }
2508   {
2509     \msg_error:nn { enumext } { anskey-nested }
2510   }
2511   \bool_if:NF \__enumext_item_number_bool
2512   {
2513     \msg_error:nn { enumext } { anskey-unnumber-item }
2514   }
2515   \mode_if_math:T
2516   {
2517     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2518   }
2519 }

```

(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.

```

2520 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2521 {
2522   \cs_undefine:c { anskey* }
2523   \cs_undefine:c { endanskey* }
2524   \cs_undefine:c { __scontents_anskey*_env_begin: }
2525   \cs_undefine:c { __scontents_anskey*_env_end: }
2526 }

```

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

```

2527 \__enumext_before_env:nn { enumext }
2528 {
2529   \bool_lazy_and:nnT
2530   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2531   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2532   {
2533     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2534     {
2535       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2536     }
2537   }
2538 }
2539 \__enumext_before_env:nn { enumext* }
2540 {
2541   \bool_lazy_and:nnT
2542   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2543   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2544   {

```



```

2545         \cs_if_free:cf { __scontents_anskey*_env_begin: }
2546         {
2547             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2548         }
2549     }
2550 }

```

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.

```

2551 \__enumext_before_env:nn { anskey* }
2552 {
2553     \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2554     {
2555         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2556     }
2557     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2558     {
2559         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2560     }
2561     \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2562     {
2563         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2564     }
2565     \bool_if:NF \__enumext_item_number_bool
2566     {
2567         \msg_error:nn { enumext } { anskey-unnumber-item }
2568     }
2569     \mode_if_math:T
2570     {
2571         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2572     }
2573 }

```

(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:($1.25.1)` and we will execute it only if the variable `__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package `scontents`.

```

2574 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2575 {
2576     \bool_if:NT \__enumext_anskey_env_bool
2577     {
2578         \newenvsc{anskey*}[store-env=#1,print-env=false]
2579         \__enumext_anskey_env_exec:
2580     }
2581 }
2582 \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`.

```

2583 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2584 {
2585     \keys_define:nn { scontents / scontents }
2586     {
2587         break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2588         break-col .default:n = true,
2589         break-col .value_forbidden:n = true,
2590         item-join .int_gset:N = \g__enumext_store_item_join_int,
2591         item-join .value_required:n = true,
2592         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2593         item-star .default:n = true,
2594         item-star .value_forbidden:n = true,
2595         item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2596         item-sym* .value_required:n = true,
2597         item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,

```

```

2598     item-pos* .value_required:n = true,
2599     print-env .undefine:,
2600     store-env .undefine:,
2601     write-out .undefine:,
2602     unknown .code:n = { \__enumext_anskey_env_unknown:n {##1} },
2603   }
2604 }

```

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`.

```

2605 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2606 {
2607   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2608 }
2609 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2610 {
2611   \tl_if_blank:nTF {#2}
2612   {
2613     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2614   }
2615   {
2616     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2617   }
2618 }

```

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`.

```

2619 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2620 {
2621   \keys_define:nn { scontents / scontents }
2622   {
2623     break-col .undefine:,
2624     item-join .undefine:,
2625     item-star .undefine:,
2626     item-sym* .undefine:,
2627     item-pos* .undefine:,
2628     write-out .code:n = {
2629       \bool_set_false:N \l__scontents_storing_bool
2630       \bool_set_true:N \l__scontents_writing_bool
2631       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2632     },
2633     write-out .value_required:n = true,
2634     print-env .meta:nn = { scontents } { print-env = ##1 },
2635     print-env .default:n = true,
2636     store-env .meta:nn = { scontents } { store-env = ##1 },
2637     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2638   }
2639 }

```

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*.

```

2640 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2641 {
2642   \group_begin:
2643     \int_set:Nn \tex_newlinechar:D { `^^J }
2644     \__scontents_rescan_tokens:x
2645     {
2646       \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2647       #1
2648     }
2649 }

```

(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⟩*.

```

2650 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2651 {
2652   \__enumext_before_env:nn { anskey* }
2653   {

```

```

2654     \__enumext_anskey_env_define_keys:
2655 }

```

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.

```

2656 \hook_if_empty:nF {env/anskey*/after}
2657 {
2658     \hook_gremove_code:nn {env/anskey*/after} { * }
2659 }
2660 \__enumext_after_env:nn { anskey* }
2661 {
2662     \__enumext_anskey_env_save_keys:
2663     \tl_clear:N \l__enumext_store_anskey_env_tl
2664     \tl_clear:N \l__enumext_store_anskey_opt_tl
2665     \bool_if:NT \l__enumext_check_answers_bool
2666     {
2667         \tl_gset:Ne \l__enumext_store_anskey_env_tl
2668         {
2669             \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2670         }
2671         \regex_match:nVTF
2672         { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2673         \l__enumext_store_anskey_env_tl
2674         {
2675             \msg_error:nn { enumext } { anskey-empty-arg }
2676         }
2677         {
2678             \__enumext_anskey_env_store:
2679         }
2680     }
2681     \__enumext_anskey_env_clean_vars:
2682     \__enumext_anskey_env_reset_keys:
2683 }
2684 }

```

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

(End of definition for `__enumext_anskey_env_exec:.`)

```

\__enumext_anskey_env_save_keys:
\__enumext_anskey_env_store:
\__enumext_anskey_env_clean_vars:

```

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

```

2685 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2686 {
2687     \bool_lazy_and:nnT
2688     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2689     { \bool_not_p:n { \l__enumext_starred_bool } }
2690     {
2691         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2692     }

```

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

```

2693     \bool_lazy_and:nnT
2694     { \bool_not_p:n { \l__enumext_starred_bool } }
2695     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2696     {
2697         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2698         {
2699             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2700         }
2701     }

```

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

```

2702     \bool_if:NT \g__enumext_store_item_star_bool
2703     {
2704         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2705         {
2706             ,item-star,

```

```

2707     }
2708     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2709     {
2710         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2711         {
2712             ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2713         }
2714     }
2715     \dim_compare:nT
2716     {
2717         \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2718     }
2719     {
2720         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2721         {
2722             ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2723         }
2724     }
2725 }
2726 }

```

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`.

```

2727 \cs_new_protected:Nn \__enumext_anskey_env_store:
2728 {
2729     \group_begin:
2730     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2731     {
2732         \exp_args:Ne
2733         \__enumext_store_anskey_code:n
2734         {
2735             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2736         }
2737     }
2738     {
2739         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2740         \exp_args:Ne
2741         \__enumext_store_anskey_code:n
2742         {
2743             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2744         }
2745     }
2746     \group_end:
2747 }

```

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

```

2748 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2749 {
2750     \bool_gset_false:N \g__enumext_store_columns_break_bool
2751     \int_gzero:N \g__enumext_store_item_join_int
2752     \bool_gset_false:N \g__enumext_store_item_star_bool
2753     \tl_gclear:N \g__enumext_store_item_symbol_tl
2754     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2755 }

```

(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 `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.

```

2756 \cs_new_protected:Nn \__enumext_execute_after_env:
2757 {

```

```

2758 \int_compare:nNt { \l__enumext_level_int } = { 0 }
2759 {
2760   \tl_if_empty:NF \g__enumext_store_name_tl
2761   {
2762     \__enumext_stop_save_ans_msg:
2763     \__enumext_item_answer_diff:
2764     \__enumext_log_global_vars:
2765     \__enumext_log_answer_vars:
2766     \bool_if:NTF \g__enumext_check_ans_key_bool
2767     {
2768       \__enumext_check_ans_show:
2769     }
2770     { \__enumext_check_ans_log: }
2771     \__enumext_undefine_anskey_env:
2772   }
2773   \__enumext_reset_global_vars:
2774 }
2775 }

```

(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*` (§11.43) 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`.

```

2776 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2777 {
2778   \tl_clear:N \l__enumext_store_current_label_tl
2779   \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
2780   {
2781     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2782   }
2783   {
2784     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2785   }
2786   \tl_if_novalue:nF { #1 }
2787   {
2788     % Set save-sep
2789     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2790     {
2791       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2792     }
2793     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2794   }
2795   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2796 }

```

(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).

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`__enumext_keyans_store_ref_aux_ii:`

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

```

2797 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2798 {
2799   \bool_if:NT \l__enumext_store_ref_key_bool
2800   {
2801     \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

2802     {
2803         \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2804         \tl_reverse:c { \__enumext_label_copy_##1_tl }
2805         \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2806         \tl_reverse:c { \__enumext_label_copy_##1_tl }
2807     }
2808     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2809     \__enumext_keyans_store_ref_aux_i:
2810 }
2811 }

```

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

```

2812 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2813 {
2814     \bool_if:NT \g__enumext_starred_bool
2815     {
2816         \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2817     }
2818     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2819     {
2820         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2821         { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2822     }
2823     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2824     {
2825         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2826         { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2827     }
2828     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2829     {
2830         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2831         { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
2832     }
2833     \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
2834     {
2835         \__enumext_store_name_tl \c_colon_str
2836         \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
2837     }
2838     \__enumext_keyans_store_ref_aux_ii:
2839 }

```

Now auxiliary function `__enumext_keyans_store_ref_aux_ii:` save the result in the variable `__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```

2840 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2841 {
2842     \tl_put_right:Ne \__enumext_write_aux_file_tl
2843     {
2844         \__enumext_newlabel:nn
2845         { \exp_not:V \__enumext_newlabel_arg_one_tl }
2846         { \__enumext_newlabel_arg_two_tl }
2847     }
2848     \__enumext_write_aux_file_tl
2849 }

```

(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 $\langle label \rangle$ `__enumext_label_v_tl` for the `keyans` environment and the `__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the $\langle contents \rangle$ of the optional argument of both commands to the `__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

2850 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2851 {
2852     \tl_clear:N \__enumext_store_current_label_tl
2853     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2854     {
2855         \tl_put_right:Ne \__enumext_store_current_label_tl { \item \__enumext_label_vi_tl }

```

```

2856     }
2857     {
2858         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2859     }
2860     \tl_if_novalue:nF { #1 }
2861     {
2862         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2863         {
2864             \tl_put_right:Ne \l__enumext_store_current_label_tl
2865             {
2866                 \l__enumext_store_keyans_item_opt_sep_tl
2867             }
2868         }
2869         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2870     }
2871     \__enumext_keyans_addto_seq_link:
2872 }

```

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.

```

2873 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2874 {
2875     \bool_lazy_and:nnT
2876     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2877     { \bool_if_p:N \l__enumext_hyperref_bool }
2878     {
2879         \tl_put_right:Ne \l__enumext_store_current_label_tl
2880         {
2881             \hfill \exp_not:N \hyperlink
2882             {
2883                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2884             }
2885             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2886         }
2887     }
2888     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2889     \bool_if:NT \l__enumext_check_answers_bool
2890     {
2891         \int_gincr:N \g__enumext_item_anskey_int
2892     }
2893 }

```

(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.

```

2894 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2895 {
2896     \tl_if_novalue:nF { #1 }
2897     {
2898         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2899     }
2900     \bool_if:NT \l__enumext_show_answer_bool
2901     {
2902         \__enumext_keyans_show_ans:
2903     }
2904     \bool_if:NT \l__enumext_show_position_bool
2905     {
2906         \__enumext_keyans_show_pos:
2907     }
2908 }
2909 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2910 {

```



```

2911 \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2912 {
2913   \bool_lazy_or:nnT
2914     { \bool_if_p:N \l__enumext_show_answer_bool }
2915     { \bool_if_p:N \l__enumext_show_position_bool }
2916   {
2917     \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2918   }
2919 }
2920 }
2921 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2922 {
2923   \bool_if:NT \l__enumext_starred_bool
2924   {
2925     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2926     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2927   }
2928   \tl_put_left:Nn \l__enumext_label_v_tl
2929   {
2930     \__enumext_print_keyans_box:NN
2931     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2932   }
2933 }
2934 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2935 {
2936   \bool_if:NT \l__enumext_starred_bool
2937   {
2938     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2939     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2940   }
2941   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2942   {
2943     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2944     {
2945       \group_begin:
2946       \exp_not:N \normalfont
2947       \exp_not:N \footnotesize [ \int_eval:n
2948         {
2949           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2950         }
2951       ]
2952       \group_end:
2953     }
2954   }
2955   {
2956     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2957     {
2958       \group_begin:
2959       \exp_not:N \normalfont
2960       \exp_not:N \footnotesize [ \int_eval:n
2961         {
2962           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2963         }
2964       ]
2965       \group_end:
2966     }
2967   }
2968   \tl_put_left:Nn \l__enumext_label_v_tl
2969   {
2970     \__enumext_print_keyans_box:NN
2971     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2972   }
2973 }

```

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

11.33 Redefining `\item` and `\makeLabel` in `enumext`

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makeLabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext` and we will add `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

`__enumext_default_item:n`

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the optional argument and the key `itemindent`.

```

2974 \cs_new_protected:Npn \__enumext_default_item:n #1
2975 {
2976   \tl_if_novalue:nTF {#1}
2977   {
2978     \bool_if:NT \__enumext_check_answers_bool
2979     {
2980       \int_gincr:N \g__enumext_item_number_int
2981       \bool_set_true:N \__enumext_item_number_bool
2982     }
2983     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2984     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2985   }
2986   {
2987     \bool_set_eq:cc
2988     { \__enumext_wrap_label_ \__enumext_level: _bool }
2989     { \__enumext_wrap_label_opt_ \__enumext_level: _bool }
2990     \__enumext_item_std:w [#1] \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2991   }
2992 }

```

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

`__enumext_starred_item:nn`

`__enumext_item_star_exec:`

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

`#1: __enumext_item_symbol_X_tl`

`#2: __enumext_item_symbol_sep_X_dim`

First we will make a copy of `__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as “*first*” optional argument in the global variable `\g__enumext_item_symbol_aux_tl`, followed by setting the variable `__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 `__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`.

```

2993 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2994 {
2995   \tl_if_novalue:nTF {#1}
2996   {
2997     \tl_gset_eq:Nc
2998     \g__enumext_item_symbol_aux_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2999   }
3000   {
3001     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3002   }
3003   \tl_if_novalue:nTF {#2}
3004   {
3005     \dim_set_eq:cc
3006     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
3007     { \__enumext_labelsep_ \__enumext_level: _dim }
3008   }
3009   {
3010     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3011   }
3012   \bool_if:NT \__enumext_check_answers_bool
3013   {
3014     \int_gincr:N \g__enumext_item_number_int
3015     \bool_set_true:N \__enumext_item_number_bool
3016   }
3017   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3018   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3019 }

```

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

```

3020 \cs_new_protected:Nn \__enumext_item_star_exec:
3021 {
3022   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3023   {
3024     \mode_leave_vertical:
3025     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3026     \makebox[0pt][r]{ \g__enumext_item_symbol_aux_tl }
3027     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3028   }
3029 }

```

(End of definition for `__enumext_starred_item:nn` and `__enumext_item_star_exec:`.)

```

\__enumext_redefine_item:
\__enumext_make_label

```

The function `__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`.

```

3030 \cs_new_protected:Nn \__enumext_redefine_item:
3031 {
3032   \RenewDocumentCommand \item { s o o }
3033   {
3034     \bool_if:nTF {##1}
3035     {
3036       \__enumext_starred_item:nn {##2} {##3}
3037     }
3038     { \__enumext_default_item:n {##2} }
3039   }
3040 }

```

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

```

3041 \cs_new_protected:Nn \__enumext_make_label:
3042 {
3043   \RenewDocumentCommand \makelabel { m }
3044   {
3045     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
3046     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
3047     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
3048     {
3049       \__enumext_item_star_exec:
3050       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3051     }
3052     { ##1 }
3053     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
3054     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3055   }
3056 }

```

(End of definition for `__enumext_redefine_item:` and `__enumext_make_label:`.)

- These functions are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§11.38).

11.34 Setting `item-sym*` and `item-pos*` keys

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

```

item-sym*
item-pos*

```

Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```

3057 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3058 {
3059   \keys_define:nn { enumext / #1 }
3060   {
3061     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
3062     item-sym* .value_required:n = true,
3063     item-sym* .initial:n = { $\star$ },
3064     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
3065     item-pos* .value_required:n = true,
3066   }
3067 }
3068 \clist_map_inline:nn
3069 {
3070   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}

```

```

3071 }
3072 { \__enumext_tmp:nn #1 }

```

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

11.35 Handling unknown keys

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

11.35.1 Handling unknown keys for keyans and keyans*

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

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3073 \cs_set_protected:Npn \__enumext_tmp:n #1
3074 {
3075   \keys_define:nn { enumext / #1 }
3076   {
3077     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3078   }
3079 }
3080 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling *unknown* key.

```

3081 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3082 {
3083   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3084 }
3085 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3086 {
3087   \tl_if_blank:nTF {#2}
3088   {
3089     \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3090   }
3091   {
3092     \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3093   }
3094 }

```

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

11.35.2 Handling unknown keys for enumext*

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

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3095 \keys_define:nn { enumext / enumext* }
3096 {
3097   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3098 }

```

Internal functions for handling *unknown* key.

```

3099 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3100 {
3101   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3102 }
3103 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3104 {
3105   \tl_if_blank:nTF {#2}
3106   {
3107     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3108   }
3109   {
3110     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3111   }
3112 }

```

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

11.35.3 Handling unknown keys for enumext

unknown Defines and set the key `unknown` for `enumext` environment.

```

3113 \cs_set_protected:Npn \__enumext_tmp:n #1
3114 {
3115   \keys_define:nn { enumext / #1 }
3116   {
3117     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3118   }
3119 }
3120 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3121 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3122 {
3123   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3124 }
3125 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3126 {
3127   \tl_if_blank:nTF {#2}
3128   {
3129     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3130   }
3131   {
3132     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3133   }
3134 }

```

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

11.36 Redefining `\item` and `\makeLabel` in keyans

The `\item` and `\item[⟨custom⟩]` commands work in the usual way in `keyans`, but the `\item*` and `\item*[⟨content⟩]` commands *store* the current `⟨label⟩` next to the `⟨content⟩` if it is present in the `⟨sequence⟩` and `⟨prop list⟩` defined by `save-ans` key.

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

```

3135 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3136 {
3137   \tl_if_novalue:nTF { #1 }
3138   {
3139     \bool_set_true:N \__enumext_wrap_label_v_bool
3140     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3141   }
3142   {
3143     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3144     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3145   }
3146 }

```

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

`__enumext_keyans_starred_item:n` The function `__enumext_keyans_starred_item:n` which will make a temporary copy of the current `⟨label⟩`, execute the `show-ans` or `show-pos` keys using the function `__enumext_keyans_show_left:n` and will display the contents of that item using the internal copy `__enumext_item_std:w`, this is necessary to prevent incrementing the current “*counter*” of the original `⟨label⟩`.

```

3147 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3148 {
3149   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3150   \__enumext_keyans_show_left:n { #1 }
3151   \bool_set_true:N \__enumext_wrap_label_v_bool
3152   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item

```

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

```

3153   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3154   \__enumext_keyans_addto_prop:n { #1 }
3155   \__enumext_keyans_store_ref:
3156   \__enumext_keyans_addto_seq:n { #1 }
3157   \int_gincr:N \g__enumext_check_starred_cmd_int
3158 }

```

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

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

```

3159 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3160 {
3161   \RenewDocumentCommand \item { s o }
3162   {
3163     \bool_if:nTF {##1}
3164     {
3165       \peek_remove_spaces:n
3166       {
3167         \__enumext_keyans_starred_item:n {##2}
3168       }
3169     }
3170     {
3171       \__enumext_keyans_default_item:n {##2}
3172     }
3173   }
3174 }

```

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

```

3175 \cs_new_protected:Nn \__enumext_keyans_make_label:
3176 {
3177   \RenewDocumentCommand \makeLabel { m }
3178   {
3179     \tl_use:N \l__enumext_label_fill_left_v_tl
3180     \tl_use:N \l__enumext_label_font_style_v_tl
3181     \bool_if:NTF \l__enumext_wrap_label_v_bool
3182     {
3183       \__enumext_wrapper_label_v:n { ##1 }
3184     }
3185     { ##1 }
3186     \tl_use:N \l__enumext_label_fill_right_v_tl
3187   }
3188 }

```

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

- This functions are passed to `__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§11.37.2).

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.

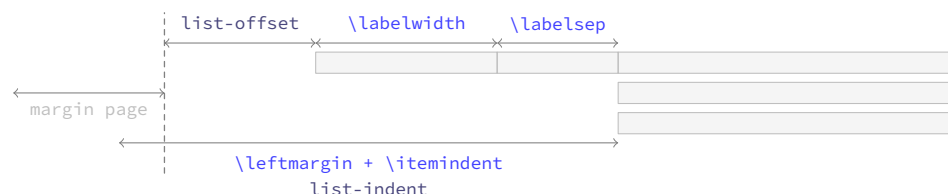


Figure 9: Representation of standard horizontal lengths in `list` environment.

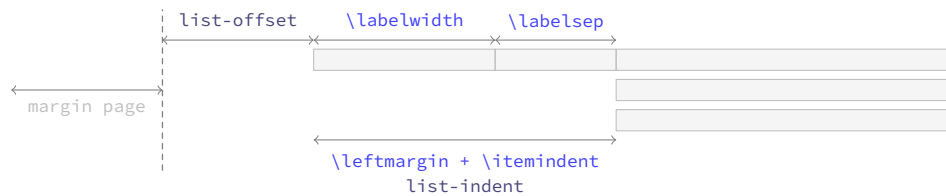
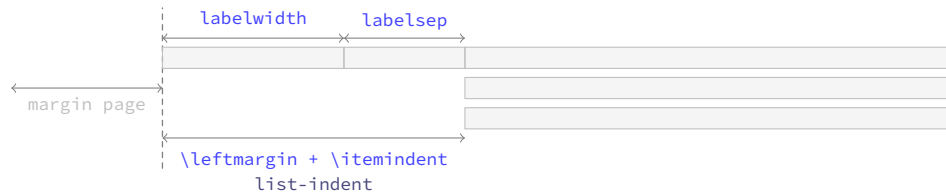
The idea is to have control over these margins so that our list does not overlap the left margin of the page. The *key* relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the *label box* is at `\leftmargin + \itemindent` minus `\labelwidth + \labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10. Where the default values will look like in the figure 11.

```

\__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:

Figure 10: Representation of horizontal lengths concept in list in `enumext`.Figure 11: Default horizontal lengths in `enumext`.

```

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

```

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

This function is passed to `__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§11.37.2).

```

3189 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3190 {
3191   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3192   {
3193     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3194     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3195   }
3196   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3197   {
3198     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3199     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3200   }

```

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

```

3201   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3202   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3203   {
3204     \dim_set:Nn #6 { #1 + #2 - #4 }
3205     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3206   }
3207   {
3208     \dim_compare:nNnT { #4 } = { #1 + #2 }
3209     { \dim_set:Nn #6 { \c_zero_dim } }
3210     \dim_compare:nNnT { #4 } < { #1 + #2 }
3211     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3212     \dim_compare:nNnT { #4 } > { #1 + #2 }
3213     {
3214       \dim_set:Nn #6 { -#1 - #2 + #4 }
3215       \dim_set:Nn #6 { #6*-1 }
3216     }
3217     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3218   }
3219 }
3220 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNN { ccccccc }

```

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

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.

```

3221 \cs_set_protected:Npn \__enumext_tmp:n #1
3222 {
3223   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3224   {
3225     \__enumext_calc_hspace:ccccc
3226     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3227     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3228     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3229     { \__enumext_leftmargin_tmp_#1_bool }
3230     \clist_map_inline:nn
3231       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3232       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3233     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3234       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3235     \usecounter { enumX#1 }
3236     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3237     \str_if_eq:nnTF {#1} { v }
3238     {
3239       \__enumext_keyans_redefine_item:
3240       \__enumext_keyans_make_label:
3241       \__enumext_keyans_ref:
3242       \__enumext_keyans_fake_item:
3243       \bool_if:cT { \__enumext_show_length_#1_bool }
3244       {
3245         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3246       }
3247     }
3248     {
3249       \__enumext_redefine_item:
3250       \__enumext_make_label:
3251       \__enumext_standar_ref:
3252       \__enumext_fake_item:
3253       \bool_if:cT { \__enumext_show_length_#1_bool }
3254       {
3255         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
3256       }
3257     }
3258   }
3259 }
3260 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

3261 \cs_set_protected:Npn \__enumext_tmp:n #1
3262 {
3263   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3264   {
3265     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3266     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3267     \__enumext_calc_hspace:ccccc
3268     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3269     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3270     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3271     { \__enumext_leftmargin_tmp_#1_bool }
3272     \clist_map_inline:nn
3273       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3274       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3275     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3276       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3277     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3278     \skip_zero:N \partopsep
3279     \usecounter { enumX#1 }
3280     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```

```

3281     \__enumext_starred_ref:
3282     \str_if_eq:nnTF {#1} { vii }
3283     {
3284         \__enumext_fake_item_vii:
3285         \bool_if:cT { \__enumext_show_length_vii_bool }
3286         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3287     }
3288     {
3289         \__enumext_fake_item_viii:
3290         \bool_if:cT { \__enumext_show_length_#1_bool }
3291         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3292     }
3293 }
3294 }
3295 \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.

```

3296 \NewDocumentEnvironment{enumext}{0}{ }
3297 {
3298     \__enumext_safe_exec:
3299     \__enumext_parse_keys:n {#1}
3300     \__enumext_before_list:
3301     \__enumext_start_store_level:
3302     \__enumext_start_list:nn
3303     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3304     {
3305         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3306         \__enumext_before_keys_exec:
3307     }
3308     \__enumext_set_item_width:
3309     \__enumext_after_args_exec:
3310 }
3311 {
3312     \__enumext_stop_list:
3313     \__enumext_stop_store_level:
3314     \__enumext_after_list:
3315 }

```

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

__enumext_set_item_width: The function **__enumext_set_item_width:** will set the value of **\itemwidth** taking into account the value established by the **list-offset** key for each level of the environment.

```

3316 \cs_new_protected:Nn \__enumext_set_item_width:
3317 {
3318     \dim_set:Nn \itemwidth
3319     {
3320         \linewidth
3321     }
3322     \dim_compare:nT
3323     {
3324         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3325     }
3326     {
3327         \dim_sub:Nn \itemwidth
3328         {
3329             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3330         }
3331     }
3332 }

```

(End of definition for __enumext_set_item_width:.)

__enumext_safe_exec: The **__enumext_safe_exec:** function first call the function **__enumext_internal_mini_page:** to create the environment **__enumext_mini_env***, then the function **__enumext_is_not_nested:** which sets **\g__enumext_standar_bool** to “true” if we are not nested within **enumext***, we will increment **__enumext_level_int** to restrict nesting of the environment, set **__enumext_standar_bool**

to “true” and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level”.

```

3333 \cs_new_protected:Nn \__enumext_safe_exec:
3334 {
3335   \__enumext_internal_mini_page:
3336   \__enumext_is_not_nested:
3337   \int_incr:N \l__enumext_level_int
3338   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3339     { \msg_fatal:nn { enumext } { list-too-deep } }
3340   \bool_set_true:N \l__enumext_standar_bool
3341   \bool_set_false:N \l__enumext_starred_bool
3342   \__enumext_is_on_first_level:
3343 }

```

(End of definition for `__enumext_safe_exec:`)

`__enumext_parse_keys:n`

The `__enumext_parse_store_keys:n` function first we will clear the variable `\l__enumext_series_str` used by the key `series` and then we check if we are at the “first level”, if so we process the `<keys>` and then execute the function `__enumext_parse_series:n` used by the key `series` and call the function `__enumext_nested_base_line_fix:` used by the key `base-fix`, otherwise we will pass the `<keys>` to the inner levels of the environment then we execute the function `__enumext_store_active_keys:n` and reprocess the `<keys>` to pass them to the storage `<sequence>` if the key `save-key` is not active.

```

3344 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3345 {
3346   \tl_if_novalue:nF {#1}
3347   {
3348     \str_clear:N \l__enumext_series_str
3349     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3350     {
3351       \keys_set:nn { enumext / level-1 } {#1}
3352       \__enumext_parse_series:n {#1}
3353       \__enumext_nested_base_line_fix:
3354     }
3355     {
3356       \exp_args:Ne \keys_set:nn
3357         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3358     }
3359     \__enumext_store_active_keys:n {#1}
3360   }
3361 }

```

(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*`.

```

3362 \cs_new_protected:Nn \__enumext_start_store_level:
3363 {
3364   \bool_lazy_all:nT
3365   {
3366     { \bool_if_p:N \l__enumext_store_active_bool }
3367     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3368     { \bool_if_p:N \g__enumext_standar_bool }
3369   }
3370   {
3371     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3372     {
3373       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3374       \__enumext_store_level_open:
3375     }
3376   }

```

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

```

3377 \bool_lazy_all:nT
3378 {
3379   { \bool_if_p:N \l__enumext_store_active_bool }
3380   { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3381   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3382 }

```

```

3383     {
3384         \int_compare:nNtT { \l__enumext_level_int } > { 0 }
3385         {
3386             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3387             \__enumext_store_level_open:
3388         }
3389     }
3390 }

```

Close the stored structure.

```

3391 \cs_new_protected:Nn \__enumext_stop_store_level:
3392 {
3393     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3394     {
3395         \__enumext_store_level_close:
3396     }
3397 }

```

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

`__enumext_before_list:` The function `__enumext_before_list:` first calls the function `__enumext_vspace_above:` used by the keys `above` and `above*`, then calls the function `__enumext_before_args_exec:` used by the key `before*` and finally execute the function `__enumext_check_ans_active:` for the check answer mechanism.

```

3398 \cs_new_protected:Nn \__enumext_before_list:
3399 {
3400     \__enumext_vspace_above:
3401     \__enumext_before_args_exec:
3402     \__enumext_check_ans_active:

```

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

```

3403     \dim_compare:nNt
3404     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3405     {
3406         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3407         {
3408             \linewidth
3409             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3410             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3411         }

```

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

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

```

3412         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3413         \int_gincr:N \g__enumext_minipage_stat_int
3414         \__enumext_mini_addvspace:
3415         \nointerlineskip\noindent
3416         \begin{\__enumext_mini_env*}
3417         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3418     }
3419     \__enumext_multicols_start:
3420 }

```

(End of definition for __enumext_before_list:.)

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

```

3421 \cs_new_protected:Nn \__enumext_multicols_start:

```

```

3422 {
3423   \int_compare:nNnT
3424     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3425   {
3426     \dim_compare:nNnT
3427       { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3428     {
3429       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3430       {
3431         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3432           + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3433         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3434         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3435       }
3436     }
3437     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3438     \skip_zero:N \multicolsep
3439     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3440     {
3441       \dim_zero:N \columnseprule
3442     }

```

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.

```

3443   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3444   {
3445     \__enumext_multi_addvspace:
3446   }
3447   \raggedcolumns
3448   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3449 }
3450 }

```

(End of definition for `__enumext_multicols_start:`)

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

```

3451 \cs_new_protected:Nn \__enumext_multicols_stop:
3452 {
3453   \int_compare:nNnT
3454     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3455   {
3456     \end{multicols}
3457     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3458     {
3459       \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3460     }
3461   }
3462 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_after_list:` The function `__enumext_after_list:` first check the state of the boolean variable `__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.

```

3463 \cs_new_protected:Nn \__enumext_after_list:
3464 {
3465   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3466   {
3467     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3468     {
3469       \msg_warning:nn { enumext } { missing-miniright }
3470       \miniright
3471     }
3472     \int_gzero:N \g__enumext_minipage_stat_int
3473     \end{\__enumext_mini_env*}
3474     \par\addvspace { \l__enumext_minipage_after_skip }

```

```

3475     }
3476     { \__enumext_multicols_stop: }

```

Now we will execute the functions `__enumext_after_stop_list:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below:` used by the keys `below` and `below*`. Finally set `\l__enumext_standar_bool` to false and call the function `__enumext_resume_save_counter:` used by the `series`, `resume` and `resume*` keys.

```

3477     \__enumext_after_stop_list:
3478     \__enumext_check_ans_key_hook:
3479     \__enumext_vspace_below:
3480     \bool_set_false:N \l__enumext_standar_bool
3481     \__enumext_resume_save_counter:
3482 }

```

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`.

```

3483 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

(End of definition for `__enumext_after_list:`.)

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.

```

3484 \NewDocumentEnvironment{keyans}{0}{ }
3485 {
3486     \__enumext_keyans_safe_exec:
3487     \__enumext_keyans_parse_keys:n {#1}
3488     \__enumext_before_list_v:
3489     \__enumext_start_list:nn
3490     { \tl_use:N \l__enumext_label_v_tl }
3491     {
3492         \__enumext_list_arg_two_v:
3493         \__enumext_before_keys_exec_v:
3494     }
3495     \__enumext_keyans_set_item_width:
3496     \__enumext_after_args_exec_v:
3497 }
3498 {
3499     \__enumext_check_starred_cmd:n { item }
3500     \__enumext_stop_list:
3501     \__enumext_after_list_v:
3502 }

```

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

`__enumext_keyans_set_item_width:` The function `__enumext_keyans_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key.

```

3503 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3504 {
3505     \dim_set:Nn \itemwidth
3506     {
3507         \linewidth
3508     }
3509     \dim_compare:nT
3510     {
3511         \l__enumext_listoffset_v_dim != \c_zero_dim
3512     }
3513     {
3514         \dim_sub:Nn \itemwidth
3515         {
3516             \l__enumext_listoffset_v_dim
3517         }
3518     }
3519 }

```

(End of definition for `__enumext_keyans_set_item_width:`.)

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

```

3520 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3521 {
3522   \bool_if:NF \l__enumext_store_active_bool
3523   {
3524     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3525   }
3526   \int_incr:N \l__enumext_keyans_level_int
3527   \bool_set_true:N \l__enumext_keyans_env_bool
3528   \__enumext_keyans_name_and_start:
3529   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3530   \bool_set_false:N \l__enumext_store_active_bool
3531   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3532   {
3533     \msg_error:nn { enumext } { keyans-nested }
3534   }
3535   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3536   {
3537     \msg_error:nn { enumext } { keyans-wrong-level }
3538   }
3539 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

3540 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3541 {
3542   \keys_set:nn { enumext / keyans } {#1}
3543 }

```

(End of definition for `__enumext_keyans_parse_keys:n`)

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

```

\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_after_list_v:
3544 \cs_new_protected:Nn \__enumext_before_list_v:
3545 {
3546   \__enumext_vspace_above_v:
3547   \__enumext_before_args_exec_v:
3548   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3549   {
3550     \dim_set:Nn \l__enumext_minipage_left_v_dim
3551     {
3552       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3553     }
3554     \bool_set_true:N \l__enumext_minipage_active_v_bool
3555     \int_gincr:N \g__enumext_minipage_stat_int
3556     \__enumext_keyans_mini_addvspace:
3557     \nointerlineskip\noident
3558     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3559   }
3560   \__enumext_keyans_multicols_start:
3561 }
3562 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3563 {
3564   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3565   {
3566     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3567     {
3568       \dim_set:Nn \l__enumext_columns_sep_v_dim
3569       {
3570         (
3571           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3572         ) / \l__enumext_columns_v_int
3573         - \l__enumext_listoffset_v_dim
3574       }
3575     }
3576     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3577     \skip_zero:N \multicolsep
3578     \dim_zero:N \columnseprule % no rule here
3579     \bool_if:NF \l__enumext_minipage_active_v_bool

```



```
3580         {
3581             \__enumext_keyans_multi_addvspace:
3582         }
3583         \raggedcolumns
3584         \begin{multicols}{\l__enumext_columns_v_int }
3585     }
3586 }
3587 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3588 {
3589     \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3590     {
3591         \end{multicols}
3592         \bool_if:NF \l__enumext_minipage_active_v_bool
3593         {
3594             \par\addvspace{ \l__enumext_multicols_below_v_skip }
3595         }
3596     }
3597 }
3598 \cs_new_protected:Nn \__enumext_after_list_v:
3599 {
3600     \bool_if:NTF \l__enumext_minipage_active_v_bool
3601     {
3602         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3603         {
3604             \msg_warning:nn { enumext } { missing-miniright }
3605             \miniright
3606         }
3607         \int_gzero:N \g__enumext_minipage_stat_int
3608         \end{__enumext_mini_env*}
3609         \par\addvspace{ \l__enumext_minipage_after_skip }
3610     }
3611     {
3612         \__enumext_keyans_multicols_stop:
3613     }
3614     \bool_set_false:N \l__enumext_keyans_env_bool
3615     \__enumext_after_stop_list_v:
3616     \__enumext_vspace_below_v:
3617 }
```

(End of definition for __enumext_before_list_v: and others.)

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.

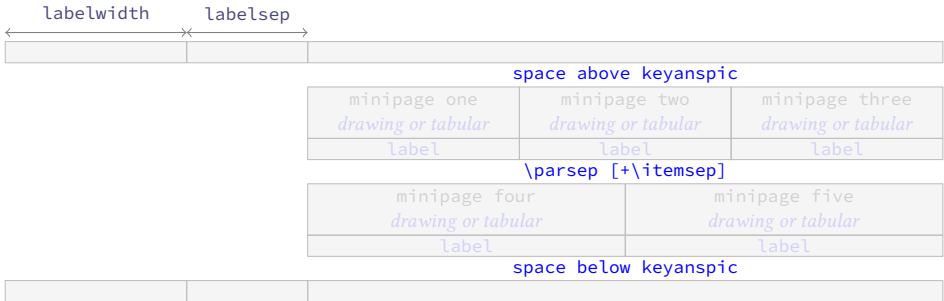


Figure 12: Representation of the **keyanspic** spacing in **enumext**.

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.

```
3618 \NewDocumentCommand \anspic { s o +m }
```

```
3619 {
```

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

```
3620 \bool_if:NF \l__enumext_store_active_bool
3621 {
3622   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3623 }
3624 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3625 {
3626   \msg_error:nn { enumext } { keyanspic-wrong-level }
3627 }
3628 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3629 {
3630   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3631 }
```

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.

```
3632 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3633 {
3634   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3635 }
3636 }
```

(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.

```
3637 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3638 {
3639   \stepcounter { enumXvi }
3640   #3 \\\
3641   \bool_if:nT { #1 }
3642   {
3643     \__enumext_keyans_addto_prop:n { #2 }
3644     \__enumext_keyans_store_ref:
3645     \__enumext_keyans_addto_seq:n { #2 }
3646     \int_gincr:N \g__enumext_check_starred_cmd_int
3647     \bool_lazy_or:nnT
3648     { \bool_if_p:N \l__enumext_show_answer_bool }
3649     { \bool_if_p:N \l__enumext_show_position_bool }
3650     {
3651       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3652       \__enumext_keyans_show_left:n { #2 }
3653       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3654     }
3655   }
3656   \tl_use:N \l__enumext_label_font_style_v_tl
3657   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3658 }
```

(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.

```
3659 \NewDocumentEnvironment{keyanspic}{ o }
3660 {
3661   \__enumext_keyans_pic_safe_exec:
3662   \__enumext_start_list:nn
3663   { }
3664   {
3665     \__enumext_keyans_pic_arg_two:
3666   }
```

We apply the “adjusted” vertical spacing above the environment

```
3667 \vspace { \l__enumext_keyans_pic_above_skip }
3668 }
```

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.

```

3669 {
3670   \tl_if_novalue:nTF { #1 }
3671   {
3672     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3673   }
3674   { \__enumext_keyans_pic_do:n { #1 } }
3675   \__enumext_stop_list:
3676   \__enumext_check_starred_cmd:n { anspic }
3677   \setcounter { enumXvi } { 0 }
3678   \vspace { \l__enumext_topsep_v_skip }
3679   %\bool_set_false:N \l__enumext_store_active_bool
3680 }

```

(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.

```

3681 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3682 {
3683   \int_incr:N \l__enumext_keyans_pic_level_int
3684   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3685   {
3686     \msg_error:nn { enumext } { keyanspic-nested }
3687   }
3688   \__enumext_keyans_name_and_start:
3689 }

```

(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`.

```

3690 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3691 {
3692   \dim_compare:nNnT { #1 } < { 0pt }
3693   { \skip_set:Nn #1 { -#1 } }
3694 }

```

(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.

```

3695 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3696 {

```

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

```

3697   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3698   \__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.

```

3699   \skip_add:Nn \parsep { \itemsep }
3700   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3701   \dim_zero:N \labelwidth
3702   \dim_zero:N \listparindent
3703   \dim_zero:N \labelsep
3704   \skip_zero:N \partopsep
3705   \skip_zero:N \itemsep

```

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

```

3706   \__enumext_keyans_pic_skip_abs:N \parsep
3707   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3708   {

```

```

3709     \box_dp:N \strutbox
3710     + \l__enumext_topsep_v_skip
3711     - \parsep
3712   }
3713   \__enumext_item_std:w \scan_stop:
3714   % paranoia
3715   \RenewDocumentCommand \item {}
3716   {
3717     \msg_error:nn { enumext } { keyanspic-item-cmd }
3718   }
3719 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
 __enumext_keyans_pic_do:e

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.

```

3720 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3721 {
3722   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3723 }
3724 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

__enumext_keyans_pic_row:n

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

```

3725 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3726 {
3727   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3728   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3729   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3730   \int_step_inline:nnn
3731   { \l__enumext_keyans_pic_above_int + 1 }
3732   { \l__enumext_keyans_pic_below_int }
3733   {
3734     \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3735     \centering
3736     \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3737     \__enumext_endminipage:
3738   }
3739   \par
3740 }

```

(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.

11.42 Redefining \footnote command

__enumext_footnotetext:nn
 __enumext_renew_footnote:
 __enumext_print_footnote:

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

```

3741 \cs_new_protected:Nn \__enumext_footnotetext:nn
3742 {
3743   \footnotetext[#1]{#2}
3744 }
3745 \cs_new_protected:Nn \__enumext_renew_footnote:
3746 {
3747   \seq_gclear:N \g__enumext_footnote_arg_seq
3748   \seq_gclear:N \g__enumext_footnote_int_seq
3749   \RenewDocumentCommand \footnote { o +m }
3750   {
3751     \tl_if_novalue:nTF {##1}
3752     {
3753       \stepcounter{footnote}

```

```

3754         \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3755     }
3756     {
3757         \int_gset:Nn \g__enumext_footnote_int { ##1 }
3758     }
3759     \footnotemark [ \g__enumext_footnote_int ]
3760     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3761     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3762 }
3763 }
3764 \cs_new_protected:Nn \__enumext_print_footnote:
3765 {
3766     \seq_if_empty:NF \g__enumext_footnote_int_seq
3767     {
3768         \seq_map_pairwise_function:NNN
3769         \g__enumext_footnote_int_seq
3770         \g__enumext_footnote_arg_seq
3771         \__enumext_footnotetext:nn
3772     }
3773 }

```

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

11.42.1 Functions for item box width

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

```

\__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.

```

3774 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3775 {
3776     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3777     {
3778         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3779         {
3780             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3781             / \l__enumext_columns_vii_int
3782         }
3783     }
3784     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3785     \dim_set:Nn \l__enumext_item_width_vii_dim
3786     {
3787         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3788         / \l__enumext_columns_vii_int
3789         - \l__enumext_labelwidth_vii_dim
3790         - \l__enumext_labelsep_vii_dim
3791     }

```

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

```

3792     \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3793     {
3794         \dim_sub:Nn \l__enumext_item_width_vii_dim
3795         {
3796             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
3797             / \l__enumext_columns_vii_int
3798         }
3799         \dim_add:Nn \l__enumext_columns_sep_vii_dim
3800         {
3801             \l__enumext_rightmargin_vii_dim
3802         }
3803     }
3804 }

```

Same implementation for the `keyans*` environment.

```

3805 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3806 {
3807     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3808     {
3809         \dim_set:Nn \l__enumext_columns_sep_viii_dim

```

```

3810         {
3811             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3812             / \l__enumext_columns_viii_int
3813         }
3814     }
3815     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3816     \dim_set:Nn \l__enumext_item_width_viii_dim
3817     {
3818         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3819         / \l__enumext_columns_viii_int
3820         - \l__enumext_labelwidth_viii_dim
3821         - \l__enumext_labelsep_viii_dim
3822     }
3823     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3824     {
3825         \dim_sub:Nn \l__enumext_item_width_viii_dim
3826         {
3827             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3828             / \l__enumext_columns_viii_int
3829         }
3830         \dim_add:Nn \l__enumext_columns_sep_viii_dim
3831         {
3832             \l__enumext_rightmargin_viii_dim
3833         }
3834     }
3835 }

```

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

11.42.2 Functions for join item columns

`__enumext_starred_joined_item_vii:n`
`__enumext_starred_joined_item_viii:n`

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

```

3836 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3837 {
3838     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3839     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3840     {
3841         \msg_warning:nnee { enumext } { item-joined }
3842         { \int_use:N \l__enumext_joined_item_vii_int }
3843         { \int_use:N \l__enumext_columns_vii_int }
3844         \int_set:Nn \l__enumext_joined_item_vii_int
3845         {
3846             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3847         }
3848     }
3849     \int_compare:nNnT
3850     { \l__enumext_joined_item_vii_int }
3851     >
3852     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3853     {
3854         \msg_warning:nnee { enumext } { item-joined-columns }
3855         { \int_use:N \l__enumext_joined_item_vii_int }
3856         {
3857             \int_eval:n
3858             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3859         }
3860         \int_set:Nn \l__enumext_joined_item_vii_int
3861         {
3862             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3863         }
3864     }
3865     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3866     {
3867         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3868         \int_decr:N \l__enumext_joined_item_aux_vii_int
3869         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3870         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3871         \dim_set:Nn \l__enumext_joined_width_vii_dim
3872         {
3873             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int

```

```

3874         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3875         + \l__enumext_columns_sep_vii_dim
3876         )*\l__enumext_joined_item_aux_vii_int
3877     }
3878     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3879 }
3880 {
3881     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3882     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3883 }
3884 }

```

Same implementation for the `keyans*` environment.

```

3885 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3886 {
3887     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3888     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3889     {
3890         \msg_warning:nnee { enumext } { item-joined }
3891         { \int_use:N \l__enumext_joined_item_viii_int }
3892         { \int_use:N \l__enumext_columns_viii_int }
3893         \int_set:Nn \l__enumext_joined_item_viii_int
3894         {
3895             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3896         }
3897     }
3898     \int_compare:nNnT
3899     { \l__enumext_joined_item_viii_int }
3900     >
3901     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3902     {
3903         \msg_warning:nnee { enumext } { item-joined-columns }
3904         { \int_use:N \l__enumext_joined_item_viii_int }
3905         {
3906             \int_eval:n
3907             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3908         }
3909         \int_set:Nn \l__enumext_joined_item_viii_int
3910         {
3911             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3912         }
3913     }
3914     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3915     {
3916         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3917         \int_decr:N \l__enumext_joined_item_aux_viii_int
3918         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3919         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3920         \dim_set:Nn \l__enumext_joined_width_viii_dim
3921         {
3922             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3923             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3924             + \l__enumext_columns_sep_viii_dim
3925             )*\l__enumext_joined_item_aux_viii_int
3926         }
3927         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3928     }
3929     {
3930         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3931         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3932     }
3933 }

```

(End of definition for `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n`)

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

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

```

3934 \cs_new_protected:Nn \__enumext_start_mini_vii:

```



```

3935 {
3936   \dim_compare:nNt { \__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3937   {
3938     \dim_set:Nn \__enumext_minipage_left_vii_dim
3939     {
3940       \linewidth
3941       - \__enumext_minipage_right_vii_dim
3942       - \__enumext_minipage_hsep_vii_dim
3943     }
3944     \bool_set_true:N \__enumext_minipage_active_vii_bool
3945     \dim_gset_eq:NN
3946       \g__enumext_minipage_right_vii_dim
3947       \__enumext_minipage_right_vii_dim
3948     \__enumext_mini_addvspace_vii:
3949     \nointerlineskip\noindent
3950     \begin{__enumext_mini_env*}{ \__enumext_minipage_left_vii_dim }
3951   }
3952 }

```

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”.

```

3953 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3954 {
3955   \bool_if:NT \__enumext_minipage_active_vii_bool
3956   {
3957     \end{__enumext_mini_env*}
3958     \hfill
3959     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3960   }
3961 }

```

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.

```

3962 \__enumext_after_env:nn {enumext*}
3963 {
3964   \bool_if:NT \g__enumext_minipage_active_vii_bool
3965   {
3966     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3967     \par\addvspace { \g__enumext_minipage_right_skip }
3968     \bool_if:NF \g__enumext_minipage_center_vii_bool
3969     {
3970       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3971       {
3972         \centering
3973       }
3974     }
3975     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3976     {
3977       \tl_use:N \g__enumext_miniright_code_vii_tl
3978     }
3979     \box_use_drop:N \l__enumext_miniright_code_vii_box
3980     \end{__enumext_mini_env*}
3981     \par\addvspace{ \g__enumext_minipage_after_skip }
3982   }
3983   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3984   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3985   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3986   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3987   \bool_gset_false:N \g__enumext_starred_bool
3988 }

```

(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.

```

3989 \cs_new_protected:Nn \__enumext_start_mini_viii:
3990 {

```

```

3991 \dim_compare:nNt { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3992 {
3993   \dim_set:Nn \l__enumext_minipage_left_viii_dim
3994   {
3995     \linewidth
3996     - \l__enumext_minipage_right_viii_dim
3997     - \l__enumext_minipage_hsep_viii_dim
3998   }
3999   \bool_set_true:N \l__enumext_minipage_active_viii_bool
4000   \dim_gset_eq:NN
4001     \g__enumext_minipage_right_viii_dim
4002     \l__enumext_minipage_right_viii_dim
4003   \__enumext_mini_addvspace_viii:
4004   \nointerlineskip\noindent
4005   \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
4006   }
4007 }
4008 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4009 {
4010   \bool_if:NT \l__enumext_minipage_active_viii_bool
4011   {
4012     \end{__enumext_mini_env*}
4013     \hfill
4014     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4015   }
4016 }
4017 \__enumext_after_env:nn {keyans*}
4018 {
4019   \bool_if:NT \g__enumext_minipage_active_viii_bool
4020   {
4021     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
4022     \par\addvspace { \g__enumext_minipage_right_skip }
4023     \bool_if:NF \g__enumext_minipage_center_viii_bool
4024     {
4025       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4026       {
4027         \centering
4028       }
4029     }
4030     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4031     {
4032       \tl_use:N \g__enumext_miniright_code_viii_tl
4033     }
4034     \box_use_drop:N \l__enumext_miniright_code_viii_box
4035     \end{__enumext_mini_env*}
4036     \par\addvspace{ \g__enumext_minipage_after_skip }
4037   }
4038   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4039   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4040   \tl_gclear:N \g__enumext_miniright_code_viii_tl
4041   \dim_gzero:N \g__enumext_minipage_right_viii_dim
4042 }

```

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

11.43 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to \noindent and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later.

```

4043 \NewDocumentEnvironment{enumext*}{ o }
4044 {
4045   \__enumext_safe_exec_vii:
4046   \__enumext_parse_keys_vii:n {#1}
4047   \__enumext_before_list_vii:
4048   \__enumext_start_store_level_vii:
4049   \__enumext_start_list:nn { }
4050   {
4051     \__enumext_list_arg_two_vii:
4052     \__enumext_before_keys_exec_vii:
4053   }
4054   \__enumext_starred_columns_set_vii:

```

```

4055     \item[] \scan_stop:
4056     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4057     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4058   }
4059   {
4060     \__enumext_stop_item_tmp_vii:
4061     \__enumext_remove_extra_parsep_vii:
4062     \__enumext_stop_list:
4063     \__enumext_stop_store_level_vii:
4064     \__enumext_after_list_vii:
4065   }

```

(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.

```

4066 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4067 {
4068   \__enumext_internal_mini_page:
4069   \__enumext_is_not_nested:
4070   \int_incr:N \l__enumext_level_h_int
4071   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4072   {
4073     \msg_error:nn { enumext } { nested }
4074   }
4075   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4076   {
4077     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4078   }
4079   \bool_set_true:N \l__enumext_starred_bool
4080   \bool_set_false:N \l__enumext_standar_bool
4081   \__enumext_is_on_first_level:
4082 }

```

(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`.

```

4083 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4084 {
4085   \tl_if_novalue:nF {#1}
4086   {
4087     \str_clear:N \l__enumext_series_str
4088     \keys_set:nn { enumext / enumext* } {#1}
4089     \__enumext_parse_series:n {#1}
4090     \__enumext_store_active_keys_vii:n {#1}
4091     \__enumext_nested_base_line_fix:
4092   }
4093 }

```

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

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

```

4094 \cs_new_protected:Nn \__enumext_before_list_vii:
4095 {
4096   \__enumext_vspace_above_vii:
4097   \__enumext_check_ans_active:
4098   \__enumext_before_args_exec_vii:
4099   \__enumext_start_mini_vii:
4100 }

```

(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.

```
4101 \cs_new_protected:Nn \__enumext_after_list_vii:
4102 {
4103   \__enumext_stop_mini_vii:
4104   \__enumext_after_stop_list_vii:
4105   \__enumext_check_ans_key_hook:
4106   \__enumext_vspace_below_vii:
4107   \bool_set_false:N \l__enumext_starred_bool
4108   \__enumext_resume_save_counter:
4109 }
```

(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 `\sequence` of the `\anskey` command and `anskey*` environment if `enumext*` are nested in `enumext`.

`__enumext_stop_store_level_vii:`

```
4110 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4111 {
4112   \bool_if:NT \l__enumext_store_active_bool
4113   {
4114     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4115     {
4116       \__enumext_store_level_open_vii:
4117     }
4118   }
4119 }
4120 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4121 {
4122   \bool_if:NT \l__enumext_store_active_bool
4123   {
4124     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4125     {
4126       \__enumext_store_level_close_vii:
4127     }
4128   }
4129 }
```

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

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

`__enumext_start_item_tmp_vii:`

First we will call the function `__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_vii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```
4130 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4131 {
4132   \__enumext_stop_item_tmp_vii:
4133   \int_incr:N \l__enumext_item_column_pos_vii_int
4134   \int_gincr:N \g__enumext_item_count_all_vii_int
4135   \__enumext_item_peek_args_vii:
4136 }
```

(End of definition for `__enumext_start_item_tmp_vii:`)

`__enumext_item_peek_args_vii:`

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

```
4137 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4138 {
4139   \peek_meaning:NTF (
4140     { \__enumext_joined_item_vii:w }
```

```

4141     { \__enumext_joined_item_vii:w (1) }
4142 }

```

(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.

```

4143 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4144 {
4145   \__enumext_starred_joined_item_vii:n {#1}
4146   \peek_meaning_remove:NTF *
4147   { \__enumext_starred_item_vii:w }
4148   { \__enumext_standar_item_vii:w }
4149 }

```

(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].

```

4150 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4151 {
4152   \bool_set_false:N \l__enumext_item_starred_vii_bool
4153   \peek_meaning:NTF [
4154   {
4155     \bool_set_eq:NN
4156     \l__enumext_wrap_label_vii_bool
4157     \l__enumext_wrap_label_opt_vii_bool
4158     \__enumext_start_item_vii:w
4159   }
4160   {
4161     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4162     \legacy_if_set_true:n { @noitemarg }
4163     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4164   }
4165 }

```

(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

```

4166 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4167 {
4168   \bool_set_true:N \l__enumext_item_starred_vii_bool
4169   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4170   \peek_meaning:NTF [
4171   { \__enumext_starred_item_vii_aux_i:w }
4172   { \__enumext_starred_item_vii_aux_ii:w }
4173   }
4174 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4175 {
4176   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4177   \__enumext_starred_item_vii_aux_ii:w
4178 }
4179 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4180 {
4181   \peek_meaning:NTF [
4182   { \__enumext_starred_item_vii_aux_iii:w }
4183   {
4184     \dim_set_eq:NN
4185     \l__enumext_item_symbol_sep_vii_dim
4186     \l__enumext_labelsep_vii_dim
4187     \legacy_if_set_true:n { @noitemarg }
4188     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]

```

```

4189     }
4190   }
4191   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4192   {
4193     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4194     \legacy_if_set_true:n { @noitemarg }
4195     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4196   }

```

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

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

`__enumext_start_item_vii:w`

The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment. The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to `__enumext_stop_item_vii:` which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

4197   \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4198   {
4199     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4200     \legacy_if:nT { @noitemarg }
4201     {
4202       \legacy_if_set_false:n { @noitemarg }
4203       \legacy_if:nT { @nmbrrlist }
4204       {
4205         \bool_if:NT \l__enumext_hyperref_bool
4206         {
4207           \legacy_if_set_true:n { @hyper@item }
4208         }
4209         \refstepcounter{enumXvii}
4210         \bool_if:NT \l__enumext_check_answers_bool
4211         {
4212           \int_gincr:N \g__enumext_item_number_int
4213           \bool_set_true:N \l__enumext_item_number_bool
4214         }
4215       }
4216     }

```

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

```

4217   \group_begin:
4218   \lrbox{ \l__enumext_item_text_vii_box }
4219   \bool_if:NF \l__enumext_footnotes_key_bool
4220   {
4221     \__enumext_renew_footnote:
4222   }
4223   \bool_if:NT \l__enumext_item_starred_vii_bool
4224   {
4225     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4226     {
4227       \tl_gset_eq:NN
4228       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4229     }
4230     \mode_leave_vertical:
4231     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4232     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4233     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4234     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4235   }
4236   \group_begin:
4237   \tl_use:N \l__enumext_label_font_style_vii_tl
4238   \bool_if:NTF \l__enumext_wrap_label_vii_bool
4239   {

```

```

4240         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4241         { \l__enumext_wrapper_label_vii:n {#1} }
4242     }
4243     {
4244         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4245     }
4246     \group_end:
4247     \skip_horizontal:N \l__enumext_labelsep_vii_dim
4248     \tl_use:N \l__enumext_after_list_args_vii_tl
4249     \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4250     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4251     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4252     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4253 }

```

(End of definition for `\l__enumext_start_item_vii:w`.)

`\l__enumext_stop_item_vii:` The function `\l__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.

```

4254 \cs_new_protected_nopar:Nn \l__enumext_stop_item_vii:
4255 {
4256     \l__enumext_endminipage:
4257     \endlrbox
4258     \group_end:
4259     \box_set_wd:Nn \l__enumext_item_text_vii_box
4260     {
4261         \l__enumext_joined_width_vii_dim
4262         + \l__enumext_labelwidth_vii_dim
4263         + \l__enumext_labelsep_vii_dim
4264     }
4265     \int_set:Nn \hbadness { 10000 }
4266     \box_use_drop:N \l__enumext_item_text_vii_box
4267     \bool_if:NF \l__enumext_footnotes_key_bool
4268     {
4269         \l__enumext_print_footnote:
4270     }
4271     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4272     {
4273         \par\noindent
4274         \int_zero:N \l__enumext_item_column_pos_vii_int
4275     }
4276     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4277 }

```

(End of definition for `\l__enumext_stop_item_vii:`.)

`\l__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.

```

4278 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
4279 {
4280     \int_compare:nNnT
4281     {
4282         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4283     }
4284     =
4285     { 0 }
4286     {
4287         \par
4288         \vspace{ -\l__enumext_itemsep_vii_skip }
4289         \int_gzero:N \g__enumext_item_count_all_vii_int
4290     }
4291 }

```

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 `\l__enumext_after_env:nn`.

```

4292 \l__enumext_after_env:nn {enumext*} { \l__enumext_execute_after_env: }

```

(End of definition for `\l__enumext_remove_extra_parsep_vii:`.)

11.44 The environment keyans*

keyans*

First we will generate the environment and we will give a temporary definition to `__enumext_stop_item_tmp_viii`: equal to `\noindent` and next to `\item` equal to `__enumext_start_item_tmp_viii`: which we will redefine later.

```

4293 \NewDocumentEnvironment{keyans*}{ o }
4294 {
4295   \__enumext_safe_exec_viii:
4296   \__enumext_parse_keys_viii:n {#1}
4297   \__enumext_before_list_viii:
4298   \__enumext_start_list:nn { }
4299   {
4300     \__enumext_list_arg_two_viii:
4301     \__enumext_before_keys_exec_viii:
4302   }
4303   \__enumext_starred_columns_set_viii:
4304   \item[] \scan_stop:
4305   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4306   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4307 }
4308 {
4309   \__enumext_stop_item_tmp_viii:
4310   \__enumext_remove_extra_parsep_viii:
4311   \__enumext_check_starred_cmd:n { item }
4312   \__enumext_stop_list:
4313   \__enumext_after_list_viii:
4314 }
```

(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.

```

4315 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4316 {
4317   \int_incr:N \__enumext_keyans_level_h_int
4318   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4319   {
4320     \msg_error:nn { enumext } { nested }
4321   }
4322   \__enumext_keyans_name_and_start:
4323   \bool_if:NT \__enumext_starred_bool
4324   {
4325     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4326   }
4327   \bool_set_true:N \__enumext_starred_bool
4328   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4329   \bool_set_false:N \__enumext_store_active_bool
4330   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4331   {
4332     \msg_error:nn { enumext } { keyans-wrong-level }
4333   }
4334 }
```

(End of definition for __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n

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

```

4335 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4336 {
4337   \tl_if_novalue:nF {#1}
4338   {
4339     \keys_set:nn { enumext / keyans* } {#1}
4340   }
4341 }
```

(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`.

```

4342 \cs_new_protected:Nn \__enumext_before_list_viii:
4343 {
4344   \__enumext_vspace_above_viii:
```

```

4345     \__enumext_before_args_exec_viii:
4346     \__enumext_start_mini_viii:
4347 }

```

(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.

```

4348 \cs_new_protected:Nn \__enumext_after_list_viii:
4349 {
4350     \__enumext_stop_mini_viii:
4351     \__enumext_after_stop_list_viii:
4352     \__enumext_vspace_below_viii:
4353 }

```

(End of definition for __enumext_after_list_viii:.)

11.44.1 The command \item in keyans*

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

__enumext_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 __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`.

```

4354 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4355 {
4356     \__enumext_stop_item_tmp_viii:
4357     \int_incr:N \__enumext_item_column_pos_viii_int
4358     \int_gincr:N \g__enumext_item_count_all_viii_int
4359     \__enumext_item_peek_args_viii:
4360 }

```

(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).

```

4361 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4362 {
4363     \peek_meaning:NTF (
4364     { \__enumext_joined_item_viii:w }
4365     { \__enumext_joined_item_viii:w (1) }
4366 }

```

(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.

```

4367 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4368 {
4369     \__enumext_starred_joined_item_viii:n {#1}
4370     \peek_meaning_remove:NTF *
4371     { \__enumext_starred_item_viii:w }
4372     { \__enumext_standar_item_viii:w }
4373 }

```

(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 \\enumext_wrap_label_opt_viii_bool equal to the state of the variable \\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 \\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].

```
4374 \\cs_new_protected:Npn \\enumext_standar_item_viii:w
4375 {
4376   \\bool_set_false:N \\enumext_item_starred_viii_bool
4377   \\peek_meaning:NTF [
4378     {
4379       \\bool_set_eq:NN
4380       \\enumext_wrap_label_viii_bool
4381       \\enumext_wrap_label_opt_viii_bool
4382       \\enumext_start_item_viii:w
4383     }
4384     {
4385       \\bool_set_true:N \\enumext_wrap_label_viii_bool
4386       \\legacy_if_set_true:n { @noitemarg }
4387       \\enumext_start_item_viii:w [ \\enumext_label_viii_tl ]
4388     }
4389   }
```

(End of definition for \\enumext_standar_item_viii:w.)

\\enumext_starred_item_viii:w

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

\\enumext_starred_item_viii_aux_i:w

\\enumext_starred_item_viii_aux_ii:w

```
4390 \\cs_new_protected:Npn \\enumext_starred_item_viii:w
4391 {
4392   \\bool_set_true:N \\enumext_item_starred_viii_bool
4393   \\bool_set_true:N \\enumext_wrap_label_viii_bool
4394   \\peek_meaning:NTF [
4395     { \\enumext_starred_item_viii_aux_i:w }
4396     { \\enumext_starred_item_viii_aux_ii:w }
4397   }
```

The function \\enumext_starred_item_viii_aux_i:w will save the optional argument to \\item* in \\enumext_store_current_opt_arg_tl and will save this argument along with the spacing set by the key `save-sep` in variable \\enumext_store_current_label_tl if present, then call the function \\enumext_starred_item_viii_aux_ii:w.

```
4398 \\cs_new_protected:Npn \\enumext_starred_item_viii_aux_i:w [#1]
4399 {
4400   \\tl_clear:N \\enumext_store_current_label_tl
4401   \\tl_if_no_value:nF { #1 }
4402   {
4403     \\tl_if_empty:NF \\enumext_store_keyans_item_opt_sep_tl
4404     {
4405       \\tl_put_right:Ne \\enumext_store_current_label_tl { \\enumext_store_keyans_item_opt_sep_tl }
4406       \\tl_put_right:Ne \\enumext_store_current_label_tl { #1 }
4407     }
4408     \\tl_set:Ne \\enumext_store_current_opt_arg_tl { #1 }
4409   }
4410   \\enumext_starred_item_viii_aux_ii:w
4411 }
4412 \\cs_new_protected:Npn \\enumext_starred_item_viii_aux_ii:w
4413 {
4414   \\legacy_if_set_true:n { @noitemarg }
4415   \\enumext_start_item_viii:w [ \\enumext_label_viii_tl ]
4416 }
```

(End of definition for \\enumext_starred_item_viii:w, \\enumext_starred_item_viii_aux_i:w, and \\enumext_starred_item_viii_aux_ii:w.)

\\enumext_starred_item_exec:

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

```
4417 \\cs_new_protected:Nn \\enumext_starred_item_exec:
```

```

4418 {
4419   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4420   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4421   \__enumext_keyans_store_ref:
4422   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4423   \__enumext_keyans_addto_seq_link:
4424   \int_gincr:N \g__enumext_check_starred_cmd_int
4425   \bool_if:NT \l__enumext_show_answer_bool
4426   {
4427     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4428   }
4429   \bool_if:NT \l__enumext_show_position_bool
4430   {
4431     \tl_set:Ne \l__enumext_mark_answer_sym_tl
4432     {
4433       \group_begin:
4434       \exp_not:N \normalfont
4435       \exp_not:N \footnotesize [ \int_eval:n
4436       {
4437         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4438       }
4439       ]
4440       \group_end:
4441     }
4442     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4443   }
4444 }

```

(End of definition for `__enumext_starred_item_exec:`)

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

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

```

4445 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4446 {
4447   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4448   \legacy_if:nT { @noitemarg }
4449   {
4450     \legacy_if_set_false:n { @noitemarg }
4451     \legacy_if:nT { @nmbrrlist }
4452     {
4453       \bool_if:NT \l__enumext_hyperref_bool
4454       {
4455         \legacy_if_set_true:n { @hyper@item }
4456       }
4457       \refstepcounter{enumXviii}
4458     }
4459   }

```

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

```

4460   \group_begin:
4461   \lrbox{ \l__enumext_item_text_viii_box }
4462   \bool_if:NF \l__enumext_footnotes_key_bool
4463   {
4464     \__enumext_renew_footnote:
4465   }
4466   \bool_if:NT \l__enumext_item_starred_viii_bool
4467   {
4468     \__enumext_starred_item_exec:
4469   }
4470   \group_begin:
4471   \tl_use:N \l__enumext_label_font_style_viii_tl
4472   \bool_if:NTF \l__enumext_wrap_label_viii_bool
4473   {
4474     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4475     { \__enumext_wrapper_label_viii:n {#1} }
4476   }
4477   {
4478     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
4479   }
4480   \group_end:
4481   \skip_horizontal:N \l__enumext_labelsep_viii_dim

```

```

4482 \tl_use:N \l__enumext_after_list_args_viii_tl
4483 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4484 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4485 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4486 \bool_if:NT \l__enumext_item_starred_viii_bool
4487 {
4488   \tl_use:N \l__enumext_fake_item_indent_viii_tl
4489   \__enumext_keyans_show_item_opt:
4490   \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4491 }
4492 {
4493   \tl_use:N \l__enumext_fake_item_indent_viii_tl
4494 }
4495 }

```

(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.

```

4496 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4497 {
4498   \__enumext_endminipage:
4499   \endlrbox
4500   \group_end:
4501   \box_set_wd:Nn \l__enumext_item_text_viii_box
4502   {
4503     \l__enumext_joined_width_viii_dim
4504     + \l__enumext_labelwidth_viii_dim
4505     + \l__enumext_labelsep_viii_dim
4506   }
4507   \int_set:Nn \hbadness { 10000 }
4508   \box_use_drop:N \l__enumext_item_text_viii_box
4509   \bool_if:NF \l__enumext_footnotes_key_bool
4510   {
4511     \__enumext_print_footnote:
4512   }
4513   \int_compare:nNnTF
4514   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4515   {
4516     \par\noindent
4517     \int_zero:N \l__enumext_item_column_pos_viii_int
4518   }
4519   { \hspace{ \l__enumext_columns_sep_viii_dim } }
4520 }

```

(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.

```

4521 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4522 {
4523   \int_compare:nNnT
4524   {
4525     \int_mod:nn
4526     { \g__enumext_item_count_all_viii_int }
4527     { \l__enumext_columns_viii_int }
4528   }
4529   =
4530   { 0 }
4531   {
4532     \par
4533     \vspace{ -\l__enumext_itemsep_viii_skip }
4534     \int_gzero:N \g__enumext_item_count_all_viii_int
4535   }
4536 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`.)

11.45 The command \getkeyans

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

```
4537 \NewDocumentCommand \getkeyans { m }
4538 {
4539   \exp_args:Ne \__enumext_getkeyans_aux:n
4540   { \tl_to_str:e { \text_expand:n {#1} } }
4541 }
```

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

```
4542 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4543 {
4544   \str_if_in:nnTF {#1} { : }
4545   {
4546     \use:e
4547     {
4548       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4549       { {##1} {##2} }
4550     }
4551     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4552   }
4553   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4554 }
```

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

`__enumext_getkeyans:nn` The internal function `__enumext_getkeyans:nn` will check for the existence of the $\langle \text{prop list} \rangle$, if it does not exist it will return an error message, then it will fetch the content specified by the second $\langle \text{argument} \rangle$ from $\langle \text{prop list} \rangle$.

```
4555 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4556 {
4557   \prop_if_exist:cF { g__enumext_#1_prop }
4558   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4559   \group_begin:
4560   \prop_item:cn { g__enumext_#1_prop }{#2}
4561   \group_end:
4562 }
```

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

11.46 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the $\langle \text{sequence} \rangle$ defined by the `save-ans` key. The first thing we will do is define a set of $\langle \text{filtered keys} \rangle$ with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default $\langle \text{keys} \rangle$ for `\printkeyans*` and will be set by `\setenumext[$\langle \text{print}^* \rangle$]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the $\langle \text{sequence} \rangle$ and will be set by `\setenumext[$\langle \text{print}^*, * \rangle$]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[$\langle \text{print}, \text{level} \rangle$]`.

```
4563 \keys_define:nn { enumext / print }
4564 {
4565   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4566               { \__enumext_filter_save_key:n {#1} }
4567               \l__enumext_print_keyans_starred_tl, % starred cmd
4568   print* .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4569   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4570               { \__enumext_filter_save_key:n {#1} }
4571               \l__enumext_print_keyans_i_tl,
4572   print-1 .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4573   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4574               { \__enumext_filter_save_key:n {#1} }
4575               \l__enumext_print_keyans_ii_tl,
4576   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4577   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4578               { \__enumext_filter_save_key:n {#1} }
```

```

4579         \l__enumext_print_keyans_iii_tl,
4580     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4581     print-4 .code:n    = \keys_precompile:neN { enumext / level-4 }
4582                     { \l__enumext_filter_save_key:n {#1} }
4583         \l__enumext_print_keyans_iv_tl,
4584     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4585     print-* .code:n    = \keys_precompile:neN { enumext / enumext* }
4586                     { \l__enumext_filter_save_key:n {#1} }
4587         \l__enumext_print_keyans_vii_tl, % starred nested
4588     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4589 }

```

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

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

```

4590 \NewDocumentCommand \printkeyans { s O{} m }
4591 {
4592     \group_begin:
4593     \tl_use:N \l__enumext_print_keyans_i_tl
4594     \tl_use:N \l__enumext_print_keyans_ii_tl
4595     \tl_use:N \l__enumext_print_keyans_iii_tl
4596     \tl_use:N \l__enumext_print_keyans_iv_tl
4597     \tl_use:N \l__enumext_print_keyans_vii_tl
4598     \l__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4599     \group_end:
4600 }

```

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

`\l__enumext_printkeyans:nnn` The internal function `\l__enumext_printkeyans:nnn` will check for the existence of the $\langle sequence \rangle$, if it does not exist it will return an error message, then it will check if not empty.

```

4601 \cs_new_protected:Npn \l__enumext_printkeyans:nnn #1 #2 #3
4602 {
4603     \seq_if_exist:cTF { g__enumext_#3_seq }
4604     {
4605         \seq_if_empty:cF { g__enumext_#3_seq }
4606         {
4607             %%\seq_show:c { g__enumext_#3_seq }

```

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

```

4608         \bool_if:nTF {#1}
4609         {
4610             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4611             {
4612                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4613             }
4614             {
4615                 \tl_use:N \l__enumext_print_keyans_starred_tl
4616                 \begin{enumext*}[#2]
4617                     \keys_set:nn { enumext / level-1 }{ base-fix }
4618                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4619                     \end{enumext*}
4620                 }
4621             }

```

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

```

4622         {
4623             \begin{enumext}[#2]
4624             \keys_set:nn { enumext / enumext* }{ base-fix }
4625             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4626             \end{enumext}
4627         }
4628     }

```

```

4629     }
4630     {
4631         \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4632     }
4633 }

```

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

11.47 The command `\setenumext`

The command `\setenumext` will be in charge of managing the $\langle keys \rangle$ 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 $\langle keys \rangle$ that complicate us.

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

```

\__enumext_filter_first_level:n
\__enumext_filter_first_level_key:n
\__enumext_filter_first_level_pair:nn
4634 \cs_new:Npn \__enumext_filter_first_level:n #1
4635 {
4636     \use:e
4637     {
4638         \keyval_parse:NNn
4639         \__enumext_filter_first_level_key:n
4640         \__enumext_filter_first_level_pair:nn {#1}
4641     }
4642 }

```

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

```

4643 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4644 {
4645     \str_case:nnF {#1}
4646     {
4647         { resume } {}
4648         { resume* } {}
4649     }
4650     { , { \exp_not:n {#1} } } }
4651 }

```

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

```

4652 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4653 {
4654     \str_case:nnF {#1}
4655     {
4656         { series } {}
4657         { resume } {}
4658         { save-ans } {}
4659     }
4660     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } } }
4661 }

```

(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 $\langle keys \rangle$ to access from `\setenumext`.

```

4662 \keys_define:nn { enumext / meta-families }
4663 {
4664     enumext-1 .code:n =
4665         {
4666             \keys_set:ne { enumext / level-1 }
4667             {
4668                 \__enumext_filter_first_level:n {#1}
4669             }
4670         } ,
4671     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4672     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4673     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4674     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4675     enumext* .code:n =
4676         {
4677             \keys_set:ne { enumext / enumext* }
4678             {
4679                 \__enumext_filter_first_level:n {#1}

```



```

4680         }
4681     },
4682     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4683     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4684     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4685     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4686     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4687     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4688     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4689     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4690 }

```

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

```

4691 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4692 {
4693     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4694     keyans*, print-1, print-2, print-3, print-4, print-, print*,
4695 }

```

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

```

4696 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4697 {
4698     \tl_if_novalue:nTF {#1}
4699     {
4700         \seq_map_inline:Nn \c__enumext_all_families_seq
4701     }
4702     {
4703         \seq_clear:N \l__enumext_setkey_tmpa_seq
4704         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4705         \int_set:Nn \l__enumext_setkey_tmpa_int
4706         {
4707             \seq_count:N \l__enumext_setkey_tmpb_seq
4708         }
4709         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4710         {
4711             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4712             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4713             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4714             {
4715                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4716             }
4717         }
4718         {
4719             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4720         }
4721         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4722         { \seq_map_inline:Nn \c__enumext_all_families_seq }
4723         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4724     }
4725     {
4726         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4727     }
4728 }

```

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

`__enumext_set_parse:n` Internal functions used by the `\setenumext` command.

```

\__enumext_set_error:nn
4729 \cs_new_protected:Npn \__enumext_set_parse:n #1
4730 {
4731     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4732     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4733     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4734     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4735     {
4736         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4737         { \tl_trim_spaces:n {#1} }
4738     }
4739     { \__enumext_set_error:nn {#1} { } }
4740 }
4741 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4742 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.48 Messages

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

```

4743 \msg_new:nnn { enumext } { package-load }
4744 {
4745   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4746 }
4747 \msg_new:nnn { enumext } { package-not-load }
4748 {
4749   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4750 }
4751 \msg_new:nnn { enumext } { package-load-foot }
4752 {
4753   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4754 }
```

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

```

4755 \msg_new:nnn { enumext } { counters }
4756 {
4757   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4758   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4759 }
```

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

```

4760 \msg_new:nnn { enumext } { unknown-choice }
4761 {
4762   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4763 }
```

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

```

4764 \msg_new:nnnn { enumext } { anskey-env-error }
4765 {
4766   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4767   'enumext' ~ package, ~ It~ is~ already~ defined.
4768 }
4769 {
4770   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4771   for ~ the ~ 'save-ans' ~ key.\\
4772 }
```

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

```

4773 \msg_new:nnn { enumext } { store-prop }
4774 {
4775   * ~ Package ~ enumext: ~ Creating ~
4776   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4777 }
4778 \msg_new:nnn { enumext } { store-seq }
4779 {
4780   * ~ Package ~ enumext: ~ Creating ~
4781   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4782 }
4783 \msg_new:nnn { enumext } { store-int }
4784 {
4785   * ~ Package ~ enumext: ~ Creating ~
4786   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4787 }
4788 \msg_new:nnn { enumext } { prop-seq-int-hook }
4789 {
4790   * ~ Package ~ enumext: ~ Elements ~ in ~
4791   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4792   * ~ Package ~ enumext: ~ Elements ~ in ~
4793   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4794   * ~ Package ~ enumext: ~ Value ~ off ~
4795   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4796 }
4797 \msg_new:nnn { enumext } { item-answer-hook }
4798 {
4799   * ~ Package ~ enumext: ~ Value ~ off ~
4800   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4801   * ~ Package ~ enumext: ~ Value ~ off ~
4802   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4803   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4804 }
```

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

```
4805 \msg_new:nnn { enumext } { invalid-key }
4806 {
4807   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4808 }
4809 \msg_new:nnn { enumext } { unknown-key-family }
4810 {
4811   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4812 }
```

Messages used in length calculation.

```
4813 \msg_new:nnn { enumext } { width-negative }
4814 {
4815   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4816   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4817 }
4818 \msg_new:nnn { enumext } { width-zero }
4819 {
4820   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4821   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4822 }
```

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

```
4823 \msg_new:nnn { enumext } { list-lengths }
4824 {
4825   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4826   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4827   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4828   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4829   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4830   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4831   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4832   \__enumext_show_length:nnn { skip } { topsep } {#1}
4833   \__enumext_show_length:nnn { skip } { parsep } {#1}
4834   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4835   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4836   ****
4837 }
```

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

```
4838 \msg_new:nnn { enumext } { list-lengths-not-nested }
4839 {
4840   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4841   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4842   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4843   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4844   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4845   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4846   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4847   \__enumext_show_length:nnn { skip } { topsep } {#1}
4848   \__enumext_show_length:nnn { skip } { parsep } {#1}
4849   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4850   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4851   ****
4852 }
```

Messages used by `ref` key.

```
4853 \msg_new:nnn { enumext } { key-ref-empty }
4854 {
4855   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4856 }
```

Messages used by `save-ans` key.

```
4857 \msg_new:nnn { enumext } { save-ans-empty }
4858 {
4859   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4860 }
4861 \msg_new:nnn { enumext } { save-ans-log }
4862 {
4863   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4864 }
4865 \msg_new:nnn { enumext } { save-ans-log-hook }
4866 {
```

```

4867     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4868   }
4869   \msg_new:nnn { enumext } { save-ans-hook }
4870   {
4871     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4872   }

```

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

```

4873   \msg_new:nnn { enumext } { need-save-ans }
4874   {
4875     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4876   }
4877   \msg_new:nnn { enumext } { items-same-answer }
4878   {
4879     *****\
4880     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4881     for ~ \c_left_brace_str #2 \c_right_brace_str\
4882     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4883     'OK', ~ all ~ items ~ with ~ answer.\
4884     *****
4885   }
4886   \msg_new:nnn { enumext } { item-greater-answer }
4887   {
4888     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4889     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4890     Items ~ > ~ Answers.
4891   }
4892   \msg_new:nnn { enumext } { item-less-answer }
4893   {
4894     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4895     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4896     Items ~ < ~ Answers.
4897   }

```

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

```

4898   \msg_new:nnn { enumext } { missing-starred }
4899   {
4900     Missing ~ '\c_backslash_str #1*' ~ #2.
4901   }
4902   \msg_new:nnn { enumext } { many-starred }
4903   {
4904     Many ~ '\c_backslash_str #1*' ~ #2.
4905   }

```

Messages used by `\printkeyans*` command.

```

4906   \msg_new:nnn { enumext } { print-starred }
4907   {
4908     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4909     #2 ~ environment ~ \msg_line_context:.
4910   }

```

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

```

4911   \msg_new:nnn { enumext } { list-too-deep }
4912   {
4913     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
4914     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4915   }

```

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

```

4916   \msg_new:nnn { enumext } { anskey-unnumber-item }
4917   {
4918     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
4919   }
4920   \msg_new:nnn { enumext } { anskey-already-stored }
4921   {
4922     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
4923   }
4924   \msg_new:nnn { enumext } { anskey-empty-arg }
4925   {
4926     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
4927   }
4928   \msg_new:nnn { enumext } { anskey-wrong-place }
4929   {

```

```

4930     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4931     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4932 }
4933 \msg_new:nnn { enumext } { anskey-nested }
4934 {
4935     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4936 }
4937 \msg_new:nnn { enumext } { anskey-math-mode }
4938 {
4939     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
4940 }
4941 \msg_new:nnn { enumext } { anskey-env-wrong }
4942 {
4943     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4944 }
4945 \msg_new:nnn { enumext } { ansPIC-wrong-place }
4946 {
4947     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4948     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4949 }
4950 \msg_new:nnn { enumext } { command-wrong-place }
4951 {
4952     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4953     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4954 }
4955 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
4956 {
4957     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
4958     'anskey*' ~ and ~ is ~ being ~ ignored.
4959 }
4960 {
4961     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4962     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4963 }
4964 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
4965 {
4966     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
4967     'anskey*' ~ and ~ is ~ being ~ ignored.
4968 }
4969 {
4970     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4971     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4972 }
4973 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
4974 { The ~ key ~ '#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
4975 {
4976     The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4977     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4978 }
4979 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
4980 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
4981 {
4982     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4983     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4984 }

```

Messages used by **keyans**, **keyans*** and **keyansPIC** environment.

```

4985 \msg_new:nnn { enumext } { keyans-nested }
4986 {
4987     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4988 }
4989 \msg_new:nnn { enumext } { keyans-wrong-level }
4990 {
4991     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4992     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4993 }
4994 \msg_new:nnn { enumext } { wrong-place }
4995 {
4996     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
4997     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
4998 }
4999 \msg_new:nnn { enumext } { keyansPIC-nested }

```

```

5000 {
5001     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
5002 }
5003 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5004 {
5005     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5006     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5007 }
5008 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5009 {
5010     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5011 }
5012 \msg_new:nnnn { enumext } { keyans-unknown-key }
5013 {
5014     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5015     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5016 }
5017 {
5018     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5019     ~ have ~ a ~ key ~ called ~'#1'.\\
5020     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5021 }
5022 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5023 {
5024     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5025     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5026 }
5027 {
5028     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5029     ~ have ~ a ~ key ~ called ~'#1'.\\
5030     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5031 }

```

Message used by unknown $\langle keys \rangle$ in `enumext*`. environment.

```

5032 \msg_new:nnnn { enumext } { starred-unknown-key }
5033 {
5034     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5035     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5036 }
5037 {
5038     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5039     ~ have ~ a ~ key ~ called ~'#1'.\\
5040     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5041 }
5042 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5043 {
5044     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5045     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5046 }
5047 {
5048     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5049     ~ have ~ a ~ key ~ called ~'#1'.\\
5050     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5051 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

5052 \msg_new:nnnn { enumext } { standar-unknown-key }
5053 {
5054     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5055     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5056 }
5057 {
5058     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5059     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5060     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5061 }
5062 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5063 {
5064     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5065     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5066 }
5067 {

```

```

5068   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5069   ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5070   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5071 }

```

Messages used by `\getkeyans` command.

```

5072 \msg_new:nnn { enumext } { undefined-storage-anskey }
5073 {
5074   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5075 }

```

Messages used by `\miniright` command.

```

5076 \msg_new:nnn { enumext } { missing-miniright }
5077 {
5078   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
5079   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5080 }
5081 \msg_new:nnn { enumext } { wrong-miniright-place }
5082 {
5083   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.\
5084   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5085 }
5086 \msg_new:nnn { enumext } { wrong-miniright-use }
5087 {
5088   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.\
5089   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5090 }
5091 \msg_new:nnn { enumext } { wrong-miniright-starred }
5092 {
5093   Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5094 }

```

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

```

5095 \msg_new:nnn { enumext } { nested }
5096 {
5097   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5098 }
5099 \msg_new:nnn { enumext } { nested-horizontal }
5100 {
5101   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~
5102 }
5103 \msg_new:nnn { enumext } { item-joined }
5104 {
5105   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5106 }
5107 \msg_new:nnn { enumext } { item-joined-columns }
5108 {
5109   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5110 }

```

11.49 Finish package

Finish package implementation.

```

5111 \file_input_stop:
5112 </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>	213
<code>\+</code>	205
<code>\-</code>	205
<code>\\</code> 221, 2646, 3640, 4757, 4766, 4771, 4791, 4793, 4800, 4802, 4815, 4820, 4825, 4840, 4879, 4881, 4883, 4888, 4889, 4894, 4895, 4913, 4930, 4947, 4952, 4961, 4970, 4976, 4982, 4991, 4996, 5005, 5019, 5029, 5039, 5049, 5059, 5069, 5078, 5083, 5088	
A	
above	<u>1460</u>
above*	<u>1460</u>
<code>\addvspace</code> 1106, 1134, 1250, 1329, 1392, 1398, 1434, 1451, 3459, 3474, 3594, 3609, 3967, 3981, 4022, 4036	
after	<u>945</u>
align	<u>500</u>
<code>\Alph</code>	36, <u>41</u>
<code>\Alpha</code>	452, 567, 612, 680, 4584
<code>\alph</code>	36, <u>41</u>
<code>\alpha</code>	453, 565, 4576
<code>\anskey</code>	12, 73, 75, 2464
anskey*	13, <u>2574</u>
<code>\anspic</code>	15, 99, <u>3618</u>
<code>\anspic*</code>	67
<code>\arabic</code>	30, <u>36</u>
<code>\arabic</code>	451, 564, 611, 4568, 4572, 4588
B	
base-fix	<u>804</u>
<code>\baselineskip</code>	<u>49</u>
<code>\baselineskip</code>	821, <u>832</u>
before	<u>945</u>
before*	<u>945</u>
below	<u>1460</u>
below*	<u>1460</u>
bool commands:	
<code>\bool_gset_false:N</code> 326, 327, 328, 2750, 2752, 3983, 3987, 4038	
<code>\bool_gset_true:N</code> 234, 244, 1048, 1952, 1958, 3959, 3984, 4014, 4039	
<code>\bool_if:NTF</code> 391, 403, 420, 1414, 1482, 1496, 1509, 1520, 1531, 1542, 1553, 1564, 1612, 1629, 1634, 1642, 1669, 1707, 1712, 1719, 1723, 1745, 1750, 1758, 1765, 1796, 1804, 1897, 2095, 2105, 2184, 2208, 2215, 2239, 2339, 2361, 2401, 2414, 2418, 2468, 2487, 2511, 2565, 2576, 2665, 2702, 2766, 2799, 2814, 2889, 2900, 2904, 2923, 2936, 2978, 3012, 3047, 3181, 3243, 3253, 3285, 3290, 3393, 3443, 3457, 3465, 3522, 3579, 3592, 3600, 3620, 3955, 3964, 3968, 4010, 4019, 4023, 4112, 4122, 4205, 4210, 4219, 4223, 4238, 4267, 4323, 4425, 4429, 4453, 4462, 4466, 4472, 4486, 4509	
<code>\bool_if:nTF</code> 1435, 1452, 3034, 3163, 3201, 3641, 4608	
<code>\bool_if_p:N</code> 253, 268, 817, 818, 828, 829, 1776, 1777, 1785, 1786, 1910, 1936, 1949, 1950, 1955, 1956, 2272, 2281, 2282, 2294, 2310, 2346, 2387, 2388, 2688, 2876, 2877, 2914, 2915, 3366, 3368, 3379, 3648, 3649	
<code>\bool_lazy_all:nTF</code> 251, 266, 1908, 1934, 2270, 2279, 2292, 2308, 3364, 3377	
<code>\bool_lazy_and:nnTF</code> 230, 240, 816, 827, 1407, 1775, 1784, 1948, 1954, 2345, 2352, 2386, 2529, 2541, 2687, 2693, 2875	
<code>\bool_lazy_or:nnTF</code> 1837, 1844, 2913, 3647	
<code>\bool_new:N</code> 34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 94, 99, 100, 105, 106, 109, 130, 131, 139, 140, 145, 147, 148, 162, 174, 176	
<code>\bool_not_p:n</code> 231, 241, 2297, 2312, 2347, 2353, 2689, 2694, 3367, 3380	
<code>\bool_set_eq:NN</code> 2987, 3143, 4155, 4379	
<code>\bool_set_false:N</code> 400, 838, 1882, 1883, 1915, 1920, 1924, 1928, 1941, 2629, 3341, 3480, 3530, 3614, 3679, 3697, 4080, 4107, 4152, 4329, 4376	
<code>\bool_set_true:N</code> 258, 259, 273, 274, 382, 386, 493, 853, 1466, 1471, 1732, 1854, 1855, 2127, 2135, 2630, 2981, 2983, 3015, 3017, 3139, 3151, 3265, 3340, 3373, 3386, 3412, 3527, 3554, 3944, 3999, 4079, 4161, 4168, 4169, 4213, 4327, 4385, 4392, 4393	
box commands:	
<code>\box_dp:N</code> 1146, 1150, 1154, 1165, 1169, 1180, 1189, 1195, 1205, 1218, 1224, 1230, 1261, 1262, 1263, 1266, 1276, 1280, 1289, 1296, 1301, 1309, 1338, 1339, 1342, 1349, 1362, 1370, 1376, 1384, 3709	
<code>\box_new:N</code> 70, 169, 175	
<code>\box_set_wd:Nn</code> 4259, 4501	
<code>\box_use_drop:N</code> 3979, 4034, 4266, 4508	
<code>\box_wd:N</code> 459	
C	
<code>\c</code>	213, 214, 717, 719, 731, 733
<code>\catcode</code>	2646
<code>\cB</code>	214
<code>\cE</code>	214
<code>\centering</code>	1437, 1454, 3735, 3972, 4027
check-ans	<u>1874</u>
Document class:	
article	42
clist commands:	
<code>\clist_const:Nn</code> 181	
<code>\clist_map_function:nN</code> 3722	
<code>\clist_map_inline:Nn</code> 499, 759, 944, 959, 1040, 1476	
<code>\clist_map_inline:nn</code> 49, 60, 78, 84, 96, 108, 133, 156, 180, 527, 547, 813, 858, 879, 1054, 1582, 1821, 1888, 2074, 2092, 2124, 2267, 2808, 3068, 3080, 3120, 3230, 3233, 3260, 3272, 3275, 3295, 4732	
<code>\columnbreak</code>	74
<code>\columnbreak</code>	2349
columns	<u>1024</u>
columns-sep	<u>1024</u>
<code>\columnsep</code>	95
<code>\columnsep</code>	3437, 3576
<code>\columnseprule</code>	95
<code>\columnseprule</code>	3441, 3578
Commands provide by enumext :	
<code>\anskey</code> 28, 63, 64, 69, 70, 72–76, 82, 84, 94, 109, 118, 119, 124	
<code>\anspic*</code> 28, 67, 70, 82, 83, 99, 101, 102, 118, 119	
<code>\anspic</code> 70, 99, 101, 124	
<code>\getkeyans</code> 70, 118, 127	
<code>\item*</code> 28, 67, 70, 82, 83, 86, 89, 110, 115, 118, 119	

`\item` 86, 89, 104, 109–111, 114, 115
`\miniright` 27, 47, 54, 55, 95, 96, 127
`\printkeyans*` 118
`\printkeyans` 28, 70, 118, 119
`\setenumext` 28, 119–121, 123

Counters defined by `enumext`:

`enumXiii` 26, 35
`enumXii` 26, 35
`enumXiv` 26, 35
`enumXi` 26, 35
`enumXviii` 26, 35
`enumXvii` 26, 35, 111
`enumXvi` 26, 35
`enumXv` 26, 35

cs commands:

`\cs_generate_variant:Nn` . 186, 461, 477, 723, 739, 2176, 2181, 2257, 2582, 3220, 3724
`\cs_if_exist:NTF` 431
`\cs_if_free:NTF` 2533, 2545
`\cs_new:Nn` 199
`\cs_new:Npn` . 217, 1583, 1592, 1600, 2139, 2148, 2156, 4634, 4643, 4652
`\cs_new_eq:NN` . 353, 354, 355, 359, 360, 405, 406, 409, 410
`\cs_new_protected:Nn` . 209, 223, 249, 282, 312, 318, 324, 330, 336, 344, 362, 377, 588, 651, 703, 814, 960, 964, 968, 972, 976, 980, 984, 988, 992, 996, 1000, 1004, 1008, 1012, 1016, 1020, 1055, 1067, 1091, 1108, 1119, 1136, 1211, 1235, 1252, 1314, 1331, 1353, 1388, 1394, 1477, 1491, 1505, 1516, 1527, 1538, 1549, 1560, 1640, 1743, 1756, 1773, 1794, 1822, 1827, 1852, 1893, 1903, 1946, 1961, 1968, 1977, 1982, 1987, 1992, 2001, 2006, 2011, 2182, 2206, 2213, 2237, 2244, 2258, 2485, 2504, 2520, 2583, 2619, 2650, 2685, 2727, 2748, 2756, 2797, 2812, 2840, 2873, 2909, 2921, 2934, 3020, 3030, 3041, 3159, 3175, 3316, 3333, 3362, 3391, 3398, 3421, 3451, 3463, 3503, 3520, 3544, 3562, 3587, 3598, 3637, 3681, 3695, 3720, 3725, 3741, 3745, 3764, 3774, 3805, 3934, 3953, 3989, 4008, 4066, 4094, 4101, 4110, 4120, 4137, 4278, 4315, 4342, 4348, 4361, 4417, 4521
`\cs_new_protected:Npn` . 187, 191, 195, 413, 429, 446, 456, 462, 568, 613, 685, 710, 724, 1424, 1443, 1608, 1627, 1697, 1730, 1832, 2016, 2093, 2103, 2125, 2133, 2168, 2177, 2335, 2398, 2412, 2450, 2454, 2574, 2605, 2609, 2640, 2776, 2850, 2894, 2974, 2993, 3081, 3085, 3099, 3103, 3121, 3125, 3135, 3147, 3189, 3223, 3263, 3344, 3540, 3690, 3836, 3885, 4083, 4143, 4150, 4166, 4174, 4179, 4191, 4335, 4367, 4374, 4390, 4398, 4412, 4542, 4555, 4601, 4729, 4741
`\cs_new_protected_nopar:Nn` . . . 4130, 4254, 4354, 4496
`\cs_new_protected_nopar:Npn` 4197, 4445
`\cs_set:Npn` 2268, 2306, 4548
`\cs_set_eq:NN` . . . 4056, 4057, 4199, 4305, 4306, 4447
`\cs_set_protected:Nn` 884, 900, 912, 924
`\cs_set_protected:Npn` . 45, 54, 71, 79, 91, 97, 126, 152, 160, 478, 500, 532, 548, 595, 740, 760, 804, 840, 863, 936, 945, 1024, 1041, 1460, 1571, 1813, 1874, 2033, 2075, 2111, 2260, 2801, 3057, 3073, 3113, 3221, 3261
`\cs_to_str:N` 448, 471
`\cs_undefine:N` 2522, 2523, 2524, 2525

D

`\d` 205
`\DeclareDocumentEnvironment` 366

dim commands:

`\dim_abs:n` 3194, 3199
`\dim_add:Nn` 3700, 3799, 3830
`\dim_compare:nNnTF` . 886, 902, 914, 926, 1426, 1445, 3191, 3196, 3202, 3208, 3210, 3212, 3403, 3426, 3548, 3566, 3692, 3776, 3792, 3807, 3823, 3936, 3991
`\dim_compare:nTF` 2371, 2715, 3322, 3509
`\dim_gset_eq:NN` 3945, 4000
`\dim_gzero:N` 2754, 3986, 4041
`\dim_new:N` . 67, 74, 75, 76, 93, 135, 168, 170, 171, 177
`\dim_set:Nn` . . 459, 854, 3010, 3194, 3199, 3201, 3204, 3205, 3209, 3211, 3214, 3215, 3217, 3318, 3406, 3429, 3505, 3550, 3568, 3727, 3778, 3785, 3809, 3816, 3871, 3920, 3938, 3993, 4193
`\dim_set_eq:NN` . 555, 602, 673, 677, 2925, 2926, 2938, 2939, 3005, 3232, 3274, 3437, 3576, 3878, 3881, 3882, 3927, 3930, 3931, 4184
`\dim_sub:Nn` 3327, 3514, 3794, 3825
`\dim_use:N` . 887, 895, 1427, 1433, 2247, 2250, 2255, 3025, 3027, 3324, 3329, 3404, 3409, 3410, 3417, 3427, 3431, 3432, 3434
`\dim_zero:N` 3266, 3441, 3578, 3701, 3702, 3703
`\dim_zero_new:N` 428
`\c_zero_dim` . 889, 903, 915, 927, 1427, 1445, 2373, 2717, 3191, 3196, 3202, 3209, 3324, 3404, 3427, 3511, 3548, 3566, 3776, 3792, 3807, 3823, 3936, 3991
`\dimeval` 2040

E

`\end` . . 1430, 1448, 2210, 2241, 3456, 3473, 3591, 3608, 3957, 3980, 4012, 4035, 4610, 4619, 4626
`\endgroup` 2646
`\endlist` 354
`\endlrbox` 4257, 4499
`\endminipage` 360
`enumext` 5, 3296

enumext internal commands:

`\l__enumext__ref_the_count_tl` 38
`\l__enumext__resume_name_tl` 59
`__enumext_add_pre_parsep:` . 48, 1065, 1067, 1067
`__enumext_after_args_exec:` . 46, 960, 972, 3309
`__enumext_after_args_exec_v:` . 976, 988, 3496
`__enumext_after_args_exec_vii:` . . 992, 1016
`__enumext_after_args_exec_viii:` 1020
`__enumext_after_env:nn` . 79, 80, 82, 97, 106, 112, 191, 191, 2660, 3483, 3962, 4017, 4292
`__enumext_after_hyperref:` . . . 34, 375, 377, 377
`__enumext_after_list:` . 96, 114, 3314, 3463, 3463
`\l__enumext_after_list_args_v_tl` 990
`\l__enumext_after_list_args_vii_tl` . 1018, 4248
`\l__enumext_after_list_args_viii_tl` . . 1022, 4482
`__enumext_after_list_v:` 3501, 3544, 3598
`__enumext_after_list_vii:` . 109, 4064, 4101, 4101
`__enumext_after_list_viii:` . . 4313, 4348, 4348
`__enumext_after_stop_list:` . . . 46, 97, 960, 968, 3477
`__enumext_after_stop_list_v:` . 976, 984, 3615
`\l__enumext_after_stop_list_v_tl` 986
`__enumext_after_stop_list_vii:` . 109, 992, 1008, 4104
`\l__enumext_after_stop_list_vii_tl` . . . 1010
`__enumext_after_stop_list_viii:` . 1012, 4351
`\l__enumext_after_stop_list_viii_tl` . . 1014
`\l__enumext_align_label_vii_str` . . 4240, 4244

```

\l__enumext_align_label_viii_str . 4474, 4478
\l__enumext_align_label_X_str . . . . . 160
\c__enumext_all_envs_clist . . 181, 499, 759, 944,
    959, 1040, 1476
\c__enumext_all_families_seq . . 121, 4691, 4700,
    4722
\l__enumext_anskey_env_bool 31, 78, 34, 259, 274,
    2576
\__enumext_anskey_env_clean_vars: . 81, 2681,
    2685, 2748
\__enumext_anskey_env_define_keys: 78, 2574,
    2583, 2654
\__enumext_anskey_env_exec: 79, 2579, 2650, 2650
\__enumext_anskey_env_make:n 63, 78, 1857, 2574,
    2574, 2582
\__enumext_anskey_env_reset_keys: 79, 80, 2619,
    2682
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n . . . . . 2574
\__enumext_anskey_env_save_keys: . . 80, 2662,
    2685, 2685
\__enumext_anskey_env_store: . . 81, 2678, 2685,
    2727
\__enumext_anskey_env_unknown:n 79, 2602, 2605
\__enumext_anskey_env_unknown:nn . 2607, 2609
\l__enumext_anskey_level_int . . 28, 2506, 2507
\__enumext_anskey_safe_inner: . 77, 2479, 2485,
    2504
\__enumext_anskey_safe_inner:n . . . . . 76
\__enumext_anskey_safe_outer: . 76, 2466, 2485,
    2485
\__enumext_anskey_show_wrap_arg:n . 74, 2398,
    2398, 2416, 2431
\__enumext_anskey_show_wrap_left:n 75, 2343,
    2412, 2412
\__enumext_anskey_unknown:n 76, 2434, 2448, 2450
\__enumext_anskey_unknown:nn . 2434, 2452, 2454
\__enumext_anskey_wrapper:n . . . . . 2037, 2410
\__enumext_at_begin_document:n . . 33, 187, 187,
    351, 357
\l__enumext_base_line_fix_bool . 808, 818, 829,
    838
\__enumext_before_args_exec: . . 45, 95, 108, 960,
    960, 3401
\__enumext_before_args_exec_v: 976, 976, 3547
\__enumext_before_args_exec_vii: . . 992, 992,
    4098
\__enumext_before_args_exec_viii: 996, 4345
\__enumext_before_env:nn 78, 191, 195, 2527, 2539,
    2551, 2652
\__enumext_before_keys_exec: 46, 960, 964, 3306
\__enumext_before_keys_exec_v: 976, 980, 3493
\__enumext_before_keys_exec_vii . . . . . 992
\__enumext_before_keys_exec_vii: . 1000, 4052
\__enumext_before_keys_exec_viii: 1004, 4301
\__enumext_before_list: . . 95, 3300, 3398, 3398
\__enumext_before_list_v: . . 3488, 3544, 3544
\__enumext_before_list_vii: . . 108, 4047, 4094,
    4094
\__enumext_before_list_viii: . . 113, 4297, 4342,
    4342
\l__enumext_before_no_starred_key_v_tl 982
\l__enumext_before_no_starred_key_vii_-
    tl . . . . . 1002
\l__enumext_before_no_starred_key_viii_-
    tl . . . . . 1006
\l__enumext_before_starred_key_v_tl . . 978
\l__enumext_before_starred_key_vii_tl . 994
\l__enumext_before_starred_key_viii_tl 998
\__enumext_calc_hspace:NNNNNNN 90, 3189, 3189,
    3220, 3225, 3267
\__enumext_check_ans_active: . 65, 95, 108, 1893,
    1893, 3402, 4097
\g__enumext_check_ans_item_tl . . . . . 84
\g__enumext_check_ans_key_bool . . 66, 139, 326,
    1952, 1958, 2766
\l__enumext_check_ans_key_bool 66, 1878, 1883,
    1949, 1955
\__enumext_check_ans_key_hook: 66, 97, 109, 1946,
    1946, 3478, 4105
\__enumext_check_ans_level: 65, 1893, 1899, 1903
\__enumext_check_ans_log: 66, 81, 1992, 1992, 2770
\__enumext_check_ans_log_msg_greater: 1992,
    1998, 2011
\__enumext_check_ans_log_msg_less: 1992, 1996,
    2001
\__enumext_check_ans_log_msg_same_ok: 1992,
    1997, 2006
\__enumext_check_ans_msg_greater: 1968, 1974,
    1987
\__enumext_check_ans_msg_less: 1968, 1972, 1977
\__enumext_check_ans_msg_same_ok: 1968, 1973,
    1982
\__enumext_check_ans_show: . . 66, 81, 1968, 1968,
    2768
\l__enumext_check_answers_bool . 63, 65, 76, 86,
    139, 1855, 1882, 1897, 2184, 2208, 2215, 2239, 2468,
    2665, 2889, 2978, 3012, 4210
\__enumext_check_starred_cmd:n 32, 67, 84, 2016,
    2016, 3499, 3676, 4311
\g__enumext_check_starred_cmd_int 139, 2019,
    2025, 2030, 3157, 3646, 4424
\l__enumext_check_start_line_env_tl . 32, 139,
    289, 297, 305, 2022, 2028, 2031
\l__enumext_columns_sep_v_dim 3566, 3568, 3576
\l__enumext_columns_sep_vii_dim . . 3776, 3778,
    3787, 3799, 3875, 4276
\l__enumext_columns_sep_viii_dim . 3807, 3809,
    3818, 3830, 3924, 4519
\l__enumext_columns_v_int 1257, 3564, 3572, 3584,
    3589
\l__enumext_columns_vii_int . . 3781, 3784, 3788,
    3797, 3839, 3843, 3846, 3852, 3858, 3862, 4271, 4282
\l__enumext_columns_viii_int . 3812, 3815, 3819,
    3828, 3888, 3892, 3895, 3901, 3907, 3911, 4514, 4527
\l__enumext_counter_i_tl . . . . . 45, 438
\l__enumext_counter_ii_tl . . . . . 45, 439
\l__enumext_counter_iii_tl . . . . . 45, 440
\l__enumext_counter_iv_tl . . . . . 45, 441
\c__enumext_counter_style_tl . . . . . 30, 50, 211
\g__enumext_counter_styles_tl . 27, 36, 67, 449,
    467
\l__enumext_counter_v_tl . . . . . 45, 442, 693
\l__enumext_counter_vi_tl . . . . . 45, 443
\l__enumext_counter_vii_tl . . . . . 45, 444, 623
\l__enumext_counter_viii_tl . . . . . 45, 445, 640
\l__enumext_current_widest_dim 27, 67, 473, 556,
    603, 674, 678

```

__enumext_default_item:n ... [2974](#), [2974](#), [3038](#)
 __enumext_define_counters:Nn [26](#), [429](#), [429](#), [438](#),
 [439](#), [440](#), [441](#), [442](#), [443](#), [444](#), [445](#)
 __enumext_endminipage: . [33](#), [357](#), [360](#), [372](#), [3737](#),
 [4256](#), [4498](#)
 \g__enumext_envir_name_tl [31](#), [34](#), [260](#), [275](#), [334](#),
 [1825](#), [1830](#), [1840](#), [1980](#), [1985](#), [1990](#), [2004](#), [2009](#), [2014](#)
 \l__enumext_envir_name_tl . [31](#), [32](#), [34](#), [229](#), [239](#),
 [288](#), [296](#), [304](#), [5015](#), [5018](#), [5025](#), [5028](#), [5035](#), [5038](#),
 [5045](#), [5048](#), [5054](#), [5058](#), [5064](#), [5068](#), [5097](#), [5101](#)
 __enumext_execute_after_env: [32](#), [33](#), [63](#), [66](#), [77](#),
 [81](#), [2756](#), [2756](#), [3483](#), [4292](#)
 __enumext_fake_item: [884](#), [884](#), [3252](#)
 \l__enumext_fake_item_indent_v_dim [903](#), [908](#)
 \l__enumext_fake_item_indent_v_tl [905](#), [3140](#),
 [3144](#), [3152](#)
 \l__enumext_fake_item_indent_vii_dim [915](#), [920](#)
 \l__enumext_fake_item_indent_vii_tl [917](#), [4252](#)
 \l__enumext_fake_item_indent_viii_dim . [927](#),
 [932](#), [4490](#)
 \l__enumext_fake_item_indent_viii_tl . . [929](#),
 [4488](#), [4493](#)
 \l__enumext_fake_item_indent_X_tl [97](#)
 __enumext_fake_item_vii: [884](#), [912](#), [3284](#)
 __enumext_fake_item_viii: [884](#), [924](#), [3289](#)
 __enumext_filter_first_level:n . . [120](#), [4634](#),
 [4634](#), [4668](#), [4679](#)
 __enumext_filter_first_level_key:n [120](#), [4634](#),
 [4639](#), [4643](#)
 __enumext_filter_first_level_pair:nn . [120](#),
 [4634](#), [4640](#), [4652](#)
 __enumext_filter_save_key:n . . [69](#), [2100](#), [2108](#),
 [2131](#), [2137](#), [2139](#), [2139](#), [4566](#), [4570](#), [4574](#), [4578](#), [4582](#),
 [4586](#)
 __enumext_filter_save_key_key:n . . [70](#), [2139](#),
 [2144](#), [2148](#)
 __enumext_filter_save_key_pair:nn [70](#), [2139](#),
 [2145](#), [2156](#)
 __enumext_filter_series:n [58](#), [1583](#), [1583](#), [1620](#),
 [1632](#), [1637](#)
 __enumext_filter_series_key:n [58](#), [1583](#), [1588](#),
 [1592](#)
 __enumext_filter_series_pair:nn . . [58](#), [1583](#),
 [1589](#), [1600](#)
 \g__enumext_footnote_arg_seq . [157](#), [3747](#), [3760](#),
 [3770](#)
 \g__enumext_footnote_int . [157](#), [3754](#), [3757](#), [3759](#),
 [3761](#)
 \g__enumext_footnote_int_seq . [157](#), [3748](#), [3761](#),
 [3766](#), [3769](#)
 __enumext_footnotes_key_bool [34](#)
 \l__enumext_footnotes_key_bool [29](#), [34](#), [111](#), [147](#),
 [386](#), [391](#), [400](#), [4219](#), [4267](#), [4462](#), [4509](#)
 __enumext_footnotetext:nn . . . [3741](#), [3741](#), [3771](#)
 __enumext_getkeyans:nn . . [118](#), [4551](#), [4555](#), [4555](#)
 __enumext_getkeyans_aux:n [118](#), [4539](#), [4542](#), [4542](#)
 \l__enumext_hyperref_bool [29](#), [34](#), [147](#), [382](#), [403](#),
 [420](#), [2388](#), [2877](#), [4205](#), [4453](#)
 __enumext_hypertarget:nn [34](#), [377](#), [405](#), [409](#), [425](#)
 __enumext_if_is_int:n [203](#)
 __enumext_if_is_int:nTF [203](#), [712](#), [726](#)
 __enumext_internal_mini_page: [34](#), [93](#), [108](#), [362](#),
 [362](#), [3335](#), [4068](#)
 __enumext_is_not_nested: [26](#), [31](#), [93](#), [108](#), [223](#), [223](#),
 [3336](#), [4069](#)
 __enumext_is_on_first_level: . [26](#), [31](#), [94](#), [108](#),
 [223](#), [249](#), [3342](#), [4081](#)
 \g__enumext_item_anskey_int [76](#), [84](#), [139](#), [321](#), [348](#),
 [349](#), [1965](#), [2337](#), [2891](#)
 __enumext_item_answer_diff: [66](#), [81](#), [1961](#), [1961](#),
 [2763](#)
 \g__enumext_item_answer_diff_int [66](#), [139](#), [322](#),
 [1963](#), [1970](#), [1994](#)
 \l__enumext_item_column_pos_vii_int [109](#), [3846](#),
 [3852](#), [3858](#), [3862](#), [3869](#), [4133](#), [4271](#), [4274](#)
 \l__enumext_item_column_pos_viii_int . . [114](#),
 [3895](#), [3901](#), [3907](#), [3911](#), [3918](#), [4357](#), [4514](#), [4517](#)
 \l__enumext_item_column_pos_X_int [160](#)
 \g__enumext_item_count_all_vii_int [109](#), [3870](#),
 [4134](#), [4282](#), [4289](#)
 \g__enumext_item_count_all_viii_int [114](#), [3919](#),
 [4358](#), [4526](#), [4534](#)
 \g__enumext_item_count_all_X_int [160](#)
 \g__enumext_item_number_bool [139](#)
 \l__enumext_item_number_bool [65](#), [145](#), [1915](#), [1920](#),
 [1924](#), [1928](#), [1941](#), [2511](#), [2565](#), [2981](#), [3015](#), [4213](#)
 \g__enumext_item_number_int . . [65](#), [139](#), [320](#), [347](#),
 [349](#), [1914](#), [1919](#), [1923](#), [1927](#), [1940](#), [1965](#), [2980](#), [3014](#),
 [4212](#)
 __enumext_item_peek_args_vii: [109](#), [4135](#), [4137](#),
 [4137](#)
 __enumext_item_peek_args_viii: . . [114](#), [4359](#),
 [4361](#), [4361](#)
 __enumext_item_star_exec: . [87](#), [2993](#), [3020](#), [3049](#)
 \l__enumext_item_starred_vii_bool [4152](#), [4168](#),
 [4223](#)
 \l__enumext_item_starred_viii_bool [4376](#), [4392](#),
 [4466](#), [4486](#)
 \l__enumext_item_starred_X_bool [160](#)
 __enumext_item_std:w . . [33](#), [86](#), [89](#), [101](#), [351](#), [355](#),
 [2984](#), [2990](#), [3018](#), [3140](#), [3144](#), [3152](#), [3713](#)
 \g__enumext_item_symbol_aux_tl . [86](#), [123](#), [2998](#),
 [3001](#), [3026](#), [3054](#)
 \g__enumext_item_symbol_aux_vii_tl [4176](#), [4225](#),
 [4228](#), [4232](#), [4234](#)
 \g__enumext_item_symbol_aux_X_tl [160](#)
 \l__enumext_item_symbol_sep_vii_dim . . [4185](#),
 [4193](#), [4231](#), [4233](#)
 \l__enumext_item_symbol_vii_tl [4228](#)
 \l__enumext_item_text_vii_box [4218](#), [4259](#), [4266](#)
 \l__enumext_item_text_viii_box [4461](#), [4501](#), [4508](#)
 \l__enumext_item_text_X_box [160](#)
 \l__enumext_item_width_vii_dim . . [3785](#), [3794](#),
 [3873](#), [3881](#), [3882](#)
 \l__enumext_item_width_viii_dim . . [3816](#), [3825](#),
 [3922](#), [3930](#), [3931](#)
 \l__enumext_item_width_X_dim [160](#)
 \l__enumext_itemindent_X_dim [71](#)
 \l__enumext_itemsep_vii_skip [4288](#)
 \l__enumext_itemsep_viii_skip [4533](#)
 \l__enumext_joined_item_aux_vii_int . . [3867](#),
 [3868](#), [3869](#), [3870](#), [3876](#)
 \l__enumext_joined_item_aux_viii_int . [3916](#),
 [3917](#), [3918](#), [3919](#), [3925](#)
 \l__enumext_joined_item_aux_X_int [160](#)
 __enumext_joined_item_vii:w . . [109](#), [110](#), [4140](#),
 [4141](#), [4143](#), [4143](#)
 \l__enumext_joined_item_vii_int . . [3838](#), [3839](#),
 [3842](#), [3844](#), [3850](#), [3855](#), [3860](#), [3865](#), [3867](#), [3873](#)

```

\__enumext_joined_item_viii:w 114, 4364, 4365,
    4367, 4367
\l__enumext_joined_item_viii_int . 3887, 3888,
    3891, 3893, 3899, 3904, 3909, 3914, 3916, 3922
\l__enumext_joined_item_X_int . . . . . 160
\l__enumext_joined_width_vii_dim . 3871, 3878,
    3881, 4249, 4261
\l__enumext_joined_width_viii_dim 3920, 3927,
    3930, 4483, 4503
\l__enumext_joined_width_X_dim . . . . . 160
\__enumext_keyans_addto_prop:n 82, 2776, 2776,
    3154, 3643
\__enumext_keyans_addto_seq:n . 83, 2850, 2850,
    3156, 3645
\__enumext_keyans_addto_seq_link: 2850, 2871,
    2873, 4423
\__enumext_keyans_anspic_code:nnn 100, 3634,
    3637, 3637
\__enumext_keyans_default_item:n . . 89, 3135,
    3135, 3171
\l__enumext_keyans_env_bool 34, 3367, 3380, 3527,
    3614
\__enumext_keyans_fake_item: . . 884, 900, 3242
\l__enumext_keyans_level_h_int . . 28, 633, 660,
    2495, 2557, 2828, 4075, 4317, 4318
\l__enumext_keyans_level_int . . 28, 1418, 2491,
    2553, 2823, 3526, 3531, 3628
\__enumext_keyans_make_label: 37, 90, 3159, 3175,
    3240
\__enumext_keyans_mini_addvspace: . 53, 1314,
    1314, 3556
\__enumext_keyans_mini_right_cmd:n 55, 1420,
    1443, 1443
\__enumext_keyans_mini_set_vskip: . 52, 1252,
    1252, 1316
\__enumext_keyans_multi_addvspace: 1108, 1119,
    3581
\__enumext_keyans_multi_set_vskip: 49, 1108,
    1108, 1121
\__enumext_keyans_multicols_start: 3544, 3560,
    3562
\__enumext_keyans_multicols_stop: 1447, 3544,
    3587, 3612
\__enumext_keyans_name_and_start: 26, 32, 282,
    282, 3528, 3688, 4322
\__enumext_keyans_parse_keys:n 3487, 3540, 3540
\l__enumext_keyans_pic_above_int . 134, 3728,
    3729, 3731
\l__enumext_keyans_pic_above_skip . 101, 134,
    3667, 3707
\__enumext_keyans_pic_arg_two: 101, 3665, 3695,
    3695
\l__enumext_keyans_pic_below_int . 134, 3728,
    3729, 3732
\l__enumext_keyans_pic_body_seq 100–102, 134,
    3632, 3672, 3736
\__enumext_keyans_pic_do:n 102, 3672, 3674, 3720,
    3720, 3724
\l__enumext_keyans_pic_level_int . . 28, 1402,
    2499, 2561, 2779, 2818, 2853, 2941, 3683, 3684
\__enumext_keyans_pic_row:n . . 102, 3722, 3725,
    3725
\__enumext_keyans_pic_safe_exec: . 101, 3661,
    3681, 3681
\__enumext_keyans_pic_skip_abs:N . 101, 3690,
    3690, 3706
\l__enumext_keyans_pic_width_dim . 134, 3727,
    3734
\__enumext_keyans_redefine_item: . . 90, 3159,
    3159, 3239
\__enumext_keyans_ref: . . . . . 41, 685, 703, 3241
\__enumext_keyans_ref:n . . . . . 40, 682, 685, 685
\__enumext_keyans_safe_exec: . 3486, 3520, 3520
\__enumext_keyans_set_item_width: . 97, 3495,
    3503, 3503
\__enumext_keyans_show_ans: . . 2894, 2902, 2921
\__enumext_keyans_show_item_opt: . 2894, 2909,
    3152, 3657, 4489
\__enumext_keyans_show_left:n . 89, 2894, 2894,
    3150, 3652
\__enumext_keyans_show_pos: . . 2894, 2906, 2934
\__enumext_keyans_starred_item:n . . 89, 3147,
    3147, 3167
\__enumext_keyans_store_ref: . . 82, 2797, 2797,
    3155, 3644, 4421
\__enumext_keyans_store_ref_aux_i: 83, 2797,
    2809, 2812
\__enumext_keyans_store_ref_aux_ii: 83, 2797,
    2838, 2840
\__enumext_keyans_unknown_keys:n . 3073, 3077,
    3081
\__enumext_keyans_unknown_keys:nn 3073, 3083,
    3085
\__enumext_keyans_wrapper_opt:n . . 2043, 2917
\l__enumext_label_copy_i_tl . . 2302, 2816, 2821,
    2826, 2831
\l__enumext_label_copy_v_tl . . . . . 2826
\l__enumext_label_copy_vi_tl . . . . . 2821
\l__enumext_label_copy_vii_tl 2277, 2288, 2319,
    2816
\l__enumext_label_copy_viii_tl . . . . . 2831
\l__enumext_label_copy_X_tl . . . . . 149
\l__enumext_label_fill_left_v_tl . . . . . 3179
\l__enumext_label_fill_left_X_tl . . . . . 97
\l__enumext_label_fill_right_v_tl . . . . . 3186
\l__enumext_label_fill_right_X_tl . . . . . 97
\l__enumext_label_font_style_v_tl 3180, 3656
\l__enumext_label_font_style_vii_tl . . 4237
\l__enumext_label_font_style_viii_tl . . 4471
\l__enumext_label_i_tl . . . . . 548
\l__enumext_label_ii_tl . . . . . 548
\l__enumext_label_iii_tl . . . . . 548
\l__enumext_label_iv_tl . . . . . 548
\__enumext_label_style:Nnn 26, 36, 462, 462, 477,
    553, 600, 671, 675
\l__enumext_label_v_tl . . 82, 83, 668, 2784, 2858,
    2928, 2968, 3149, 3153, 3490, 3651, 3653
\l__enumext_label_vi_tl . 82, 83, 668, 2781, 2855,
    3651, 3653, 3657
\l__enumext_label_vii_tl . 595, 4163, 4188, 4195
\l__enumext_label_viii_tl 595, 4387, 4415, 4419
\l__enumext_label_width_by_box . . 67, 458, 459
\__enumext_label_width_by_box:Nn 36, 456, 456,
    461, 473, 736
\l__enumext_labelsep_i_dim . . 2926, 2931, 2939,
    2971, 4427, 4442
\l__enumext_labelsep_v_dim . . . . . 3571
\l__enumext_labelsep_vii_dim . 2403, 2926, 2939,
    3780, 3790, 3874, 4186, 4247, 4263

```



```

\l__enumext_labelsep_viii_dim 3811, 3821, 3923,
    4481, 4490, 4505
\l__enumext_labelwidth_i_dim . 2925, 2931, 2938,
    2971, 4427, 4442
\l__enumext_labelwidth_v_dim . . . . . 3571
\l__enumext_labelwidth_vii_dim . . . 2403, 2925,
    2938, 3780, 3789, 3874, 4240, 4244, 4262
\l__enumext_labelwidth_viii_dim . . 3811, 3820,
    3923, 4474, 4478, 4504
\l__enumext_leftmargin_tmp_v_bool . 101, 3697
\l__enumext_leftmargin_tmp_X_bool . . . . . 71
\l__enumext_leftmargin_tmp_X_dim . . . . . 71
\l__enumext_leftmargin_X_dim . . . . . 71
\__enumext_level: 199, 199, 577, 580, 581, 590, 592,
    887, 891, 895, 962, 966, 970, 974, 1057, 1059, 1061,
    1063, 1096, 1098, 1100, 1102, 1106, 1139, 1142, 1161,
    1170, 1176, 1181, 1185, 1196, 1200, 1201, 1206, 1242,
    1246, 1427, 1433, 1480, 1482, 1484, 1487, 1494, 1496,
    1498, 1501, 2095, 2097, 2099, 2127, 2128, 2130, 2186,
    2194, 2198, 2202, 2407, 2408, 2983, 2984, 2988, 2989,
    2990, 2998, 3006, 3007, 3010, 3017, 3018, 3022, 3025,
    3027, 3045, 3046, 3047, 3050, 3053, 3303, 3305, 3324,
    3329, 3373, 3386, 3393, 3404, 3406, 3409, 3410, 3412,
    3417, 3424, 3427, 3429, 3431, 3432, 3433, 3434, 3437,
    3443, 3448, 3454, 3457, 3459, 3465
\l__enumext_level_h_int 108, 28, 232, 255, 269, 616,
    653, 1409, 1911, 1931, 2296, 2313, 2531, 2543, 3381,
    4070, 4071
\l__enumext_level_int . 93, 28, 201, 242, 254, 270,
    364, 1069, 1213, 1408, 1905, 1937, 2273, 2283, 2289,
    2295, 2303, 2311, 2318, 2530, 2542, 2758, 3255, 3337,
    3338, 3349, 3357, 3371, 3384, 3439, 3535, 3624, 4114,
    4124, 4330, 5055, 5059, 5065, 5069
\__enumext_list_arg_two_i: . . . . . 3221
\__enumext_list_arg_two_ii: . . . . . 3221
\__enumext_list_arg_two_iii: . . . . . 3221
\__enumext_list_arg_two_iv: . . . . . 3221
\__enumext_list_arg_two_v: . 90, 3221, 3492, 3698
\__enumext_list_arg_two_vii: . . . . . 3261, 4051
\__enumext_list_arg_two_viii: . . . . . 3261, 4300
\l__enumext_listoffset_v_dim . 3511, 3516, 3573
\l__enumext_listparindent_vii_dim . . . . 4250
\l__enumext_listparindent_viii_dim . . . 4484
\__enumext_log_answer_vars: . 33, 336, 344, 2765
\__enumext_log_global_vars: . 33, 336, 336, 2764
\__enumext_make_label . . . . . 3030
\__enumext_make_label: . . . . . 37, 87, 3041, 3250
\l__enumext_mark_answer_sym_tl 72, 2049, 2252,
    2420, 2943, 2956, 4431
\l__enumext_mark_position_str 123, 2053, 2054,
    2080, 2081, 2250
\l__enumext_mark_ref_sym_tl . . 2066, 2393, 2885
\__enumext_mini_addvspace: . . 51, 95, 1235, 1235,
    3414
\__enumext_mini_addvspace_vii: 54, 1388, 1388,
    3948
\__enumext_mini_addvspace_viii: 54, 1388, 1394,
    4003
\__enumext_mini_env* . . . . . 362
\__enumext_mini_right_cmd:n 55, 1422, 1424, 1424
\__enumext_mini_set_vskip: . 50, 1136, 1136, 1237
\__enumext_mini_set_vskip_vii: 53, 1331, 1331,
    1390
\__enumext_mini_set_vskip_viii: 53, 1331, 1353,
    1396
\__enumext_minipage:w 33, 357, 359, 368, 3734, 4249,
    4483
\l__enumext_minipage_active_v_bool 3554, 3579,
    3592, 3600
\g__enumext_minipage_active_vii_bool . . 106,
    3959, 3964, 3983
\l__enumext_minipage_active_vii_bool . 3944,
    3955
\g__enumext_minipage_active_viii_bool 4014,
    4019, 4038
\l__enumext_minipage_active_viii_bool 3999,
    4010
\g__enumext_minipage_active_X_bool . . . 160
\l__enumext_minipage_active_X_bool . . . . 85
\g__enumext_minipage_after_skip 85, 1335, 1347,
    3981, 4036
\l__enumext_minipage_after_skip 50, 51, 96, 85,
    1152, 1167, 1187, 1203, 1218, 1224, 1230, 1244, 1254,
    1263, 1266, 1278, 1296, 1307, 1323, 1355, 1368, 1382,
    3474, 3609
\g__enumext_minipage_center_vii_bool . 3968,
    3984
\g__enumext_minipage_center_viii_bool 4023,
    4039
\g__enumext_minipage_center_X_bool . . . 160
\l__enumext_minipage_hsep_v_dim . . . . . 3552
\l__enumext_minipage_hsep_vii_dim . . . . 3942
\l__enumext_minipage_hsep_viii_dim . . . 3997
\l__enumext_minipage_left_skip . . 50, 85, 1144,
    1159, 1178, 1193, 1240, 1250, 1255, 1261, 1270, 1287,
    1299, 1319, 1329, 1333, 1338, 1342, 1356, 1360, 1374,
    1392, 1398
\l__enumext_minipage_left_v_dim . . 3550, 3558
\l__enumext_minipage_left_vii_dim 3938, 3950
\l__enumext_minipage_left_viii_dim 3993, 4005
\l__enumext_minipage_left_X_dim . . . . . 85
\g__enumext_minipage_right_skip 85, 1334, 1339,
    1343, 3967, 4022
\l__enumext_minipage_right_skip . 50, 85, 1148,
    1163, 1183, 1198, 1256, 1262, 1274, 1292, 1303, 1357,
    1364, 1378, 1434, 1451
\l__enumext_minipage_right_v_dim . 1445, 1450,
    3548, 3552
\g__enumext_minipage_right_vii_dim 105, 3946,
    3966, 3986
\l__enumext_minipage_right_vii_dim 105, 3936,
    3941, 3947
\g__enumext_minipage_right_viii_dim . . 4001,
    4021, 4041
\l__enumext_minipage_right_viii_dim . . 3991,
    3996, 4002
\g__enumext_minipage_right_X_dim . . . . . 160
\g__enumext_minipage_right_X_skip . . . . 160
\g__enumext_minipage_stat_int 95, 85, 1439, 1456,
    3413, 3467, 3472, 3555, 3602, 3607
\l__enumext_miniright_code_vii_box 3975, 3979
\g__enumext_miniright_code_vii_tl 106, 3970,
    3977, 3985
\l__enumext_miniright_code_viii_box . . 4030,
    4034
\g__enumext_miniright_code_viii_tl 4025, 4032,
    4040
\l__enumext_miniright_code_X_box . . . . . 160
\__enumext_multi_addvspace: . 48, 96, 1091, 1091,

```

3445
 __enumext_multi_set_vskip: 48, 1055, 1055, 1093
 \l__enumext_multicols_above_ii_skip ... 1074
 \l__enumext_multicols_above_iii_skip .. 1080
 \l__enumext_multicols_above_iv_skip ... 1086
 \l__enumext_multicols_above_v_skip 1110, 1124,
 1134
 \l__enumext_multicols_above_X_skip 79
 \l__enumext_multicols_below_v_skip 1114, 1128,
 3594
 \l__enumext_multicols_below_X_skip 79
 __enumext_multicols_start: 95, 3419, 3421, 3421
 __enumext_multicols_stop: 96, 1429, 3451, 3451,
 3476
 __enumext_nested_base_line_fix: . 43, 94, 108,
 804, 814, 3353, 4091
 __enumext_newlabel:nn 29, 35, 73, 413, 413, 2329,
 2844
 \l__enumext_newlabel_arg_one_tl 29, 35, 73, 83,
 149, 2322, 2330, 2392, 2833, 2845, 2883
 \l__enumext_newlabel_arg_two_tl 29, 35, 72, 149,
 2276, 2286, 2300, 2316, 2331, 2820, 2825, 2830, 2846
 __enumext_parse_keys:n 43, 59, 3299, 3344, 3344
 __enumext_parse_keys_vii:n . 43, 59, 4046, 4083,
 4083
 __enumext_parse_keys_viii:n . 4296, 4335, 4335
 __enumext_parse_save_key:n 69, 2120, 2125, 2125
 __enumext_parse_save_key_vii:n 69, 2115, 2125,
 2133
 __enumext_parse_series:n 58, 94, 108, 1608, 1608,
 3352, 4089
 __enumext_parse_store_keys:n 94
 \l__enumext_parsep_i_skip 1072, 1074, 1216, 1264
 \l__enumext_parsep_ii_skip ... 1078, 1080, 1222
 \l__enumext_parsep_iii_skip .. 1084, 1086, 1228
 \l__enumext_parsep_vii_skip 4251
 \l__enumext_parsep_viii_skip 4485
 \l__enumext_partopsep_v_skip . 1126, 1130, 1290,
 1294, 1301, 1305, 1321, 1325
 \l__enumext_partopsep_viii_skip 1366
 __enumext_phantomsection: 34, 377, 406, 410, 426
 __enumext_print_footnote: ... 3741, 3764, 4269,
 4511
 __enumext_print_keyans_box:NN 72, 2244, 2244,
 2257, 2403, 2406, 2930, 2970, 4427, 4442
 \l__enumext_print_keyans_i_tl ... 4571, 4593
 \l__enumext_print_keyans_ii_tl ... 4575, 4594
 \l__enumext_print_keyans_iii_tl .. 4579, 4595
 \l__enumext_print_keyans_iv_tl ... 4583, 4596
 \l__enumext_print_keyans_starred_tl 118, 119,
 123, 4567, 4615
 \l__enumext_print_keyans_vii_tl 118, 4587, 4597
 \l__enumext_print_keyans_X_tl 123
 __enumext_printkeyans:nnn 119, 4598, 4601, 4601
 __enumext_redefine_item: . 87, 3030, 3030, 3249
 \l__enumext_ref_key_arg_tl 38, 50, 214, 570, 571,
 584, 615, 618, 629, 635, 646, 687, 688, 699
 \l__enumext_ref_the_count_tl . 38, 50, 577, 580,
 583, 623, 625, 628, 640, 642, 645, 693, 695, 698
 __enumext_regex_counter_style: .. 30, 38, 209,
 209, 578, 624, 641, 694
 __enumext_register_counter_style:Nn .. 446,
 446, 451, 452, 453, 454, 455
 __enumext_remove_extra_parsep_vii: .. 4061,
 4278, 4278
 __enumext_remove_extra_parsep_viii: . 4310,
 4521, 4521
 __enumext_renew_footnote: ... 3741, 3745, 4221,
 4464
 \l__enumext_renew_the_count_v_tl 696, 705, 707
 \l__enumext_renew_the_count_vii_tl 626, 655,
 657
 \l__enumext_renew_the_count_viii_tl 643, 662,
 664
 \l__enumext_renew_the_count_X_tl 50
 __enumext_rescan_anskey_env:n .. 79, 81, 2640,
 2735, 2743
 __enumext_reset_global_bool: .. 312, 315, 324
 __enumext_reset_global_int: ... 312, 314, 318
 __enumext_reset_global_tl: 312, 316, 330
 __enumext_reset_global_vars: . 32, 81, 312, 312,
 2773
 \l__enumext_resume_active_bool . 58, 59, 61, 61,
 1612, 1732
 __enumext_resume_counter: .. 60, 61, 1730, 1736,
 1743
 __enumext_resume_counter:n . 59, 61, 1701, 1706,
 1730, 1730, 1800, 1808
 __enumext_resume_counter_save_ans: .. 61, 62,
 1730, 1741, 1773
 __enumext_resume_counter_series: . 61, 1730,
 1739, 1756
 \g__enumext_resume_int ... 61, 1653, 1747, 1748
 __enumext_resume_last:n 58, 59, 1608, 1614, 1627
 \l__enumext_resume_name_tl 61, 1649, 1657, 1660,
 1676, 1684, 1687, 1733, 1734, 1762, 1769
 __enumext_resume_save_counter: .. 59, 97, 109,
 1640, 1640, 3481, 4108
 __enumext_resume_series:n . 60, 1577, 1697, 1697
 __enumext_resume_starred: . 62, 1578, 1794, 1794
 \g__enumext_resume_vii_int 61, 1680, 1752, 1753
 \l__enumext_rightmargin_vii_dim .. 3792, 3796,
 3801
 \l__enumext_rightmargin_viii_dim . 3823, 3827,
 3832
 __enumext_safe_exec: .. 34, 93, 3298, 3333, 3333
 __enumext_safe_exec_vii: . 34, 4045, 4066, 4066
 __enumext_safe_exec_viii: ... 4295, 4315, 4315
 \l__enumext_series_name_tl 61
 \l__enumext_series_str .. 59, 94, 108, 1575, 1610,
 1618, 1619, 1621, 1623, 1644, 1647, 1651, 1671, 1674,
 1678, 3348, 4087
 __enumext_set_error:nn 4729, 4739, 4741
 __enumext_set_item_width: . 93, 3308, 3316, 3316
 __enumext_set_parse:n 4712, 4729, 4729
 \l__enumext_setkey_tmpa_int ... 118, 4705, 4709
 \l__enumext_setkey_tmpa_seq .. 118, 4703, 4713,
 4719, 4721, 4723, 4736
 \l__enumext_setkey_tmpa_tl ... 118, 4711, 4715
 \l__enumext_setkey_tmpb_seq .. 118, 4704, 4707,
 4711, 4712
 \l__enumext_setkey_tmpb_tl 118, 4731, 4733, 4734
 \l__enumext_show_answer_bool . 2060, 2084, 2414,
 2900, 2914, 3648, 4425
 __enumext_show_length:nnn .. 45, 217, 217, 4826,
 4827, 4828, 4829, 4830, 4831, 4832, 4833, 4834, 4835,
 4841, 4842, 4843, 4844, 4845, 4846, 4847, 4848, 4849,
 4850

```

\l__enumext_show_position_bool ... 2063, 2087,
    2418, 2904, 2915, 3649, 4429
\g__enumext_standar_bool 31, 93, 34, 231, 234, 253,
    327, 1642, 1707, 1719, 1745, 1758, 1796, 1936, 1950,
    3368
\l__enumext_standar_bool . 93, 97, 34, 2281, 2294,
    2310, 3340, 3480, 4080
\l__enumext_standar_first_bool 31, 94, 34, 258,
    817, 1629, 1776, 1838, 1845
\__enumext_standar_item_vii:w . 110, 4148, 4150,
    4150
\__enumext_standar_item_viii:w 114, 115, 4372,
    4374, 4374
\__enumext_standar_ref: . . . . 39, 568, 588, 3251
\__enumext_standar_ref:n . . . . 38, 560, 568, 568
\g__enumext_standar_series_tl . 61, 1631, 1632,
    1798, 1801
\__enumext_standar_unknown_keys:n 3113, 3117,
    3121
\__enumext_standar_unknown_keys:nn 3113, 3123,
    3125
\g__enumext_starred_bool 31, 108, 34, 241, 244, 268,
    328, 1669, 1712, 1723, 1750, 1765, 1804, 1910, 1956,
    2272, 2282, 2312, 2814, 3987
\l__enumext_starred_bool 108, 109, 34, 1414, 2297,
    2347, 2353, 2401, 2689, 2694, 2923, 2936, 3341, 4079,
    4107, 4323, 4327
\__enumext_starred_columns_set_vii: . 3774,
    3774, 4054
\__enumext_starred_columns_set_viii: . 3774,
    3805, 4303
\l__enumext_starred_first_bool 31, 108, 34, 273,
    828, 1634, 1785, 1838, 1845
\__enumext_starred_item:nn . . 2993, 2993, 3036
\__enumext_starred_item_exec: . 115, 4417, 4417,
    4468
\__enumext_starred_item_vii:w . 110, 4147, 4166,
    4166
\__enumext_starred_item_vii_aux_i:w . 4166,
    4171, 4174
\__enumext_starred_item_vii_aux_ii:w . 4166,
    4172, 4177, 4179
\__enumext_starred_item_vii_aux_iii:w 4166,
    4182, 4191
\__enumext_starred_item_viii:w 114, 115, 4371,
    4390, 4390
\__enumext_starred_item_viii_aux_i:w . 115,
    4390, 4395, 4398
\__enumext_starred_item_viii_aux_ii:w . 115,
    4390, 4396, 4410, 4412
\__enumext_starred_joined_item_vii:n 104, 110,
    3836, 3836, 4145
\__enumext_starred_joined_item_viii:n . 104,
    114, 3836, 3885, 4369
\__enumext_starred_ref: . . . . 40, 613, 651, 3281
\__enumext_starred_ref:n . . . . 39, 607, 613, 613
\g__enumext_starred_series_tl . 61, 1636, 1637,
    1806, 1809
\__enumext_starred_unknown_keys:n 3095, 3097,
    3099
\__enumext_starred_unknown_keys:nn 3095, 3101,
    3103
\__enumext_start_from:NNn 41, 710, 710, 723, 745
\l__enumext_start_i_int . . . . 1748, 1760, 1779
\__enumext_start_item_tmp_vii: 107, 4057, 4130,
    4130
\__enumext_start_item_tmp_viii: . 113, 4306,
    4354, 4354
\__enumext_start_item_vii:w 110, 111, 4158, 4163,
    4188, 4195, 4197, 4197
\__enumext_start_item_viii:w . 115, 4382, 4387,
    4415, 4445, 4445
\g__enumext_start_line_tl 31, 34, 261, 276, 333,
    1980, 1985, 1990, 2004, 2009, 2014
\__enumext_start_list:nn . 33, 90, 101, 351, 353,
    3302, 3489, 3662, 4049, 4298
\__enumext_start_mini_vii: 108, 3934, 3934, 4099
\__enumext_start_mini_viii: . 113, 3989, 3989,
    4346
\__enumext_start_save_ans_msg: 63, 1822, 1822,
    1847
\__enumext_start_store_level: . 94, 3301, 3362,
    3362
\__enumext_start_store_level_vii: 109, 4048,
    4110, 4110
\l__enumext_start_vii_int . . 1753, 1767, 1788
\l__enumext_start_X_int . . . . . 97, 740
\__enumext_stop_item_tmp_vii: . 107, 109, 111,
    4056, 4060, 4132, 4199
\__enumext_stop_item_tmp_viii: 113, 114, 4305,
    4309, 4356, 4447
\__enumext_stop_item_vii: 111, 112, 4199, 4254,
    4254
\__enumext_stop_item_viii: 117, 4447, 4496, 4496
\__enumext_stop_list: . 33, 351, 354, 3312, 3500,
    3675, 4062, 4312
\__enumext_stop_mini_vii: 106, 109, 3934, 3953,
    4103
\__enumext_stop_mini_viii: 114, 3989, 4008, 4350
\__enumext_stop_save_ans_msg: . 63, 1822, 1827,
    2762
\__enumext_stop_store_level: . 94, 3313, 3362,
    3391
\__enumext_stop_store_level_vii: . 109, 4063,
    4110, 4120
\l__enumext_store_active_bool 28, 63, 109, 1777,
    1786, 1854, 2487, 3366, 3379, 3522, 3530, 3620, 3679,
    4112, 4122, 4329
\__enumext_store_active_keys:n 68, 69, 94, 2093,
    2093, 3359
\__enumext_store_active_keys_vii:n 68, 69, 108,
    2093, 2103, 4090
\__enumext_store_addto_prop:n 70, 82, 2168, 2168,
    2176, 2338, 2795, 4420
\__enumext_store_addto_seq:n 70, 84, 2177, 2177,
    2181, 2188, 2202, 2210, 2219, 2233, 2241, 2396, 2888
\l__enumext_store_anskey_arg_tl 28, 73, 74, 109,
    2344, 2349, 2351, 2356, 2363, 2366, 2376, 2381, 2384,
    2390, 2396
\__enumext_store_anskey_code:n 73, 76, 81, 2335,
    2335, 2480, 2733, 2741
\l__enumext_store_anskey_env_tl . 28, 80, 109,
    2663, 2667, 2673, 2735, 2743
\l__enumext_store_anskey_opt_tl . 28, 80, 109,
    2664, 2691, 2697, 2704, 2710, 2720, 2730, 2739
\__enumext_store_anskey_safe_outer: . . . . 76
\g__enumext_store_columns_break_bool . 2587,
    2688, 2750

```

`\l__enumext_store_columns_break_bool` . 2346, 2436
`\l__enumext_store_current_label_tl` 28, 82–84, 115, 109, 2778, 2781, 2784, 2791, 2793, 2795, 2852, 2855, 2858, 2864, 2869, 2879, 2888, 4400, 4405, 4406, 4419, 4420, 4422
`\l__enumext_store_current_label_tmp_tl` . 28, 109, 3149, 3153
`\l__enumext_store_current_opt_arg_tl` 28, 115, 109, 2898, 2911, 2917, 4408
`__enumext_store_internal_ref:` .. 72, 73, 2258, 2258, 2341
`\g__enumext_store_item_join_int` .. 2590, 2695, 2699, 2751
`\l__enumext_store_item_join_int` .. 2354, 2358, 2439
`\g__enumext_store_item_star_bool` . 2592, 2702, 2752
`\l__enumext_store_item_star_bool` . 2361, 2441
`\g__enumext_store_item_symbol_sep_dim` 2597, 2717, 2722, 2754
`\l__enumext_store_item_symbol_sep_dim` 2373, 2378, 2446
`\g__enumext_store_item_symbol_tl` . 2595, 2708, 2712, 2753
`\l__enumext_store_item_symbol_tl` . 2364, 2368, 2444
`\l__enumext_store_keyans_item_opt_sep_tl` 2046, 2789, 2791, 2862, 2866, 4403, 4405
`__enumext_store_level_close:` . 71, 2182, 2206, 3395
`__enumext_store_level_close_vii:` . 71, 2213, 2237, 4126
`__enumext_store_level_open:` 71, 94, 2182, 2182, 3374, 3387
`__enumext_store_level_open_vii:` .. 71, 2213, 2213, 4116
`\g__enumext_store_name_tl` 28, 63, 109, 332, 339, 340, 341, 342, 1830, 1856, 1979, 1984, 1989, 2003, 2008, 2013, 2760
`\l__enumext_store_name_tl` 28, 63, 65, 109, 1663, 1666, 1690, 1693, 1781, 1790, 1825, 1834, 1835, 1856, 1857, 1858, 1860, 1861, 1863, 1865, 1866, 1868, 1870, 1871, 1895, 2170, 2172, 2179, 2324, 2325, 2426, 2669, 2835, 2836, 2949, 2962, 4437
`\l__enumext_store_ref_key_bool` 73, 2069, 2339, 2387, 2799, 2876
`\l__enumext_store_save_key_vii_bool` .. 2105, 2135
`\l__enumext_store_save_key_vii_tl` 2107, 2108, 2136, 2137, 2217, 2225, 2229, 2233
`\l__enumext_store_save_key_X_bool` .. 68, 123
`\l__enumext_store_save_key_X_tl` .. 68, 69, 123
`\l__enumext_store_upper_level_X_bool` .. 123
`__enumext_storing_exec:` 63, 78, 1832, 1848, 1852
`__enumext_storing_set:n` .. 63, 1817, 1832, 1832
`\l__enumext_the_counter_v_tl` 695
`\l__enumext_the_counter_vii_tl` 625
`\l__enumext_the_counter_viii_tl` 642
`\l__enumext_the_counter_X_tl` 50
`__enumext_tmp:n` 45, 49, 54, 60, 71, 78, 79, 84, 91, 96, 97, 108, 126, 133, 152, 156, 160, 180, 804, 813, 1571, 1582, 1813, 1821, 1874, 1892, 2033, 2074, 2075, 2092, 2111, 2124, 2260, 2267, 2268, 2289, 2303, 2306, 2318, 2801, 2808, 3073, 3080, 3113, 3120, 3221, 3260, 3261, 3295
`__enumext_tmp:nn` 478, 499, 500, 531, 532, 547, 740, 759, 840, 862, 863, 883, 936, 944, 945, 959, 1024, 1040, 1041, 1054, 1460, 1476, 3057, 3072
`__enumext_tmp:nnn` 548, 564, 565, 566, 567, 595, 611, 612
`__enumext_tmp:nnnnnn` 760, 785, 788, 791, 793, 795, 798, 801
`__enumext_tmp:w` 4548, 4551
`\l__enumext_tmpa_vii_int` 3784, 3787, 3796, 3827
`\l__enumext_tmpa_viii_int` 3815, 3818
`\l__enumext_tmpa_X_dim` 160
`\l__enumext_tmpa_X_int` 160
`\l__enumext_topsep_v_skip` 1112, 1116, 1259, 1272, 1280, 1285, 1305, 1309, 3678, 3710
`\l__enumext_topsep_vii_skip` .. 1336, 1345, 1349
`\l__enumext_topsep_viii_skip` . 1358, 1380, 1384
`__enumext_undefine_anskey_env:` . 77, 81, 2520, 2520, 2771
`\l__enumext_vspace_a_star_v_bool` 1509
`\l__enumext_vspace_a_star_vii_bool` ... 1531
`\l__enumext_vspace_a_star_viii_bool` ... 1542
`\l__enumext_vspace_a_star_X_bool` 97
`__enumext_vspace_above:` 56, 95, 1477, 1477, 3400
`__enumext_vspace_above_v:` . 56, 1505, 1505, 3546
`\l__enumext_vspace_above_v_skip` .. 1507, 1511, 1513
`__enumext_vspace_above_vii:` 57, 108, 1527, 1527, 4096
`\l__enumext_vspace_above_vii_skip` 1529, 1533, 1535
`__enumext_vspace_above_viii:` . 57, 1527, 1538, 4344
`\l__enumext_vspace_above_viii_skip` 1540, 1544, 1546
`\l__enumext_vspace_b_star_v_bool` 1520
`\l__enumext_vspace_b_star_vii_bool` ... 1553
`\l__enumext_vspace_b_star_viii_bool` ... 1564
`\l__enumext_vspace_b_star_X_bool` 97
`__enumext_vspace_below:` 56, 97, 1491, 1491, 3479
`__enumext_vspace_below_v:` . 57, 1516, 1516, 3616
`\l__enumext_vspace_below_v_skip` .. 1518, 1522, 1524
`__enumext_vspace_below_vii:` 57, 109, 1549, 1549, 4106
`\l__enumext_vspace_below_vii_skip` 1551, 1555, 1557
`__enumext_vspace_below_viii:` . 57, 1549, 1560, 4352
`\l__enumext_vspace_below_viii_skip` 1562, 1566, 1568
`__enumext_widest_from:nnnn` .. 41, 724, 724, 739, 751
`\g__enumext_widest_label_tl` 27, 36, 67, 466, 470, 474
`\l__enumext_wrap_label_opt_v_bool` 3143
`\l__enumext_wrap_label_opt_vii_bool` 110, 4157
`\l__enumext_wrap_label_opt_viii_bool` .. 115, 4381
`\l__enumext_wrap_label_opt_X_bool` 97
`\l__enumext_wrap_label_v_bool` 3139, 3143, 3151, 3181
`\l__enumext_wrap_label_vii_bool` .. 110, 4156, 4161, 4169, 4238

\l_enumext_wrap_label_viii_bool	115, 4380, 4385, 4393, 4472
\l_enumext_wrap_label_X_bool	97
__enumext_wrapper_label_v:n	3183, 3657
__enumext_wrapper_label_vii:n	4241
__enumext_wrapper_label_viii:n	4475
\l_enumext_write_aux_file_tl	29, 73, 83, 149, 2327, 2333, 2842, 2848
__enumext_zero_parsep:	51, 1156, 1211, 1211
enumext*	5, 4043
enumXi	438
enumXii	438
enumXiii	438
enumXiv	438
enumXv	438
enumXvi	438
enumXvii	438
enumXviii	438
Environments provide by enumext:	
anskey*	28, 63, 72–75, 77, 78, 80, 81, 94, 109, 119, 122, 124
enumext*	25, 26, 29–31, 34, 35, 39–45, 47, 53, 54, 57–60, 62–65, 67–77, 80, 82, 83, 87, 88, 92–94, 102–104, 106, 109, 111, 112, 114, 116, 118–120, 123, 126, 127
enumext	25, 26, 30, 31, 34–43, 45–56, 58–60, 62–65, 67–77, 80, 82, 83, 86, 87, 89–91, 93, 94, 97, 98, 101, 103, 105, 108, 109, 118–120, 123, 124, 126
keyans*	25, 26, 28–30, 32, 35, 39–45, 47, 53, 54, 57, 63, 64, 67, 68, 70, 78, 82, 88, 92, 102, 103, 105, 113, 123, 125, 127
keyanspic	25, 26, 28, 32, 35, 37, 40, 54, 63, 64, 67, 70, 78, 82–84, 99–101, 125
keyans	25, 26, 28, 30, 32, 35–37, 40–43, 45, 47, 49, 52–57, 63, 64, 67, 68, 70, 78, 82–84, 88–91, 97–101, 105, 114, 123, 125
Environments:	
list	30, 33, 90, 93
lrbox	103, 111, 112, 116, 117
minipage	30, 33, 34, 47, 49, 99–103, 111, 112, 117
multicols	47–50, 55, 95, 96
scontents	78, 80
exp commands:	
\exp_after:wN	4551
\exp_args:Ne	2732, 2740, 3356, 4539
\exp_args:NV	2452, 2607, 3083, 3101, 3123
\exp_not:N	58, 469, 583, 628, 645, 698, 893, 907, 908, 919, 920, 931, 932, 2392, 2423, 2424, 2881, 2946, 2947, 2959, 2960, 4434, 4435, 4548
\exp_not:n	263, 278, 291, 299, 307, 522, 542, 583, 584, 628, 629, 645, 646, 698, 699, 894, 1598, 1606, 2057, 2154, 2166, 2330, 2358, 2368, 2378, 2392, 2393, 2699, 2712, 2722, 2845, 2883, 2885, 4650, 4660
F	
\fbbox	2040
\fbboxrule	2040
\fbboxsep	2040
file commands:	
\file_input_stop:	5111
first	945
font	478
\footnote	102
\footnote	102, 3749
\footnotemark	3759
\footnotesize	2424, 2947, 2960, 4435
\footnotetext	3743

G	
\getkeyans	16, 118, 4537
group commands:	
\group_begin:	2422, 2467, 2642, 2729, 2945, 2958, 4217, 4236, 4433, 4460, 4470, 4559, 4592
\group_end:	2429, 2483, 2746, 2952, 2965, 4246, 4258, 4440, 4480, 4500, 4561, 4599
H	
\hbadness	4265, 4507
hbox commands:	
\hbox_set:Nn	458
\hfill	508, 512, 517, 518, 1431, 1449, 2392, 2881, 3958, 4013
hook commands:	
\hook_gput_code:nnn	9, 189, 193, 197, 375
\hook_gremove_code:nn	80, 2658
\hook_gset_rule:nnnn	376
\hook_if_empty:nTF	2656
\hspace	4276, 4519
\hyperlink	74, 84
\hyperlink	2392, 2881
\hypertarget	34
\hypertarget	405
I	
\IfHyperBoolean	383
\IfPackageLoadedTF	11, 19, 379, 393
\ignorespaces	896
\inputlineno	263, 278, 291, 299, 307
int commands:	
\int_add:Nn	3869, 3918
\int_case:nn	1069, 1213, 1905, 1931, 1970, 1994
\int_compare:nNnTF	364, 616, 633, 653, 660, 1138, 1257, 1402, 1418, 2018, 2024, 2491, 2495, 2499, 2507, 2553, 2557, 2561, 2758, 2779, 2818, 2823, 2828, 2853, 2941, 3338, 3349, 3371, 3384, 3423, 3439, 3453, 3467, 3531, 3535, 3564, 3589, 3602, 3624, 3628, 3684, 3839, 3849, 3865, 3888, 3898, 3914, 4071, 4075, 4114, 4124, 4271, 4280, 4318, 4330, 4513, 4523, 4709
\int_compare_p:nNn	232, 242, 254, 255, 269, 270, 1408, 1409, 1911, 1937, 2273, 2283, 2295, 2296, 2311, 2313, 2354, 2530, 2531, 2542, 2543, 2695, 3381
\int_decr:N	3868, 3917
\int_eval:n	349, 2172, 2325, 2424, 2836, 2947, 2960, 3236, 3280, 3857, 3906, 4435
\int_from_alph:n	718, 732
\int_from_roman:n	720, 734
\int_gadd:Nn	3870, 3919
\int_gdecr:N	1914, 1919, 1923, 1927, 1940
\int_gincr:N	1747, 1752, 2337, 2891, 2980, 3014, 3157, 3413, 3555, 3646, 4134, 4212, 4358, 4424
\int_gset:Nn	1963, 3757
\int_gset_eq:NN	1646, 1653, 1659, 1665, 1673, 1680, 1686, 1692, 3754
\int_gzero:N	320, 321, 322, 1439, 1456, 2030, 2751, 3472, 3607, 4289, 4534
\int_if_exist:N	1621, 1657, 1663, 1684, 1690, 1868
\int_incr:N	2506, 3337, 3526, 3683, 4070, 4133, 4317, 4357
\int_mod:nn	4282, 4525
\int_new:N	28, 29, 30, 31, 32, 33, 61, 62, 85, 101, 120, 136, 137, 142, 143, 144, 146, 157, 163, 164, 165, 166, 167, 1623, 1871
\int_set:Nn	714, 718, 720, 1760, 1767, 1779, 1788, 2643, 3728, 3729, 3784, 3815, 3838, 3844, 3860, 3887, 3893, 3909, 4265, 4507, 4705

<code>\int_set_eq:NN</code>	1748, 1753, 3867, 3916
<code>\int_sign:n</code>	1965
<code>\int_step_function:nnN</code>	2289, 2303, 2318
<code>\int_step_inline:nnn</code>	3730
<code>\int_to_roman:n</code>	201, 2269, 2307
<code>\int_use:N</code>	342, 347, 348, 1139, 1762, 1769, 1781, 1790, 3236, 3255, 3280, 3357, 3424, 3433, 3448, 3454, 3842, 3843, 3855, 3891, 3892, 3904, 5055, 5059, 5065, 5069
<code>\int_zero:N</code>	4274, 4517
<code>\item</code>	85, 89, 109, 111, 114, 116, 355, 2190, 2196, 2221, 2227, 2351, 2855, 2858, 3032, 3161, 3715, 4055, 4057, 4304, 4306, 4422
<code>\item*</code>	5, 14, 67, 3159
<code>item-pos*</code>	3057
<code>item-sym*</code>	3057
<code>\itemindent</code>	91
<code>\itemindent</code>	90
<code>itemindent</code>	840
<code>\itemsep</code>	100, 101
<code>\itemsep</code>	3699, 3705
<code>\itemwidth</code>	428, 2040, 3318, 3327, 3505, 3514, 3878, 3882, 3927, 3931

K

<code>keyans</code>	14, 3484
<code>keyans*</code>	14, 4293
<code>keyanspic</code>	15, 3659
Keys for command provide by enumext :	
<code>break-col</code>	74, 75, 78–80
<code>item-join</code>	74, 75, 78–80
<code>item-pos*</code>	74, 75, 78–80
<code>item-star</code>	74, 75, 78–80
<code>item-sym*</code>	74, 75, 78–80

Keys for environments provide by [enumext](#):

<code>above*</code>	27, 56, 57, 95, 108
<code>above</code>	27, 56, 57, 95, 108, 113
<code>after</code>	45, 46, 97, 109, 114
<code>align</code>	27, 37, 87, 90, 111, 122
<code>base-fix</code>	43, 58, 70, 94, 108, 119
<code>before*</code>	45, 95, 108, 113
<code>before</code>	45, 46
<code>below*</code>	27, 56, 57, 97, 109
<code>below</code>	27, 56, 57, 97, 109, 114
<code>check-ans</code>	28, 30, 31, 62–64, 66, 67, 70, 81, 84, 97, 109, 112, 124
<code>columns-sep</code>	47, 95
<code>columns</code>	27, 47, 50, 55, 95
<code>first</code>	45, 46, 111
<code>font</code>	37, 87, 90, 111
<code>item-pos*</code>	86, 87
<code>item-sym*</code>	28, 86, 87
<code>itemindent</code>	27, 43, 44, 86, 90, 111
<code>itemsep</code>	42, 92
<code>labelsep</code>	37, 91, 111
<code>labelwidth</code>	36–41, 91
<code>label</code>	26, 27, 36, 38, 41, 103
<code>lisparindent</code>	92
<code>list-indent</code>	27, 43, 44, 101
<code>list-offset</code>	43, 44, 93, 97
<code>listparindent</code>	43, 111
<code>mark-ans</code>	67, 70, 75
<code>mark-pos</code>	67, 68, 122
<code>mark-ref</code>	67, 70, 72, 74
<code>mini-env</code>	27, 34, 47, 55, 70, 95, 105, 106, 108, 109, 113
<code>mini-right*</code>	27, 30, 47, 70, 106, 108, 109

<code>mini-right</code>	27, 30, 47, 54, 70, 106, 108, 109
<code>mini-sep</code>	27, 47, 70, 95
<code>no-store</code>	28, 62–65, 70, 76, 86
<code>noitemsep</code>	42, 51
<code>nosep</code>	42, 51
<code>parindent</code>	92
<code>parsep</code>	42, 92, 111
<code>partopsep</code>	42
<code>ref</code>	26, 30, 38–40, 123
<code>resume*</code>	26, 58, 59, 62, 63, 70, 97, 109, 120
<code>resume</code>	26, 33, 58–63, 70, 97, 109, 120
<code>rightmargin</code>	43, 103
<code>save-ans</code>	28, 33, 58–63, 65, 66, 68–70, 76–78, 81–83, 89, 98–100, 114, 115, 118, 120, 123
<code>save-key</code>	28, 58, 69, 94, 108
<code>save-pos</code>	70
<code>save-ref</code>	29, 35, 67, 70, 72–74, 82, 84, 89, 115
<code>save-sep</code>	67, 70, 115
<code>series</code>	26, 58–62, 70, 94, 97, 108, 109, 120
<code>show-ans</code>	67, 68, 70, 72, 73, 75, 89, 115
<code>show-length</code>	30, 45, 123
<code>show-pos</code>	28, 67, 68, 72, 73, 75, 84, 89, 115
<code>start</code>	27, 30, 41, 58
<code>store-key</code>	68
<code>topsep</code>	42
<code>widest</code>	27, 30, 41
<code>wrap-ans</code>	35, 67, 70, 72, 74
<code>wrap-label*</code>	37, 86, 87, 90, 110, 111, 115
<code>wrap-label</code>	37, 86, 87, 90, 110, 111, 115
<code>wrap-opt</code>	67, 70

keys commands:

<code>\keys_define:nn</code>	480, 502, 534, 550, 597, 668, 742, 762, 806, 842, 865, 938, 947, 1026, 1043, 1462, 1573, 1815, 1876, 2035, 2077, 2113, 2118, 2434, 2585, 2621, 3059, 3075, 3095, 3115, 4563, 4662
<code>\l_keys_key_str</code>	76, 79, 2452, 2607, 3083, 3101, 3123, 4811
<code>\keys_precompile:nnN</code>	119, 186, 186, 4565, 4569, 4573, 4577, 4581, 4585
<code>\keys_set:nn</code>	494, 822, 833, 1049, 1467, 1472, 1709, 1714, 1801, 1809, 2472, 3351, 3356, 3542, 4088, 4339, 4617, 4624, 4666, 4671, 4672, 4673, 4674, 4677, 4682, 4683, 4684, 4685, 4686, 4687, 4688, 4726
<code>\keys_set_known:nn</code>	2739

keyval commands:

<code>\keyval_parse:NNn</code>	1587, 2143, 4638
--------------------------------	------------------

L

<code>label</code>	548, 595, 668
Labels provide by enumext :	
<code>\Alph*</code>	36
<code>\Roman*</code>	36
<code>\alph*</code>	36
<code>\arabic*</code>	30, 36
<code>\roman*</code>	36
<code>\labelsep</code>	101
<code>\labelsep</code>	3700, 3703
<code>labelsep</code>	478
<code>\labelwidth</code>	36, 101
<code>\labelwidth</code>	3700, 3701
<code>labelwidth</code>	478
<code>\leftmargin</code>	91
<code>\leftmargin</code>	90, 3700
legacy commands:	
<code>\legacy_if:nTF</code>	4200, 4203, 4448, 4451

\legacy_if_gset_false:n 369
\legacy_if_set_false:n 4202, 4450
\legacy_if_set_true:n 4162, 4187, 4194, 4207, 4386, 4414, 4455
\linewidth 95
\linewidth 3320, 3408, 3507, 3552, 3727, 3787, 3818, 3940, 3995
\list 353
list-indent 840
list-offset 840
\listparindent 3702
listparindent 840
\lrbbox 4218, 4461

M

\makebox 103
\makebox .. 2248, 2250, 3026, 4232, 4240, 4244, 4474, 4478
\makelabel 85, 87, 90, 102
\makelabel 85, 89, 3043, 3177
\makesavenoteenv 399
mark-ans 2033
mark-pos 2033, 2075
mark-ref 2033
mini-env 1024
mini-sep 1024
\minipage 359
\miniright 10, 54, 1400, 3470, 3605

mode commands:

\mode_if_math:TF 2515, 2569
\mode_if_vertical:TF 1094, 1122, 1238, 1317
\mode_leave_vertical: 820, 831, 893, 907, 919, 931, 2246, 3024, 4230

msg commands:

\msg_error:nn .. 2476, 2509, 2513, 2567, 2675, 3533, 3537, 3626, 3686, 3717, 4073, 4320, 4332, 4689
\msg_error:nnn 573, 620, 637, 690, 1404, 1411, 1416, 1441, 1458, 1721, 1725, 1840, 2458, 2517, 2535, 2547, 2555, 2559, 2563, 2571, 2613, 3089, 3107, 3129, 4077, 4325, 4553, 4558, 4631, 4742
\msg_error:nnnn 2461, 2489, 2493, 2497, 2501, 2616, 3092, 3110, 3132, 3524, 3622, 3630, 4612
\msg_error:nnnnn 521, 541, 2056
\msg_fatal:nn 3339
\msg_fatal:nnn 432
\msg_info:nnn 13, 16, 21, 24, 381, 395
\msg_line_context: .. 4776, 4781, 4786, 4815, 4820, 4825, 4840, 4855, 4859, 4863, 4867, 4871, 4875, 4882, 4889, 4895, 4909, 4913, 4918, 4922, 4926, 4930, 4935, 4939, 4943, 4947, 4952, 4987, 4991, 4996, 5001, 5005, 5010, 5074, 5078, 5083, 5088, 5093, 5097, 5101, 5105, 5109
\msg_log:nnn 1860, 1865, 1870
\msg_log:nnnnn 346, 2003, 2008, 2013
\msg_log:nnnnnn 338
\msg_new:nnn 4743, 4747, 4751, 4755, 4760, 4773, 4778, 4783, 4788, 4797, 4805, 4809, 4813, 4818, 4823, 4838, 4853, 4857, 4861, 4865, 4869, 4873, 4877, 4886, 4892, 4898, 4902, 4906, 4911, 4916, 4920, 4924, 4928, 4933, 4937, 4941, 4945, 4950, 4985, 4989, 4994, 4999, 5003, 5008, 5072, 5076, 5081, 5086, 5091, 5095, 5099, 5103, 5107
\msg_new:nnnn .. 4764, 4955, 4964, 4973, 4979, 5012, 5022, 5032, 5042, 5052, 5062
\msg_term:nnnn . 1824, 1829, 3245, 3255, 3286, 3291
\msg_term:nnnnn 1984

\msg_warning:nn 3469, 3604
\msg_warning:nnnn 2021, 2027, 3193, 3198, 3841, 3854, 3890, 3903
\msg_warning:nnnnn 1979, 1989
\multicolsep 95
\multicolsep 3438, 3577

N

\NeedsTeXFormat 3
\newcounter 435
\NewDocumentCommand 1400, 2464, 3618, 4537, 4590, 4696
\NewDocumentEnvironment . 3296, 3484, 3659, 4043, 4293
\newenvsc 2578
\newlabel 35
\newlabel 417
no-store 1874
\noindent . 3415, 3557, 3949, 4004, 4056, 4273, 4305, 4516
\nointerlineskip 3415, 3557, 3949, 4004
noitemsep 760
\nopagebreak 1105, 1133, 1249, 1328, 1391, 1397
\normalfont 2423, 2946, 2959, 4434
nosep 760

P

Packages:

caption 106
enumext 25, 35, 38, 62, 91, 99, 122
enumitem 35, 36
expl3 102
footnotehyper 34
hyperref 29, 30, 34, 35, 74, 84, 111, 122
lua-visual-debug 49
multicol 25, 122
scontents 25, 77, 78
shortlst 102
\par .. 1105, 1133, 1249, 1328, 1391, 1397, 1434, 1451, 2400, 3459, 3474, 3594, 3609, 3739, 3967, 3981, 4022, 4036, 4273, 4287, 4516, 4532
\parbox 2040
\parindent 4250, 4484
\parsep 48, 51, 100, 101
\parsep 3277, 3699, 3706, 3711
parsep 760
\parskip 4251, 4485
\partopsep 101
\partopsep 3278, 3704
partopsep 760
peek commands:
 \peek_meaning:NTF 4139, 4153, 4170, 4181, 4363, 4377, 4394
 \peek_meaning_remove:NTF 4146, 4370
 \peek_remove_spaces:n 3165
\phantomsection 34
\phantomsection 406
prg commands:
 \prg_do_nothing: 410
 \prg_new_protected_conditional:Npnn ... 203
 \prg_replicate:nn 220
 \prg_return_false: 207
 \prg_return_true: 206
\printkeyans 16, 118, 4590
prop commands:
 \prop_count:N 340, 2172, 2325, 2426, 2836, 2949, 2962, 4437
 \prop_gput_if_not_in:Nnn 2170
 \prop_if_exist:NTF 1858, 4557

\prop_item:Nn	4560
\prop_new:N	1861
\ProvidesExplPackage	4
R	
\raggedcolumns	3447, 3583
\ref	72, 82
ref	548, 595, 668
\refstepcounter	4209, 4457
regex commands:	
\regex_match:nnTF	205, 717, 719, 731, 733, 2671
\regex_replace_once:nnN	213
\renewcommand	583, 628, 645, 698
\RenewDocumentCommand	3032, 3043, 3161, 3177, 3715, 3749
\RequirePackage	17, 25
resume	1571
resume*	1571
rightmargin	840
\Roman	36, 41
\Roman	454
\roman	36, 41
\roman	455, 566, 4580
S	
\s	2672
save-ans	1813
save-key	2111
save-ref	2033
save-sep	2033
scan commands:	
\scan_stop:	101, 3713, 4055, 4304, 4548, 4551
scontents internal commands:	
\l_scontents_fname_out_tl	2631
__scontents_parse_environment_keys:n	2637
__scontents_rescan_tokens:n	2644
\l_scontents_storing_bool	2629
\l_scontents_writing_bool	2630
seq commands:	
\seq_clear:N	4703
\seq_const_from_clist:Nn	4691
\seq_count:N	341, 3672, 4707
\seq_gclear:N	3747, 3748
\seq_gput_right:Nn	2179, 3760, 3761
\seq_if_empty:NTF	3766, 4605, 4721
\seq_if_exist:NTF	1863, 4603
\seq_if_in:NnTF	4610
\seq_item:Nn	2669, 3736
\seq_map_function:NN	4712
\seq_map_inline:Nn	4618, 4625, 4700, 4722, 4723
\seq_map_pairwise_function:NNN	3768
\seq_new:N	121, 122, 134, 158, 159, 1866
\seq_pop_left:NN	4711
\seq_put_right:Nn	3632, 4719, 4736
\seq_set_from_clist:Nn	4704
\seq_set_map_e:NNn	4713
\seq_show:N	4607
series	1571
\setcounter	728, 732, 734, 3236, 3280, 3677
\setenumext	6, 120, 4696
show-ans	2033, 2075
show-length	936
show-pos	2075
skip commands:	
\skip_add:Nn	1074, 1080, 1086, 1096, 1100, 1124, 1128, 1218, 1224, 1230, 1240, 1244, 1266, 1319, 1323, 3699

\skip_gset:Nn	1339, 1343, 1347
\skip_gzero_new:N	1334, 1335
\skip_horizontal:N	908, 920, 932, 4233, 4247, 4481
\skip_horizontal:n	894, 2247, 2255, 3025, 3027, 4231, 4490
\skip_if_eq:nnTF	1072, 1078, 1084, 1141, 1175, 1216, 1222, 1228, 1259, 1264, 1285, 1336, 1358, 1479, 1493, 1507, 1518, 1529, 1540, 1551, 1562
\skip_new:N	81, 82, 86, 87, 88, 89, 90, 138, 178
\skip_set:Nn	1057, 1061, 1110, 1114, 1144, 1148, 1152, 1159, 1163, 1167, 1178, 1183, 1187, 1193, 1198, 1203, 1261, 1262, 1263, 1270, 1274, 1278, 1287, 1292, 1296, 1299, 1303, 1307, 1338, 1342, 1360, 1364, 1368, 1374, 1378, 1382, 3693, 3707
\skip_set_eq:NN	3234, 3276, 3277, 4250, 4251, 4484, 4485
\skip_use:N	1059, 1063, 1098, 1102, 1106, 1126, 1130, 1142, 1161, 1170, 1176, 1181, 1185, 1196, 1200, 1201, 1206, 1242, 1246, 1272, 1480, 1484, 1487, 1494, 1498, 1501, 3459
\skip_zero:N	3278, 3438, 3577, 3704, 3705
\skip_zero_new:N	1254, 1255, 1256, 1333, 1355, 1356, 1357
\c_zero_skip	1072, 1078, 1084, 1142, 1176, 1216, 1222, 1228, 1259, 1264, 1285, 1336, 1358, 1480, 1494, 1507, 1518, 1529, 1540, 1551, 1562
\small	4568, 4572, 4576, 4580, 4584, 4588
\star	3063
start	740
\stepcounter	3639, 3753
str commands:	
\c_backslash_str	2517, 4776, 4781, 4786, 4791, 4793, 4795, 4800, 4802, 4900, 4904, 4908, 4918, 4922, 4930, 4931, 4935, 4947, 4948, 4952, 4953, 4974, 4976, 4980, 4982, 5010, 5078, 5079, 5083, 5088, 5089, 5093
\c_colon_str	2324, 2835, 4548
\c_left_brace_str	4881, 4888, 4894
\c_right_brace_str	4881, 4888, 4894
\str_case:nn	225, 284
\str_case:nnTF	1594, 1602, 2150, 2158, 4645, 4654
\str_clear:N	3348, 4087
\str_count:n	220
\str_if_empty:NTF	1610, 1651, 1678
\str_if_eq:nnTF	3237, 3282
\str_if_in:nnTF	4544
\str_new:N	124, 173
\str_set:Nn	537, 538, 539, 2053, 2054, 2080, 2081
\string	399
\strutbox	1146, 1150, 1154, 1165, 1169, 1180, 1189, 1195, 1205, 1218, 1224, 1230, 1261, 1262, 1263, 1266, 1276, 1280, 1289, 1296, 1301, 1309, 1338, 1339, 1342, 1349, 1362, 1370, 1376, 1384, 3709

T	
T _E X and L ^A T _E X 2 _ε commands:	
\@auxout	415
\@currentvir	225, 284
\protected@write	415
tex commands:	
\tex_newlinechar:D	2643
text commands:	
\text_expand:n	4540
\textasteriskcentered	2050, 2067
\thepage	421

tl commands:

`\c_space_tl` 2917, 4825, 4840, 4863, 4867, 5054, 5055, 5064, 5065, 5097, 5101

`\tl_clear:N` . . . 507, 513, 2031, 2097, 2107, 2128, 2136, 2344, 2663, 2664, 2778, 2852, 4400

`\tl_clear_new:N` 464

`\tl_const:Nn` 50, 448

`\tl_gclear:N` . . . 332, 333, 334, 1631, 1636, 2753, 3054, 3985, 4040, 4234

`\tl_gclear_new:N` 1618

`\tl_gput_right:Nn` 449

`\tl_greplace_all:Nnn` 470

`\tl_gset:Nn` 260, 261, 275, 276, 1619, 1632, 1637, 1856, 2667, 3001, 4176

`\tl_gset_eq:NN` 466, 2997, 4227

`\tl_if_blank:nTF` 2456, 2474, 2611, 3087, 3105, 3127, 4225

`\tl_if_empty:NTF` . . 571, 590, 618, 635, 655, 662, 688, 705, 1644, 1649, 1671, 1676, 1734, 1798, 1806, 1835, 1895, 2186, 2217, 2364, 2708, 2730, 2760, 2789, 2862, 2911, 3022, 4403, 4734

`\tl_if_empty:nTF` 1699

`\tl_if_exist:NTF` 1704

`\tl_if_novalue:nTF` . . 2470, 2786, 2860, 2896, 2976, 2995, 3003, 3137, 3346, 3670, 3751, 4085, 4337, 4401, 4698

`\tl_map_inline:Nn` 211, 467

`\tl_new:N` 42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 102, 103, 104, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 123, 125, 128, 129, 141, 149, 150, 151, 154, 172

`\tl_put_left::Ne` 2697

`\tl_put_left:Nn` 2194, 2225, 2349, 2691, 2704, 2710, 2720, 2928, 2968, 3970, 4025, 4419, 4422

`\tl_put_right:Nn` 465, 581, 626, 643, 696, 2198, 2229, 2276, 2286, 2300, 2316, 2322, 2327, 2351, 2356, 2363, 2366, 2376, 2381, 2384, 2390, 2781, 2784, 2791, 2793, 2820, 2825, 2830, 2833, 2842, 2855, 2858, 2864, 2869, 2879, 4405, 4406

`\tl_remove_all:Nn` 4733

`\tl_remove_once:Nn` 2264, 2805

`\tl_replace_all:Nnn` 469

`\tl_reverse:N` 2263, 2265, 2804, 2806

`\tl_set:Nn` . . 58, 229, 239, 288, 289, 296, 297, 304, 305,

434, 508, 512, 517, 518, 570, 615, 687, 891, 905, 917, 929, 1733, 1834, 2098, 2108, 2129, 2137, 2420, 2631, 2898, 2943, 2956, 4408, 4431, 4731

`\tl_set_eq:NN` 475, 576, 579, 623, 625, 640, 642, 693, 695, 2262, 2803, 2816, 3149, 3153, 3651, 3653

`\tl_to_str:n` 1704, 1710, 1715, 4540

`\tl_trim_spaces:n` 465, 4719, 4731, 4737

`\tl_use:N` . . 471, 474, 592, 657, 664, 707, 962, 966, 970, 974, 978, 982, 986, 990, 994, 998, 1002, 1006, 1010, 1014, 1018, 1022, 2252, 2269, 2277, 2288, 2302, 2307, 2319, 2984, 2990, 3018, 3045, 3046, 3053, 3140, 3144, 3152, 3179, 3180, 3186, 3303, 3490, 3656, 3977, 4032, 4237, 4248, 4252, 4471, 4482, 4488, 4493, 4593, 4594, 4595, 4596, 4597, 4615, 4715

token commands:

`\token_to_str:N` 417

`topsep` 760

`\typeout` 385, 388, 398, 399

U

`\u` 214, 2672

`unknown` 3073, 3095, 3113

use commands:

`\use:N` 221, 3050, 3305

`\use:n` 1585, 2141, 4546, 4636

`\use_none:nn` 409

`\usecounter` 3235, 3279

V

`\value` 1647, 1653, 1660, 1666, 1674, 1680, 1687, 1693

vbox commands:

`\vbox_set_top:Nn` 3975, 4030

`\vspace` . . 370, 821, 832, 1484, 1487, 1498, 1501, 1511, 1513, 1522, 1524, 1533, 1535, 1544, 1546, 1555, 1557, 1566, 1568, 3667, 3678, 4288, 4533

W

`widest` 740

`wrap-ans` 2033

`wrap-label` 478

`wrap-label*` 478

`wrap-opt` 2033

Z

`\z` 2672