

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-07\*

©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	<b>Introduction</b>	1	5	<b>The storage system</b>	10
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for <code>\label</code> and <code>\ref</code>	11
1.3	User interface	3	5.1.2	Keys for <code>\wrap</code> and <code>\display</code>	11
1.3.1	Internal counters	3	5.1.3	Keys for debug and checking	11
1.3.2	Support for <code>multicol</code>	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>minipage</code>	3	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	13
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
2	<b>The environments provided</b>	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	16
3	<b>The command <code>\setenumext</code></b>	6	6	<b>Full examples</b>	17
4	<b>The <code>keyval</code> system</b>	6	7	<b>The way of non-enumerated lists</b>	19
4.1	Keys for <code>\label</code> and <code>\ref</code>	6	8	<b>References</b>	21
4.2	Keys for spaces	7	9	<b>Change history</b>	22
4.2.1	Vertical spaces	7	10	<b>Index of Documentation</b>	23
4.2.2	Horizontal spaces	8	11	<b>Implementation</b>	25
4.3	Keys for <code>\addcode</code>	8	12	<b>Index of Implementation</b>	122
4.4	Keys for <code>\start</code> , <code>\series</code> and <code>\resume</code>	9			
4.5	Keys for <code>multicols</code>	9			
4.6	Keys for <code>minipage</code>	9			
4.6.1	The command <code>\miniright</code>	9			
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  $\text{\LaTeX}$  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`, `\parsec` - 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 2024-06-07.

†E-mail: [pablgonz@educarchile.cl](mailto:pablgonz@educarchile.cl).

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.  
The enumext package loads and requires multicol[3] and scontents[4] packages, need to have a modern T<sub>E</sub>X distribution such as T<sub>E</sub>X Live or MiK<sub>T</sub>E<sub>X</sub>. It has been tested with the standard classes provided by L<sup>A</sup>T<sub>E</sub>X: book, report, article and letter on 10pt, 11pt and 12pt.

1 Introduction

In the L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>Xze 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<sup>A</sup>T<sub>E</sub>Xze is cool?

\* Very True!

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

\* Yes

(b) Usually uses the package manager?

\* Yes, dnf

(c) Rate the following package and class

i. xsim-exam

\* doesn't exist for now :(

ii. xsim

\* very good

iii. exsheets

\* obsolete

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

1.  $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II.  $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

★ 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

E) value

4. Question with image and label below:

A

B

A

A) B) C)

A

duck

D) E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B
- Where what we are interested in the <label> and a “short note” that we leave as an explanation, and then print them:
- ©2024 by Pablo González L
- 2 / 135

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  $\text{\LaTeX}$ , 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  $\text{\TeX}$  Live and  $\text{\MiKTeX}$ , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

enumext.sty

enumext.pdf

README.md

enumext.dtx

>>

>>

>>

>>

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

TDS:source/latex/enumext/

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

### 1.3 User interface

The user interface consists in `enumext`, `enumext*`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments, `\anskey`, `\item*` and `\anspic*` commands to *stored content*, `\getkeyans` command to get the individual *stored content*, `\printkeyans` to print all *stored content*, `\miniright` for `minipage` and `\setenumext` to config all [`<key = val>`] options.

#### 1.3.1 Internal counters

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

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

#### 1.3.2 Support for multicols

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



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

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §4.5).

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

#### 1.3.3 Support for minipage

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



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

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

#### 1.3.4 The \label and \ref system

This package provides a user interface like the `enumitem`[6] package to customize the references which is activated by the `ref` key (§4.1), the standard  $\TeX$  `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§5.1.1) when the key `save-ans` (§5.1) is active.

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

#### 1.3.5 Support for \footnote

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

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

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

## 2 The environments provided

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

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

### 2.1 The environment `enumext`

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

Example with `columns=2`

1. This text is in the first level.
- A. This text is in the 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*` environment 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 value set by the `labelsep` key and can be `\langle offset \rangle` using the second optional argument. The default values for `\langle symbol \rangle` and `\langle offset \rangle` are `\$ \star \$` and the value set by `labelsep` key.

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

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

### 2.3.1 Keys for `\item*`

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

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

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

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

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

## 3 The command `\setenumext`

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

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

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

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

## 4 The keyval system

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

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

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

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



## 4.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 4.2 Keys for spaces

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

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

### 4.2.1 Vertical spaces

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

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

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

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

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

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

- The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T<sub>E</sub>X 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.

- 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<sub>E</sub>X 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 “discardable”.

`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 discardable”.

`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 “discardable”.

`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 discardable”.

### 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` = {*<rigid length>*} 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` = {*<rigid length>*} 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` = {*<rigid length>*} 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 *<label>* 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 *<keys>* should be used with “*caution*”, they are intended to inject *<code>* 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  $\text{\LaTeX}$  which is defined (simplified) as plain form `\list{<arg one>}{<arg two>}`. Using the `before*` key does not allow access to the `list` parameters defined by `[<key = val>]`.

`before` = {*<code>*} default: not used  
 Execute *<code>* “*before*” the environment starts. The *<code>* must be passed between braces, is executed “*after*” performing all calculations related to the *list parameters* in the environment and the parameters sets by `[<key = val>]` that is, in the second argument of the list after setting all the parameters `\list{<arg one>}{<arg two>}{<code>}`.

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

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

- Keep in mind that the code set in this key will affect the entire “*body*” of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.

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

### 4.4 Keys for start, series and resume

`start` = {*<integer | string>*} default: 1  
 Sets the *start value* of the numbering on the current level. Internally *<string>* 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 *<keys>* are “*only*” available for the “*first level*” of `enumext` and `enumext*` and are ignored if set when nested inside each other.

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

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

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

- For security reasons the `series` key will never save in *<series name>* the keys `series`, `resume`, `resume*`, `save-ans`, `save-key` and `start`. When using the key `resume={<series name>}` it will have hierarchy in the *<keys>* that are saved in *<series name>*, in order to establish the value of a *<key>* already saved in *<series name>* it must be placed to the

“right” of `resume={⟨series name⟩}`, the same thing happens with the `resume*` key, the exception is the `save-ans` key that must be placed on the “left” if you want to start the numbering with its value. The `resume` key passed “without value” must be exactly “without value”, i.e. `resume=` cannot be used and if executed before `resume*` it will affect the start value.

## 4.5 Keys for multicol

`columns = {⟨integer⟩}`

default: 1

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

`columns-sep = {⟨rigid length⟩}`

default: by level

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

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

## 4.6 Keys for minipage

`mini-env = {⟨rigid length⟩}`

default: not used

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

`mini-sep = {⟨rigid length⟩}`

default: 0.3333em

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

### 4.6.1 The command \miniright

---

`\miniright`  
`\miniright*`

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<sup>A</sup>T<sub>E</sub>X 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.

### 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 = {⟨code for drawing or tabular⟩}`

default: not used

Set the *code* for the drawing or tabular to be placed in the `minipage` environment on the “right side” by starting it with `\centering`.

`mini-right* = {⟨code for drawing or tabular⟩}`

default: not used

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

## 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 `⟨key⟩` 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 `⟨labels⟩` for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be stored in a `⟨sequence⟩` and at the same time will be stored (without the environment structure or optional arguments) in a `⟨prop list⟩`.

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

## 5.1 Keys for storage system

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

`save-ans = {⟨store name⟩}` default: *not set*

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

`save-key = {⟨key list⟩}` default: *not set*

This key *overrides* the default “stored keys” of the optional arguments of the inner levels or nested environment that will be passed to the *sequence*. The *key list* passed to this key ignores any *keys* 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 *keys* that will be stored by default in the *sequence* would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the *sequence* only the key `columns=3` ignoring all the others.

`save-sep = {⟨text symbol⟩}` default: `{,}`

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

### 5.1.1 Keys for label and ref

`save-ref = {⟨true | false⟩}` default: *false*

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

`mark-ref = {⟨symbol⟩}` default: `\textasteriskcentered`

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

### 5.1.2 Keys for wrap and display

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

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

`wrap-opt = {⟨code {#1} more code⟩}` default: `[{#1}]`

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

`show-ans = {⟨true | false⟩}` default: *false*

Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the *label* for `\item*` and `\anspic*` at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

`mark-ans = {⟨symbol⟩}` default: `\textasteriskcentered`

Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

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

### 5.1.3 Keys for debug and checking

`show-pos = {⟨true | false⟩}` default: *false*  
 Displays the *position* occupied by the “*stored content*” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* {⟨*store name*⟩} set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans = {⟨true | false⟩}` default: *false*  
 Enables the *checking answer* mechanism displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “*only one answer*” or “*only one execution*” of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.

`no-store` *⟨value forbidden⟩* default: *not used*  
 This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* {⟨*store name*⟩} set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “*without*” using the `\anskey`, “*without*” use `anskey*`, “*without*” interfering with the `check-ans` key and “*without*” storing an unwanted structure in the *sequence* {⟨*store name*⟩}.

## 5.2 The command `\anskey`

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

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

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

### Example

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

- ★ 1. Text containing our instructions or questions.

\* 

first answer

2. Text containing our instructions or questions.

(a) Question.

\* 

second answer
3. Text containing our instructions or questions.

\* 

third answer

4. Text containing our instructions or questions.

\* 

fourth answer

5.3 The environment anskey\*

`anskey*` `\begin{anskey*}[(key = val)] <body content> \end{anskey*}`

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

By design the environment cannot be nested but full supports “*verbatim material*” in the body and it is assumed that each `\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 `<keys>` must be passed separated by commas and “without separation” of the start of the environment. Comments “%” or “any character” after `\begin{anskey*}` or `[(key = val)]` 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 `<keys>` 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(<columns>)` is available for the `keyans*` environment.



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

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

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

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

```
\item* \item*
\item* [<content>]
```

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

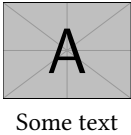
Example

```
\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}
```

1. Text containing a question.

A) Choice                      \* B) Correct choice  
C) Choice                      D) Choice  
E) Choice
2. Text containing a question and image.

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



5.5 The environment `keyanspic`

```
keyanspic \begin{keyanspic}[<n° above, n° below>]\anspic{<drawing>}\anspic* [<content>]{<drawing>}
```

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.

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





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

`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 { <drawing or tabular> }
\anspic* [ <content> ] { <drawing or tabular> }
```

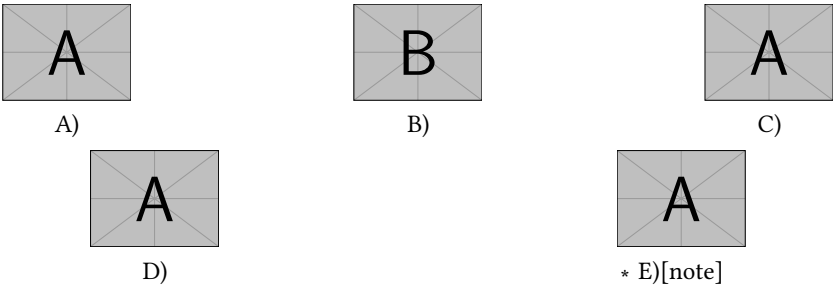
The `\anspic` command take three arguments, the *starred argument* `*` store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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.



5.6 Printing stored content

5.6.1 The command `\getkeyans`

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

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

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

### 5.6.2 The command `\printkeyans`

---

```
\printkeyans \printkeyans[⟨keys⟩]{⟨store name⟩}
\printkeyans* [⟨keys⟩]{⟨store name⟩}
```

---

The command `\printkeyans` prints “all stored content” in sequence  $\{\langle store\ name \rangle\}$  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  $\{\langle store\ name \rangle\}$  does not exist the command will return an error.

The optional argument allows managing the  $\langle keys \rangle$  in the “first level” of the environment in which the “stored content” of the *sequence*  $\{\langle store\ name \rangle\}$  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*  $\{\langle store\ name \rangle\}$  the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the *sequence*  $\{\langle store\ name \rangle\}$  it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

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

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

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[⟨print, 1⟩]{⟨keys⟩}` and `\setenumext[⟨print*⟩]{⟨keys⟩}`. If we need to set the  $\langle keys \rangle$  for the environment `enumext` “saved” in the *sequence*  $\{\langle store\ name \rangle\}$  we will use `\setenumext[⟨print, level⟩]{⟨keys⟩}` and if we need to set the  $\langle keys \rangle$  for the environment `enumext*` “saved” in the *sequence*  $\{\langle store\ name \rangle\}$  we will use `\setenumext[⟨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 \LaTeXe\ is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1]  $3(x + y + z)$

2. True False

(a) ~~ETEX~~e is cool?

[2] Very True!

3. Related to Linux

(a) You use linux?

[3] Yes

(b) Rate the following package and class

i. xsim

[4] very good

ii. exsheets

[5] obsolete

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 arara<sup>1</sup> 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.
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.
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  $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$ .
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  $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$ .

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

\*

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

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)  $\angle EDF = 45^\circ$

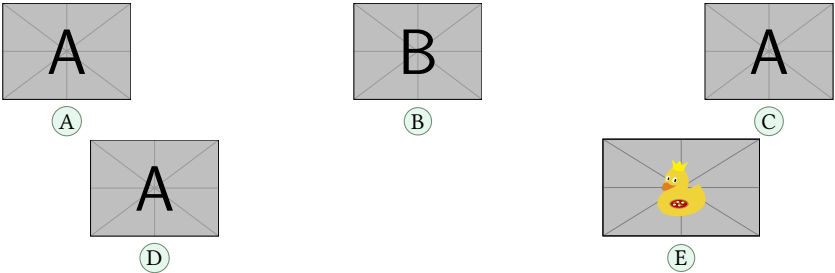
A value

B value

C value

D value

E value
4. Question with image and label below:



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 :) 📄.

1.

Factor  $x^2 - 2x + 1$

2.

Factor  $3x + 3y + 3z$

The following questions need to be cuaqtified :)

3.

True False

(a)  $\alpha > \delta$

(b)  $\LaTeX$ 2e 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`
- The answer to 1 is  $(x - 1)^2$  and the answer to 3.(a) is False.

1.  $(x-1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- \*

\*

\*

\*

\*
- (b) Yes, dnf

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

ii. very good

iii. obsolete
- \*


\*

\*

\*

\*

Example 5

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

1

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character for pursuing an unexpected course of action.

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 L<sup>A</sup>T<sub>E</sub>X for the four levels \textbullet, \textendash, \textasteriskcentered and \textperiodcentered together with the nosepe 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 nosepe key

- First level item

– Second level item

\* Third level item

· Fourth level item

• First level item
- \* First level item

◇ Second level item

○ Third level item

★ Fourth level item

\* First level item

Fake description environment

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

- Something A short one-line description.

This is an entry without a label.
- Something A short one-line description text.
- Something long A much longer description text may take more than one line or more than one paragraph.

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

**SomeThing** A short one-line description.  
This is an entry *without* a label.  
**Something** A short one-line description.  
**Something long** A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.  
**SoMeThInG** A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.  
**LoNg** 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  $\text{\LaTeX}$ , that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

## 8 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores  $\text{\LaTeX}$  contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The  $\text{\LaTeX}$  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<sub>ε</sub>: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The  $\text{\LaTeX}$  Project. “Extensive support for hypertext in  $\text{\LaTeX}$ ”. 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  $\text{\LaTeX}$  Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The  $\text{\LaTeX}$  Project. “The  $\text{\LaTeX}$ 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The  $\text{\LaTeX}$  Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.
- [13] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [14] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [15] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

## 9 Change history

**v1.0** **2024-06-07** – First public release.

10 Index of Documentation

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

C

Document class:

article2

book2

exam2

letter2

report2

\columnbreak4, 12

\columnsep10

Commands provide by enumext:

\anskey4, 10–13

\anspic\*14, 15

\anspic4, 10–12, 14, 15

\getkeyans4, 12, 15

\item\*4–7, 10–14

\itemwidth5

\item5–7, 9, 10, 12, 13

\miniright4, 10

\printkeyans4, 6, 11, 16

\setenumext4–7, 11, 14, 16

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

anskey\*4, 10–13

enumext\*4–14, 16

enumext4–14, 16, 19

keyans\*4–14

keyanspic4, 6, 8, 10–15, 19

keyans4–15, 19

Environments:

enumerate1, 3, 5, 21

figure5

list3, 9, 21

minipage3–5, 10, 21

multicols3, 4, 10

table5

task5

F

\footnote5

I

\item3, 5

\itemsep8

K

Keys for command provide by enumext:

break-col12

item-join12

item-pos\*12

item-star12

item-sym\*12

Keys for environments provide by enumext:

above\*8

above8

after9, 10

align7, 20

before\*9

before9

below\*8

below8

check-ans12

columns-sep4, 10

columns4, 8, 10

first9

font7

item-pos\*5, 6

item-sym\*5, 6

itemindent8

itemsep8, 15

labelsep3, 5–10, 12, 20

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

labelwidth5

label7, 9, 14, 19, 20

list-indent3, 9

list-offset3, 9, 20

listparindent9

mark-ans11, 12

mark-pos12

mark-ref11

mini-env4, 8, 10

mini-right\*6, 10

mini-right6, 10

mini-sep4, 10

no-store11–13

noitemsep8

nosep8, 19

parsep8, 15

partopsep8

ref4, 7

resume\*6, 9, 10

resume6, 9, 10

rightmargin8

save-ans4, 6, 9–16

save-key9, 11, 16

save-ref4, 7, 11–13, 15

save-sep11

series6, 9, 10

show-ans11

show-length7

show-pos11, 12, 15

start9

topsep8

widest7

wrap-ans11

wrap-label\*7, 20

wrap-label7

wrap-opt11

L

\label4

Labels provide by enumext:

\Alph\*7, 14

<code>\Roman*</code> .....	7	<code>l3prop</code> .....	1, 21
<code>\alph*</code> .....	7	<code>l3seq</code> .....	1, 21
<code>\arabic*</code> .....	7	<code>multicol</code> .....	1, 2, 4, 21
<code>\roman*</code> .....	7	<code>scontents</code> .....	1, 2, 13
<code>\labelsep</code> .....	3, 7	<code>task</code> .....	5, 6
<code>\labelwidth</code> .....	3, 7	<code>xsim</code> .....	2
<code>\linewidth</code> .....	10	<code>\parsep</code> .....	8
<code>\listparindent</code> .....	9	<code>\partopsep</code> .....	8

P

Packages:

<code>enumerate</code> .....	21
<code>enumext</code> .....	1–6, 15, 21
<code>enumitem</code> .....	3–5, 9, 20, 21
<code>footnotehyper</code> .....	4, 5
<code>hyperref</code> .....	4, 5, 11–13, 21
<code>l3keys</code> .....	6

R

<code>\raggedcolumns</code> .....	4
<code>\ref</code> .....	4
<code>\rightmargin</code> .....	8

T

<code>\topsep</code> .....	8
----------------------------	---

## 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  $\text{\TeX}$ 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 ( $\text{\LaTeX}$ 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  $\text{\LaTeX}$  to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-06-07}
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_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl
```

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

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

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

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

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

The initial values of these variables are set by the function `\__enumext_define_counters:Nn` (§11.9) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§11.12).

```
43 \cs_set_protected:Npn \__enumext_tmp:n #1
44 {
45   \tl_new:c { l__enumext_counter_#1_tl }
46 }
47 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by `ref` key (§11.12).

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

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

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

Internal variables used by `resume`, `resume*` and `series` keys (§11.22).

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

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



```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default `<label style>` and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§11.13) and `label` (§11.11) keys.

```

65 \dim_new:N \l__enumext_current_widest_dim
66 \tl_new:N \g__enumext_counter_styles_tl
67 \tl_new:N \g__enumext_widest_label_tl
68 \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 (§11.15). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:Nnnnnnnnnnn` (§11.33.1).

```

69 \cs_set_protected:Npn \__enumext_tmp:n #1
70 {
71   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
72   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
73   \dim_new:c { \l__enumext_leftmargin_#1_dim }
74   \dim_new:c { \l__enumext_itemindent_#1_dim }
75 }
76 \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 §11.19).

```

77 \cs_set_protected:Npn \__enumext_tmp:n #1
78 {
79   \skip_new:c { \l__enumext_multicols_above_#1_skip }
80   \skip_new:c { \l__enumext_multicols_below_#1_skip }
81 }
82 \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 (§11.20.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.18, §11.20).

```

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

```

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

```

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

```

The integer variable `\l__enumext_start_X_int` are used by the `start` key (§11.13), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§11.15.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.11). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§11.17).

```

95 \cs_set_protected:Npn \__enumext_tmp:n #1
96 {
97   \bool_new:c { \l__enumext_wrap_label_#1_bool }
98   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
99   \int_new:c { \l__enumext_start_#1_int }
100   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
101   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
102   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
103   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
104   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
105 }
106 \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 (§11.23.1) 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` (§11.26) and the variables `\l__enumext_store_anskey_env_tl`, `\l__enumext_store_anskey_opt_tl` save the  $\langle \textit{body} \rangle$  and  $\langle \textit{keys} \rangle$  of the environment `anskey*` (§11.27).

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*` (§11.31.2) and `\anspic*` (§11.36.1) 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.

```
107 \bool_new:N \l__enumext_store_active_bool
108 \tl_new:N \l__enumext_store_name_tl
109 \tl_new:N \g__enumext_store_name_tl
110 \tl_new:N \l__enumext_store_anskey_arg_tl
111 \tl_new:N \l__enumext_store_anskey_env_tl
112 \tl_new:N \l__enumext_store_anskey_opt_tl
113 \tl_new:N \l__enumext_store_current_label_tl
114 \tl_new:N \l__enumext_store_current_opt_arg_tl
115 \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` (§11.41).

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

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

```
\l__enumext_print_keyans_starred_tl
\l__enumext_mark_position_str
\g__enumext_item_symbol_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool
\l__enumext_store_upper_level_X_bool
```

Internal variables used by command `\printkeyans` (§11.40), `show-pos` key (§11.25), `item-sym*` key (§11.29), `save-key` key (§11.25.2) and “*storage level system*”.

```
121 \tl_new:N \l__enumext_print_keyans_starred_tl
122 \str_new:N \l__enumext_mark_position_str
123 \tl_new:N \g__enumext_item_symbol_tl
124 \cs_set_protected:Npn \__enumext_tmp:n #1
125 {
126   \tl_new:c { \l__enumext_print_keyans_#1_tl }
127   \tl_new:c { \l__enumext_store_save_key_#1_tl }
128   \bool_new:c { \l__enumext_store_save_key_#1_bool }
129   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
130 }
131 \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 (§11.36.2).

```
132 \seq_new:N \l__enumext_keyans_pic_body_seq
133 \dim_new:N \l__enumext_keyans_pic_width_dim
134 \int_new:N \l__enumext_keyans_pic_above_int
135 \int_new:N \l__enumext_keyans_pic_below_int
136 \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
```

Internal variables used by “*check answer*” mechanism (§11.23.3) 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.

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

```
144 \bool_new:N \l__enumext_hyperref_bool
145 \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 (§11.25). 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.

```
146 \tl_new:N \l__enumext_newlabel_arg_one_tl
147 \tl_new:N \l__enumext_newlabel_arg_two_tl
148 \tl_new:N \l__enumext_write_aux_file_tl
149 \cs_set_protected:Npn \__enumext_tmp:n #1
150 {
151   \tl_new:c { l__enumext_label_copy_#1_tl }
152 }
153 \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` (§11.30).

```
154 \int_new:N \g__enumext_footnote_int
155 \seq_new:N \g__enumext_footnote_arg_seq
156 \seq_new:N \g__enumext_footnote_int_seq
```

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

```
\l__enumext_item_starred_X_bool
\l__enumext_item_column_pos_X_int
\g__enumext_item_count_all_X_int
\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_item_text_X_box
\l__enumext_joined_width_X_dim
\l__enumext_item_width_X_dim
\g__enumext_item_symbol_aux_X_tl
\l__enumext_align_label_X_str
\g__enumext_minipage_active_X_bool
\g__enumext_miniright_code_X_tl
\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.

```
157 \cs_set_protected:Npn \__enumext_tmp:n #1
158 {
159   \bool_new:c { l__enumext_item_starred_#1_bool }
160   \int_new:c { l__enumext_item_column_pos_#1_int }
161   \int_new:c { g__enumext_item_count_all_#1_int }
162   \int_new:c { l__enumext_joined_item_#1_int }
163   \int_new:c { l__enumext_joined_item_aux_#1_int }
164   \int_new:c { l__enumext_tmpa_#1_int }
165   \box_new:c { l__enumext_item_text_#1_box }
166   \dim_new:c { l__enumext_joined_width_#1_dim }
167   \dim_new:c { l__enumext_item_width_#1_dim }
168   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
169   \str_new:c { l__enumext_align_label_#1_str }
170   \bool_new:c { g__enumext_minipage_active_#1_bool }
171   \tl_new:c { g__enumext_miniright_code_#1_tl }
172   \bool_new:c { g__enumext_minipage_center_#1_bool }
173   \dim_new:c { g__enumext_minipage_right_#1_dim }
174   \skip_new:c { g__enumext_minipage_right_#1_skip }
175 }
176 \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`.

```
177 \clist_const:Nn \c__enumext_all_envs_clist
178 {
179   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
180   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
181 }
```

(End of definition for `\c__enumext_all_envs_clist`.)

## 11.5 Some utility functions

`\__enumext_at_begin_document:n`

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

```
182 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
183 {
184   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
185 }
```

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

`\__enumext_after_env:nn`  
`\__enumext_before_env:nn`

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

```
186 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
187 {
188   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
189 }
190 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
191 {
192   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
193 }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
194 \cs_new:Nn \__enumext_level:
195 {
196   \int_to_roman:n { \l__enumext_level_int }
197 }
```

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

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

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

```
198 \prg_new_protected_conditional:Nppn \__enumext_if_is_int:n #1 { T, F, TF }
199 {
200   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
201   { \prg_return_true: }
202   { \prg_return_false: }
203 }
```

(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.

```
204 \cs_new_protected:Nn \__enumext_regex_counter_style:
205 {
206   \tl_map_inline:Nn \c__enumext_counter_style_tl
207   {
208     \regex_replace_once:nnN { \c{##1}\* }
209     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
210   }
211 }
```

(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.

```
212 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
213 {
214   * ~ #2
215   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
216   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
217 }
```

(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.

```

218 \cs_new_protected:Nn __enumext_is_not_nested:
219 {
220   \str_case:en { \@currentenv }
221   {
222     {enumext}
223     {
224       \bool_lazy_and:nnT
225       { \bool_not_p:n { \g__enumext_standar_bool } }
226       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
227       {
228         \bool_gset_true:N \g__enumext_standar_bool
229       }
230     }
231     {enumext*}
232     {
233       \bool_lazy_and:nnT
234       { \bool_not_p:n { \g__enumext_starred_bool } }
235       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
236       {
237         \bool_gset_true:N \g__enumext_starred_bool
238       }
239     }
240   }
241 }

```

The function `__enumext_is_on_first_level:` will set the variables `l__enumext_standar_first_bool` and `l__enumext_starred_first_bool` 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.

```

242 \cs_new_protected:Nn __enumext_is_on_first_level:
243 {
244   \bool_lazy_all:nT
245   {
246     { \bool_if_p:N \g__enumext_standar_bool }
247     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
248     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
249   }
250   {
251     \bool_set_true:N \l__enumext_standar_first_bool
252     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
253     \tl_gset:Nn \g__enumext_start_line_tl
254     {
255       on ~ line ~ \exp_not:V \inputlineno
256     }
257   }
258   \bool_lazy_all:nT
259   {
260     { \bool_if_p:N \g__enumext_starred_bool }
261     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
262     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
263   }
264   {
265     \bool_set_true:N \l__enumext_starred_first_bool
266     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
267     \tl_gset:Nn \g__enumext_start_line_tl
268     {
269       on ~ line ~ \exp_not:V \inputlineno
270     }
271   }
272 }

```

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

`__enumext_keyans_start_line:`

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

```

273 \cs_new_protected:Nn \__enumext_keyans_start_line:
274 {
275   \str_case:en { \@currenvir }
276   {
277     {keyans}
278     {
279       \tl_set:Nc \l__enumext_check_start_line_env_tl
280       {
281         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
282       }
283     }
284     {keyans*}
285     {
286       \tl_set:Nc \l__enumext_check_start_line_env_tl
287       {
288         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
289       }
290     }
291     {keyanspic}
292     {
293       \tl_set:Nc \l__enumext_check_start_line_env_tl
294       {
295         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
296       }
297     }
298   }
299 }

```

(End of definition for `\__enumext_keyans_start_line:`)

### 11.5.2 Utilities for log and terminal

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

```

\__enumext_reset_global_vars:
\__enumext_reset_global_int:
\__enumext_reset_global_bool:
\__enumext_reset_global_tl:
300 \cs_new_protected:Nn \__enumext_reset_global_vars:
301 {
302   \__enumext_reset_global_int:
303   \__enumext_reset_global_bool:
304   \__enumext_reset_global_tl:
305 }
306 \cs_new_protected:Nn \__enumext_reset_global_int:
307 {
308   \int_gzero:N \g__enumext_item_number_int
309   \int_gzero:N \g__enumext_item_anskey_int
310   \int_gzero:N \g__enumext_item_answer_diff_int
311 }
312 \cs_new_protected:Nn \__enumext_reset_global_bool:
313 {
314   \bool_gset_false:N \g__enumext_check_ans_key_bool
315   \bool_gset_false:N \g__enumext_standar_bool
316   \bool_gset_false:N \g__enumext_starred_bool
317 }
318 \cs_new_protected:Nn \__enumext_reset_global_tl:
319 {
320   \tl_gclear:N \g__enumext_store_name_tl
321   \tl_gclear:N \g__enumext_start_line_tl
322   \tl_gclear:N \g__enumext_envir_name_tl
323 }

```

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

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

```

324 \cs_new_protected:Nn \__enumext_log_global_vars:
325 {
326   \msg_log:nneeee { enumext } { prop-seq-int-hook }
327   { \g__enumext_store_name_tl }
328   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
329   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
330   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
331 }

```



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.

```

332 \cs_new_protected:Nn \__enumext_log_answer_vars:
333 {
334   \msg_log:nneee { enumext } { item-answer-hook }
335   { \int_use:N \__enumext_item_number_int }
336   { \int_use:N \__enumext_item_anskey_int }
337   { \int_eval:n { \__enumext_item_number_int - \__enumext_item_anskey_int } }
338 }

```

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

## 11.6 Copying list and minipage environments

The `list` environment provided by L<sup>A</sup>T<sub>E</sub>X 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.

```

339 \__enumext_at_begin_document:n
340 {
341   \cs_new_eq:NN \__enumext_start_list:nn \list
342   \cs_new_eq:NN \__enumext_stop_list: \endlist
343   \cs_new_eq:NN \__enumext_item_std:w \item
344 }

```

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

The `minipage` environment provided by L<sup>A</sup>T<sub>E</sub>X 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.

```

345 \__enumext_at_begin_document:n
346 {
347   \cs_new_eq:NN \__enumext_minipage:w \minipage
348   \cs_new_eq:NN \__enumext_endminipage: \endminipage
349 }

```

(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 (§11.34) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.37)

```

350 \cs_new_protected:Nn \__enumext_internal_mini_page:
351 {
352   \int_compare:nNt { \__enumext_level_int } = { 0 }
353   {
354     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
355     {
356       \__enumext_minipage:w [ t ] { ##1 }
357       \legacy_if_gset_false:n { @minipage }
358       \vspace { 0pt }
359     }
360   }

```

```

360         { \__enumext_endminipage: }
361     }
362 }

```

(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.

```

363 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
364 \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 `\l__enumext_footnotes_key_bool` to “true”.

```

365 \cs_new_protected:Nn \__enumext_after_hyperref:
366 {
367     \IfPackageLoadedTF { hyperref }
368     {
369         \msg_info:nnn { enumext } { package-load } { hyperref }
370         \bool_set_true:N \l__enumext_hyperref_bool
371         \IfHyperBoolean{hyperfootnotes}
372         {
373             \typeout{hyperfootnotes=true}
374             \bool_set_true:N \l__enumext_footnotes_key_bool
375         }
376         { \typeout{hyperfootnotes=false} }
377     }
378     { }

```

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`.

```

379     \bool_if:NT \l__enumext_footnotes_key_bool
380     {
381         \IfPackageLoadedTF { footnotehyper }
382         {
383             \msg_info:nnn { enumext } { package-load } { footnotehyper }
384         }
385         {
386             \typeout{No ~ footnotehyper ~ load}
387             \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
388             \bool_set_false:N \l__enumext_footnotes_key_bool
389         }
390     }

```

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.

```

391     \bool_if:NTF \l__enumext_hyperref_bool
392     {
393         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
394         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
395     }
396     {
397         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
398         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
399     }
400 }

```

(End of definition for \\_\_enumext\_after\_hyperref:, \\_\_enumext\_hypertarget:nn, and \\_\_enumext\_phantomsection:.)

```
\__enumext_newlabel:nn
```

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

#1: `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

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

```

401 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
402 {
403   \protected@write \@auxout { }
404   {
405     \token_to_str:N \newlabel {#1}
406     {
407       {#2}
408       \bool_if:NT \l__enumext_hyperref_bool
409       { { \thepage } {#2} {#1} }
410       { }
411     }
412   }
413   \__enumext_hypertarget:nn {#1} { }
414   \__enumext_phantomsection:
415 }

```

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

## 11.9 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

#2: The counter’s name.

```

416 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
417 {
418   \cs_if_exist:cTF { c@ #2 }
419   { \msg_fatal:nnn { enumext } { counters } { #2 } }
420   {
421     \tl_set:Nn #1 { #2 }
422     \newcounter { #2 }
423   }
424 }

```

(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      425 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     426 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    427 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii    428 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii   429 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXv      430 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXvi     431 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXviii   432 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

## 11.10 Definition of labels

This part of the code is inspired by the `enumitem` package. The idea is to be able to access the counters using `\arabic*`, `\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.

```

433 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
434 {
435   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
436   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
437 }
438 \__enumext_register_counter_style:Nn \arabic { 0 }
439 \__enumext_register_counter_style:Nn \Alph { M }
440 \__enumext_register_counter_style:Nn \alph { m }
441 \__enumext_register_counter_style:Nn \Roman { VIII }
442 \__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.

```
443 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
444 {
445   \hbox_set:Nn \__enumext_label_width_by_box {#2}
446   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
447 }
448 \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.

```
449 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
450 {
451   \tl_clear_new:N #1
452   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
453   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
454   \tl_map_inline:Nn \g__enumext_counter_styles_tl
455   {
456     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
457     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
458     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
459   }
460   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
461   { \tl_use:N \g__enumext_widest_label_tl }
462   \tl_set_eq:cN { the #2 } #1
463 }
464 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

### 11.11 Setting keys associated with label

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

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

```
465 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
466 {
467   \keys_define:nn { enumext / #1 }
468   {
469     font      .tl_set:c    = { l__enumext_label_font_style_#2_tl },
470     font      .value_required:n = true,
471     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
472     labelsep  .initial:n   = {0.3333em},
473     labelsep  .value_required:n = true,
474     labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
475     labelwidth .value_required:n = true,
476     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
477     wrap-label .initial:n   = {##1},
478     wrap-label .value_required:n = true,
479     wrap-label* .code:n = {
480       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
481       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
482     },
483     wrap-label* .value_required:n = true,
484   }
485 }
486 \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.

```

487 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
488 {
489   \keys_define:nn { enumext / #1 }
490   {
491     align .choice:,
492     align / left .code:n =
493       {
494         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
495         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
496       },
497     align / right .code:n =
498       {
499         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
500         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
501       },
502     align / center .code:n =
503       {
504         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
505         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
506       },
507     align / unknown .code:n =
508       \msg_error:nneee { enumext } { unknown-choice }
509       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
510     align .initial:n = left,
511     align .value_required:n = true,
512   }
513 }
514 \clist_map_inline:nn
515 {
516   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
517 }
518 { \__enumext_tmp:nn #1 }

519 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
520 {
521   \keys_define:nn { enumext / #1 }
522   {
523     align .choice:,
524     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
525     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
526     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
527     align / unknown .code:n =
528       \msg_error:nneee { enumext } { unknown-choice }
529       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
530     align .initial:n = left,
531     align .value_required:n = true,
532   }
533 }
534 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

## 11.12 Setting label and ref keys

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

### 11.12.1 Define and set label and ref keys for enumext environment

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

```

\__enumext_label_i_tl
\__enumext_label_ii_tl
\__enumext_label_iii_tl
\__enumext_label_iv_tl
535 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
536 {
537   \keys_define:nn { enumext / #1 }
538   {
539     label .code:n = {
540       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
541       { l__enumext_counter_#2_tl } {##1}
542       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
543       \l__enumext_current_widest_dim

```

```

544         },
545         label .initial:n = #3,
546         label .value_required:n = true,
547         ref .code:n = \__enumext_standar_ref:n {##1},
548         ref .value_required:n = true,
549     }
550 }
551 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
552 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
553 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
554 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

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

```

555 \cs_new_protected:Npn \__enumext_standar_ref:n #1
556 {
557     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
558     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
559     {
560         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
561     }
562     {
563         \tl_set_eq:Nc
564         \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
565         \__enumext_regex_counter_style:
566         \tl_set_eq:Nc
567         \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
568         \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
569         {
570             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
571             { \exp_not:V \l__enumext_ref_key_arg_tl }
572         }
573     }
574 }

```

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

```

575 \cs_new_protected:Npn \__enumext_standar_ref:
576 {
577     \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
578     {
579         \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
580     }
581 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
582 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
583 {
584     \keys_define:nn { enumext / #1 }
585     {
586         label .code:n = {
587             \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
588             { \l__enumext_counter_#2_tl } {##1}
589             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
590             \l__enumext_current_widest_dim
591         },
592         label .initial:n = #3,
593         label .value_required:n = true,
594         ref .code:n = \__enumext_starred_ref:n {##1},
595         ref .value_required:n = true,
596     }

```



```

597   }
598   \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
599   \__enumext_tmp:nnn { keyans* } { viii } { \Alph*} }

```

(End of definition for label and others.)

```

\__enumext_starred_ref:n
\__enumext_starred_ref:

```

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

```

600 \cs_new_protected:Npn \__enumext_starred_ref:n #1
601 {
602   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
603   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
604   {
605     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
606     {
607       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
608     }
609     {
610       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
611       \__enumext_regex_counter_style:
612       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
613       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
614       {
615         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
616         { \exp_not:V \l__enumext_ref_key_arg_tl }
617       }
618     }
619   }
620   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
621   {
622     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
623     {
624       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
625     }
626     {
627       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
628       \__enumext_regex_counter_style:
629       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
630       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
631       {
632         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
633         { \exp_not:V \l__enumext_ref_key_arg_tl }
634       }
635     }
636   }
637 }

```

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.

```

638 \cs_new_protected:Nn \__enumext_starred_ref:
639 {
640   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
641   {
642     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
643     {
644       \tl_use:N \l__enumext_renew_the_count_vii_tl
645     }
646   }
647   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
648   {
649     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
650     {
651       \tl_use:N \l__enumext_renew_the_count_viii_tl
652     }
653   }
654 }

```

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

### 11.12.3 Define and set label and ref keys for keyans and keyanspic environments

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

```

label
ref
\__enumext_label_v_tl 655 \keys_define:nn { enumext / keyans }
\__enumext_label_vi_tl 656 {
657   label .code:n = {
658     \__enumext_label_style:cnv { \__enumext_label_v_tl }
659     { \__enumext_counter_v_tl } {#1}
660     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
661     \__enumext_current_widest_dim
662     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
663     { \__enumext_counter_vi_tl } {#1}
664     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
665     \__enumext_current_widest_dim
666   },
667   label .initial:n = \Alph*,
668   label .value_required:n = true,
669   ref .code:n = \__enumext_keyans_ref:n {#1},
670   ref .value_required:n = true,
671 }

```

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

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

```

\__enumext_keyans_ref:n 672 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
673 {
674   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
675   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
676   {
677     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
678   }
679   {
680     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_v_tl
681     \__enumext_regex_counter_style:
682     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
683     \tl_put_right:Ne \__enumext_renew_the_count_v_tl
684     {
685       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
686       { \exp_not:V \__enumext_ref_key_arg_tl }
687     }
688   }
689 }

```

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

```

690 \cs_new_protected:Nn \__enumext_keyans_ref:
691 {
692   \tl_if_empty:NF \__enumext_renew_the_count_v_tl
693   {
694     \tl_use:N \__enumext_renew_the_count_v_tl
695   }
696 }

```

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

### 11.13 Setting start and widest keys

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

```

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

```

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

```

697 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
698 {
699   \__enumext_if_is_int:nTF { #3 }
700   {
701     \int_set:Nn #2 {#3}
702   }

```

```

703     {
704         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
705         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
706         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
707         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
708     }
709 }
710 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

(End of definition for \\_\_enumext\_start\_from:NNn.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

The function \\_\_enumext\_widest\_from:nNNn used by the `widest` key take four arguments:

- #1: The counter associated with the environment level
- #2: \l\_\_enumext\_label\_X\_tl
- #3: \l\_\_enumext\_labelwidth\_X\_dim
- #4: *<integer or string>*

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

```

711 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
712 {
713     \__enumext_if_is_int:nTF {#4}
714     {
715         \setcounter{enumX#1} { #4 }
716     }
717     {
718         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
719         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
720         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
721         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
722     }
723     \__enumext_label_width_by_box:cv
724     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
725 }
726 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for \\_\_enumext\_widest\_from:nNNn.)

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

```

start
widest
\l__enumext_start_X_int
727 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
728 {
729     \keys_define:nn { enumext / #1 }
730     {
731         start .code:n = {
732             \__enumext_start_from:ccn
733             { \l__enumext_label_#2_tl }
734             { \l__enumext_start_#2_int } {##1}
735         },
736         start .initial:n = 1,
737         widest .code:n = {
738             \__enumext_widest_from:nccn {#2}
739             { \l__enumext_label_#2_tl }
740             { \l__enumext_labelwidth_#2_dim } {##1}
741         },
742         widest .value_required:n = true,
743         start .value_required:n = true,
744     }
745 }
746 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

## 11.14 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
747 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
748 {
749     \keys_define:nn { enumext / #1 }

```

```

750     {
751         topsep      .skip_set:c = { l__enumext_topsep_#2_skip },
752         topsep      .initial:n  = {#3},
753         topsep      .value_required:n = true,
754         partopsep   .skip_set:c = { l__enumext_partopsep_#2_skip },
755         partopsep   .initial:n  = {#4},
756         partopsep   .value_required:n = true,
757         parsep      .skip_set:c = { l__enumext_parsep_#2_skip },
758         parsep      .initial:n  = {#5},
759         parsep      .value_required:n = true,
760         itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
761         itemsep     .initial:n  = {#6},
762         itemsep     .value_required:n = true,
763         noitemsep   .meta:n      = { itemsep = 0pt, parsep = 0pt },
764         noitemsep   .value_forbidden:n = true,
765         nosep       .meta:n      = {
766                                 itemsep = 0pt, parsep= 0pt,
767                                 topsep = 0pt, partopsep = 0pt,
768                                 },
769         nosep       .value_forbidden:n = true,
770     }
771 }

```

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

```

772 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
773 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
774 { 4.0pt plus 2.0pt minus 1.0pt }
775 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
776 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
777 { 2.0pt plus 1.0pt minus 1.0pt }
778 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
779 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
780 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
781 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
782 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
783 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
784 { 2.0pt plus 1.0pt minus 1.0pt }
785 \__enumext_tmp:nnnnnn { enumext* } { vii } { 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 { keyans* } { viii } { 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 }

```

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

## 11.15 Setting keys for horizontal spaces

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

```

791 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
792 {
793     \keys_define:nn { enumext / #1 }
794     {
795         itemindent      .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
796         itemindent      .value_required:n = true,
797         rightmargin     .dim_set:c = { l__enumext_rightmargin_#2_dim },
798         rightmargin     .value_required:n = true,
799         listparindent   .dim_set:c = { l__enumext_listparindent_#2_dim },
800         listparindent   .value_required:n = true,
801         list-offset     .dim_set:c = { l__enumext_listoffset_#2_dim },
802         list-offset     .value_required:n = true,
803         list-indent     .code:n      =
804                         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
805                         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
806         list-indent     .value_required:n = true,
807     }
808 }
809 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

For `enumext*` and `keyans*` environments the situation is a bit different, the `list-indent` key behaves like the `list-offset` key.

```

810 \cs_set_protected:Npn \__enumext_tmp:n #1
811 {
812   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
813 }
814 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

### 11.15.1 Functions for setting the fake itemindent

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

```

815 \cs_set_protected:Npn \__enumext_fake_item:
816 {
817   \dim_compare:nNnT
818     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
819     >
820     { \c_zero_dim }
821   {
822     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
823     {
824       \exp_not:N \mode_leave_vertical:
825       \exp_not:n { \skip_horizontal:n }
826       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
827       \ignorespaces
828     }
829   }
830 }
831 \cs_set_protected:Npn \__enumext_keyans_fake_item:
832 {
833   \dim_compare:nNnT
834     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
835   {
836     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
837     {
838       \exp_not:N \mode_leave_vertical:
839       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
840     }
841   }
842 }
843 \cs_set_protected:Npn \__enumext_fake_item_vii:
844 {
845   \dim_compare:nNnT
846     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
847   {
848     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
849     {
850       \exp_not:N \mode_leave_vertical:
851       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
852     }
853   }
854 }
855 \cs_set_protected:Npn \__enumext_fake_item_viii:
856 {
857   \dim_compare:nNnT
858     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
859   {
860     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
861     {
862       \exp_not:N \mode_leave_vertical:
863       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
864     }
865   }
866 }

```

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

## 11.16 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 `\__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.

```

867 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
868 {
869   \keys_define:nn { enumext / #1 }
870   {
871     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
872     show-length .initial:n = false,
873   }
874 }
875 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

## 11.17 Setting before, after and first keys

before

before\*

after

first

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

```

876 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
877 {
878   \keys_define:nn { enumext / #1 }
879   {
880     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
881     before .value_required:n = true,
882     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
883     before* .value_required:n = true,
884     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
885     after .value_required:n = true,
886     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
887     first .value_required:n = true,
888   }
889 }
890 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 11.17.1 Functions for before, after and first keys in enumext

\\_\_enumext\_before\_args\_exec:

\\_\_enumext\_before\_keys\_exec:

\\_\_enumext\_after\_stop\_list:

\\_\_enumext\_after\_args\_exec:

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

```

891 \cs_new_protected:Nn \__enumext_before_args_exec:
892 {
893   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
894 }

```

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

```

895 \cs_new_protected:Nn \__enumext_before_keys_exec:
896 {
897   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
898 }

```

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

```

899 \cs_new_protected:Nn \__enumext_after_stop_list:
900 {
901   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
902 }

```

The function `\__enumext_after_args_exec:` executes the  $\{\langle code \rangle\}$  set by the `first` key after the end of the *second argument* of the list defining the `enumext` environment, just before the first occurrence of `\item`.

```

903 \cs_new_protected:Nn \__enumext_after_args_exec:
904 {
905   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
906 }

```

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



### 11.17.2 Functions for before, after and first keys in keyans

```

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:

```

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

```

907 \cs_new_protected:Nn __enumext_before_args_exec_v:
908 {
909     \tl_use:N \l__enumext_before_starred_key_v_tl
910 }

```

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

```

911 \cs_new_protected:Nn __enumext_before_keys_exec_v:
912 {
913     \tl_use:N \l__enumext_before_no_starred_key_v_tl
914 }

```

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

```

915 \cs_new_protected:Nn __enumext_after_stop_list_v:
916 {
917     \tl_use:N \l__enumext_after_stop_list_v_tl
918 }

```

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

```

919 \cs_new_protected:Nn __enumext_after_args_exec_v:
920 {
921     \tl_use:N \l__enumext_after_list_args_v_tl
922 }

```

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

### 11.17.3 Functions for before, after and first keys in enumext\* and keyans\*

```

__enumext_before_args_exec_vii:
__enumext_before_keys_exec_vii:
__enumext_after_stop_list_vii:
__enumext_after_args_exec_vii:

```

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

```

923 \cs_new_protected:Nn __enumext_before_args_exec_vii:
924 {
925     \tl_use:N \l__enumext_before_starred_key_vii_tl
926 }
927 \cs_new_protected:Nn __enumext_before_args_exec_viii:
928 {
929     \tl_use:N \l__enumext_before_starred_key_viii_tl
930 }

```

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

```

931 \cs_new_protected:Nn __enumext_before_keys_exec_vii:
932 {
933     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
934 }
935 \cs_new_protected:Nn __enumext_before_keys_exec_viii:
936 {
937     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
938 }

```

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

```

939 \cs_new_protected:Nn __enumext_after_stop_list_vii:
940 {
941     \tl_use:N \l__enumext_after_stop_list_vii_tl
942 }
943 \cs_new_protected:Nn __enumext_after_stop_list_viii:
944 {
945     \tl_use:N \l__enumext_after_stop_list_viii_tl
946 }

```

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

```

947 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
948 {
949   \tl_use:N \l__enumext_after_list_args_vii_tl
950 }
951 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
952 {
953   \tl_use:N \l__enumext_after_list_args_viii_tl
954 }

```

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

### 11.18 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*`, `keyans` and `keyans*` environments.

```

955 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
956 {
957   \keys_define:nn { enumext / #1 }
958   {
959     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
960     mini-env .value_required:n = true,
961     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
962     mini-sep .initial:n = 0.3333em,
963     mini-sep .value_required:n = true,
964     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
965     columns-sep .value_required:n = true,
966     columns .int_set:c = { l__enumext_columns_#2_int },
967     columns .initial:n = 1,
968     columns .value_required:n = true,
969   }
970 }
971 \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.

```

972 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
973 {
974   \keys_define:nn { enumext / #1 }
975   {
976     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
977     mini-right .value_required:n = true,
978     mini-right* .code:n = {
979       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
980       \keys_set:nn { enumext / #1 } { miniright = {##1} }
981     },
982     mini-right* .value_required:n = true,
983   }
984 }
985 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

### 11.19 Adjustment of vertical spaces for multicols

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



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

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

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

### 11.19.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

986 \cs_new_protected:Nn \__enumext_multi_set_vskip:
987 {
988   \skip_set:cn { \__enumext_multicols_above_ \__enumext_level: } _skip {
989     {
990       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
991       }
992     }
993   \skip_set:cn { \__enumext_multicols_below_ \__enumext_level: } _skip {
994     {
995       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
996     }
997   }

```

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

`\__enumext_add_pre_parsep:` The function `\__enumext_add_pre_parsep:` “adjusted” the value of `\__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*.

```

998 \cs_new_protected:Nn \__enumext_add_pre_parsep:
999 {
1000   \int_case:nn { \__enumext_level_int }
1001   {
1002     { 2 }{
1003       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
1004       {
1005         \skip_add:Nn \__enumext_multicols_above_ii_skip { \__enumext_parsep_i_skip }
1006       }
1007     }
1008     { 3 }{
1009       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip }
1010       {
1011         \skip_add:Nn \__enumext_multicols_above_iii_skip { \__enumext_parsep_ii_skip }
1012       }
1013     }
1014     { 4 }{
1015       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip }
1016       {
1017         \skip_add:Nn \__enumext_multicols_above_iv_skip { \__enumext_parsep_iii_skip }
1018       }
1019     }
1020   }
1021 }

```

(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*.

```

1022 \cs_new_protected:Nn \__enumext_multi_addvspace:
1023 {
1024   \__enumext_multi_set_vskip:
1025   \mode_if_vertical:T
1026   {

```

```

1027     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1028     {
1029         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1030     }
1031     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1032     {
1033         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1034     }
1035 }
1036 \par\nopagebreak
1037 \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1038 }

```

(End of definition for `\__enumext_multi_addvspace:`)

### 11.19.2 Adjustment of vertical spaces for multicols in keyans

`\__enumext_keyans_multi_set_vskip:`  
`\__enumext_keyans_multi_addvspace:`

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

```

1039 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1040 {
1041     \skip_set:Nn \l__enumext_multicols_above_v_skip
1042     {
1043         \l__enumext_topsep_v_skip
1044     }
1045     \skip_set:Nn \l__enumext_multicols_below_v_skip
1046     {
1047         \l__enumext_topsep_v_skip
1048     }
1049 }
1050 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1051 {
1052     \__enumext_keyans_multi_set_vskip:
1053     \mode_if_vertical:T
1054     {
1055         \skip_add:Nn \l__enumext_multicols_above_v_skip
1056         {
1057             \skip_use:N \l__enumext_partopsep_v_skip
1058         }
1059         \skip_add:Nn \l__enumext_multicols_below_v_skip
1060         {
1061             \skip_use:N \l__enumext_partopsep_v_skip
1062         }
1063     }
1064     \par\nopagebreak
1065     \addvspace{ \l__enumext_multicols_above_v_skip }
1066 }

```

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

### 11.20 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.

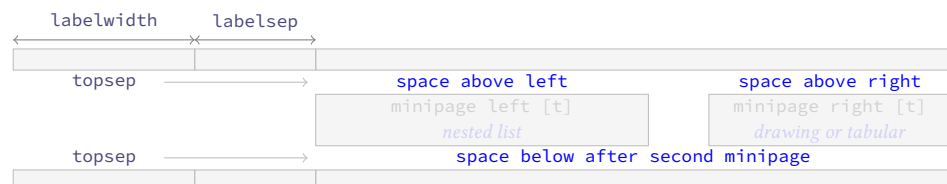


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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`[13] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

### 11.20.1 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`.

```

1067 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1068 {
1069     \int_compare:nNnTF
1070     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1071     {

```

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 `\parsecp` of the previous level, if these are zero we will use `\strutbox` as the basis for the calculations.

```

1072     \skip_if_eq:nnTF
1073     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1074     {
1075         \skip_set:Nn \l__enumext_minipage_left_skip
1076         {
1077             -0.15\box_dp:N \strutbox
1078         }
1079         \skip_set:Nn \l__enumext_minipage_right_skip
1080         {
1081             0.695\box_dp:N \strutbox
1082         }
1083         \skip_set:Nn \l__enumext_minipage_after_skip
1084         {
1085             \box_dp:N \strutbox
1086         }
1087         \__enumext_zero_parsep:
1088     }
1089     {
1090         \skip_set:Nn \l__enumext_minipage_left_skip
1091         {
1092             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1093         }
1094         \skip_set:Nn \l__enumext_minipage_right_skip
1095         {
1096             0.695\box_dp:N \strutbox
1097         }
1098         \skip_set:Nn \l__enumext_minipage_after_skip
1099         {
1100             1.85\box_dp:N \strutbox
1101             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1102         }
1103     }
1104 }
1105 {

```

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.

```

1106     \skip_if_eq:nnTF
1107     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1108     {
1109         \skip_set:Nn \l__enumext_minipage_left_skip
1110         {
1111             0.5\box_dp:N \strutbox
1112             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1113         }
1114         \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1115         {
1116             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1117         }
1118         \skip_set:Nn \l__enumext_minipage_after_skip
1119         {
1120             1.6\box_dp:N \strutbox
1121         }
1122     }
1123     {
1124         \skip_set:Nn \l__enumext_minipage_left_skip
1125         {
1126             0.5875\box_dp:N \strutbox
1127             - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1128         }
1129         \skip_set:Nn \l__enumext_minipage_right_skip
1130         {
1131             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1132             + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1133         }
1134         \skip_set:Nn \l__enumext_minipage_after_skip
1135         {
1136             0.325\box_dp:N \strutbox
1137             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1138         }
1139     }
1140 }
1141 }

```

(End of definition for \l\_\_enumext\_mini\_set\_vskip:.)

\l\_\_enumext\_zero\_parsep: The function \l\_\_enumext\_zero\_parsep: “adjusted” the value of \l\_\_enumext\_minipage\_after\_skip detecting the value of \l\_\_enumext\_minipage\_after\_skip from the previous level. This is necessary since \l\_\_enumext\_minipage\_after\_skip from the previous level affects the *vertical spaces* and this is noticeable when using the `nosep` or `noitemsep` keys.

```

1142 \cs_new_protected:Nn \l__enumext_zero_parsep:
1143 {
1144     \int_case:nn { \l__enumext_level_int }
1145     {
1146         { 2 }{
1147             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1148             {
1149                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1150             }
1151         }
1152         { 3 }{
1153             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1154             {
1155                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1156             }
1157         }
1158         { 4 }{
1159             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1160             {
1161                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1162             }
1163         }
1164     }
1165 }

```

(End of definition for \l\_\_enumext\_zero\_parsep:.)

\l\_\_enumext\_mini\_addvspace: The function \l\_\_enumext\_mini\_addvspace: will apply the spaces set using \l\_\_enumext\_mini\_addvspace “above” the \l\_\_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 \l\_\_enumext\_partopsep parameter comes into play and this affects the *vertical spacing*.

```

1166 \cs_new_protected:Nn \l__enumext_mini_addvspace:
1167 {
1168     \l__enumext_mini_set_vskip:
1169     \mode_if_vertical:T
1170     {
1171         \skip_add:Nn \l__enumext_minipage_left_skip
1172         {

```



```

1173         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1174     }
1175     \skip_add:Nn \l__enumext_minipage_after_skip
1176     {
1177         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1178     }
1179 }
1180 \par\nopagebreak
1181 \addvspace { \l__enumext_minipage_left_skip }
1182 }

```

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

### 11.20.2 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`.

```

1183 \cs_new_protected:Nn \l__enumext_keyans_mini_set_vskip:
1184 {
1185     \skip_zero_new:N \l__enumext_minipage_after_skip
1186     \skip_zero_new:N \l__enumext_minipage_left_skip
1187     \skip_zero_new:N \l__enumext_minipage_right_skip
1188     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1189     {
1190         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1191         {
1192             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1193             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1194             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1195             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1196             {
1197                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1198             }
1199         }
1200     }
1201     \skip_set:Nn \l__enumext_minipage_left_skip
1202     {
1203         \skip_use:N \l__enumext_topsep_v_skip
1204     }
1205     \skip_set:Nn \l__enumext_minipage_right_skip
1206     {
1207         0.705\box_dp:N \strutbox
1208     }
1209     \skip_set:Nn \l__enumext_minipage_after_skip
1210     {
1211         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1212     }
1213 }
1214 {
1215     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1216     {
1217         \skip_set:Nn \l__enumext_minipage_left_skip
1218         {
1219             0.5\box_dp:N \strutbox
1220             + \l__enumext_partopsep_v_skip
1221         }
1222         \skip_set:Nn \l__enumext_minipage_right_skip
1223         {
1224             \l__enumext_partopsep_v_skip
1225         }
1226         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1227     }
1228     {
1229         \skip_set:Nn \l__enumext_minipage_left_skip
1230         {
1231             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1232         }
1233         \skip_set:Nn \l__enumext_minipage_right_skip
1234         {
1235             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1236         }
1237     }
1238 }

```

```

1237         }
1238         \skip_set:Nn \l__enumext_minipage_after_skip
1239         {
1240             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1241         }
1242     }
1243 }
1244 }

```

(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  $\text{\TeX}$  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`.

```

1245 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1246 {
1247     \__enumext_keyans_mini_set_vskip:
1248     \mode_if_vertical:T
1249     {
1250         \skip_add:Nn \l__enumext_minipage_left_skip
1251         {
1252             \l__enumext_partopsep_v_skip
1253         }
1254         \skip_add:Nn \l__enumext_minipage_after_skip
1255         {
1256             \l__enumext_partopsep_v_skip
1257         }
1258     }
1259     \par\nopagebreak
1260     \addvspace { \l__enumext_minipage_left_skip }
1261 }

```

(End of definition for `\__enumext_keyans_mini_addvspace:`)

### 11.20.3 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*`.

```

1262 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1263 {
1264     \skip_zero_new:N \l__enumext_minipage_left_skip
1265     \skip_gzero_new:N \g__enumext_minipage_right_skip
1266     \skip_gzero_new:N \g__enumext_minipage_after_skip
1267     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1268     {
1269         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1270         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1271     }
1272     {
1273         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1274         \skip_gset:Nn \g__enumext_minipage_right_skip
1275         {
1276             \l__enumext_topsep_vii_skip
1277         }
1278         \skip_gset:Nn \g__enumext_minipage_after_skip
1279         {
1280             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1281         }
1282     }
1283 }
1284 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1285 {
1286     \skip_zero_new:N \l__enumext_minipage_after_skip
1287     \skip_zero_new:N \l__enumext_minipage_left_skip
1288     \skip_zero_new:N \l__enumext_minipage_right_skip
1289     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1290     {
1291         \skip_set:Nn \l__enumext_minipage_left_skip

```

```

1292     {
1293         0.5\box_dp:N \strutbox
1294     }
1295     \skip_set:Nn \l__enumext_minipage_right_skip
1296     {
1297         \l__enumext_partopsep_viii_skip
1298     }
1299     \skip_set:Nn \l__enumext_minipage_after_skip
1300     {
1301         1.6\box_dp:N \strutbox
1302     }
1303 }
1304 {
1305     \skip_set:Nn \l__enumext_minipage_left_skip
1306     {
1307         0.5875\box_dp:N \strutbox
1308     }
1309     \skip_set:Nn \l__enumext_minipage_right_skip
1310     {
1311         \l__enumext_topsep_viii_skip
1312     }
1313     \skip_set:Nn \l__enumext_minipage_after_skip
1314     {
1315         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1316     }
1317 }
1318 }

```

(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  $\text{\TeX}$  is in  $\langle\textit{horizontal mode}\rangle$  or  $\langle\textit{vertical mode}\rangle$ , since `\partopsep` is equal to `0pt` in both environments.

```

1319 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1320 {
1321     \__enumext_mini_set_vskip_vii:
1322     \par\nopagebreak
1323     \addvspace { \l__enumext_minipage_left_skip }
1324 }
1325 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1326 {
1327     \__enumext_mini_set_vskip_viii:
1328     \par\nopagebreak
1329     \addvspace { \l__enumext_minipage_left_skip }
1330 }

```

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

#### 11.20.4 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  $\text{\TeX}$  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.

```

1331 \NewDocumentCommand \miniright { s }
1332 {
1333     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1334     {
1335         \msg_error:nnn { enumext } { wrong-miniright-place }
1336     }
1337     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1338     {
1339         \msg_error:nnn { enumext } { wrong-miniright-place }
1340     }

```

```

1341 \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1342 {
1343   \l__enumext_keyans_mini_right_cmd:n {#1}
1344 }
1345 { \l__enumext_mini_right_cmd:n {#1} }
1346 }

```

(End of definition for `\miniright`. This function is documented on page 9.)

`\l__enumext_mini_right_cmd:n`

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

```

1347 \cs_new_protected:Npn \l__enumext_mini_right_cmd:n #1
1348 {
1349   \dim_compare:nNnTF
1350   { \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim } } > { \c_zero_dim }
1351   {
1352     \l__enumext_multicols_stop:
1353     \end{__enumext_mini_env*}
1354     \hfill
1355     \begin{__enumext_mini_env*}
1356     { \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim } }
1357     \par\addvspace { \l__enumext_minipage_right_skip }
1358     \bool_if:nF {#1}
1359     {
1360       \centering
1361     }
1362     \int_gzero:N \g__enumext_minipage_stat_int
1363   }
1364   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1365 }

```

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

`\l__enumext_keyans_mini_right_cmd:n`

The function `\l__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 `\l__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1366 \cs_new_protected:Npn \l__enumext_keyans_mini_right_cmd:n #1
1367 {
1368   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1369   {
1370     \l__enumext_keyans_multicols_stop:
1371     \end{__enumext_mini_env*}
1372     \hfill
1373     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1374     \par\addvspace { \l__enumext_minipage_right_skip }
1375     \bool_if:nF {#1}
1376     {
1377       \centering
1378     }
1379     \int_gzero:N \g__enumext_minipage_stat_int
1380   }
1381   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1382 }

```

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

### 11.21 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.

```

1383 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1384 {
1385   \keys_define:nn { enumext / #1 }
1386   {
1387     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1388     above .value_required:n = true,
1389     above* .code:n      = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1390                       \keys_set:nn { enumext / #1 } { above = {##1} },
1391     above* .value_required:n = true,
1392     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1393     below .value_required:n = true,
1394     below* .code:n      = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1395                       \keys_set:nn { enumext / #1 } { below = {##1} },
1396     below* .value_required:n = true,
1397   }
1398 }
1399 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

#### 11.21.1 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.

```

1400 \cs_new_protected:Nn \__enumext_vspace_above:
1401 {
1402   \skip_if_eq:nnF
1403   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1404   {
1405     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1406     {
1407       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1408     }
1409     {
1410       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1411     }
1412   }
1413 }

```

(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.

```

1414 \cs_new_protected:Nn \__enumext_vspace_below:
1415 {
1416   \skip_if_eq:nnF
1417   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1418   {
1419     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1420     {
1421       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1422     }
1423     {
1424       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1425     }
1426   }
1427 }

```

(End of definition for \\_\_enumext\_vspace\_below:.)

#### 11.21.2 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.

```

1428 \cs_new_protected:Nn \__enumext_vspace_above_v:
1429 {
1430   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1431   {
1432     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1433     {
1434       \vspace*{ \l__enumext_vspace_above_v_skip }

```

```

1435     }
1436     { \vspace { \l__enumext_vspace_above_v_skip } }
1437   }
1438 }

```

(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.

```

1439 \cs_new_protected:Nn \__enumext_vspace_below_v:
1440 {
1441   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1442   {
1443     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1444     {
1445       \vspace*{ \l__enumext_vspace_below_v_skip }
1446     }
1447     { \vspace { \l__enumext_vspace_below_v_skip } }
1448   }
1449 }

```

(End of definition for `\__enumext_vspace_below_v:`)

### 11.21.3 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:`

```

1450 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1451 {
1452   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1453   {
1454     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1455     {
1456       \vspace*{ \l__enumext_vspace_above_vii_skip }
1457     }
1458     { \vspace { \l__enumext_vspace_above_vii_skip } }
1459   }
1460 }
1461 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1462 {
1463   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1464   {
1465     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1466     {
1467       \vspace*{ \l__enumext_vspace_above_viii_skip }
1468     }
1469     { \vspace { \l__enumext_vspace_above_viii_skip } }
1470   }
1471 }

```

(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:`

```

1472 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1473 {
1474   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1475   {
1476     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1477     {
1478       \vspace*{ \l__enumext_vspace_below_vii_skip }
1479     }
1480     { \vspace { \l__enumext_vspace_below_vii_skip } }
1481   }
1482 }
1483 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1484 {
1485   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1486   {
1487     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1488     {
1489       \vspace*{ \l__enumext_vspace_below_viii_skip }

```

```

1490     }
1491     { \vspace { \l__enumext_vspace_below_viii_skip } }
1492   }
1493 }

```

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

## 11.22 Setting series, resume and resume\* keys

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

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

```

series
resume
resume*
1494 \cs_set_protected:Npn \__enumext_tmp:n #1
1495 {
1496   \keys_define:nn { enumext / #1 }
1497   {
1498     series .str_set:N = \l__enumext_series_str,
1499     series .value_required:n = true,
1500     resume .code:n = \__enumext_resume_series:n {##1},
1501     resume* .code:n = \__enumext_resume_starred:,
1502     resume* .value_forbidden:n = true,
1503   }
1504 }
1505 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

### 11.22.1 Internal functions for series key

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

```

1506 \cs_new:Npn \__enumext_filter_series:n #1
1507 {
1508   \use:e
1509   {
1510     \keyval_parse:NNn
1511     \__enumext_filter_series_key:n
1512     \__enumext_filter_series_pair:nn {#1}
1513   }
1514 }

```

The function `\__enumext_filter_series_key:n` will be responsible for filtering the `<keys>` that are passed “without value” by excluding the `resume` and `resume*` keys.

```

1515 \cs_new:Npn \__enumext_filter_series_key:n #1
1516 {
1517   \str_case:nnF {#1}
1518   {
1519     { resume } {}
1520     { resume* } {}
1521   }
1522   { , { \exp_not:n {#1} } }
1523 }

```

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

```

1524 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1525 {
1526   \str_case:nnF {#1}
1527   {
1528     { series } {}
1529     { resume } {}
1530     { start } {}
1531     { save-ans } {}
1532     { save-key } {}
1533   }
1534   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1535 }

```

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



```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `\__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§11.34) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.37).

```

1536 \cs_new_protected:Npn \__enumext_parse_series:n #1
1537 {
1538   \str_if_empty:NTF \l__enumext_series_str
1539   {
1540     \bool_if:NF \l__enumext_resume_active_bool
1541     {
1542       \__enumext_resume_last:n {#1}
1543     }
1544   }
1545   {
1546     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1547     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1548       { \__enumext_filter_series:n {#1} }
1549     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1550     {
1551       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1552     }
1553   }
1554 }

```

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.

```

1555 \cs_new_protected:Npn \__enumext_resume_last:n #1
1556 {
1557   \bool_if:NT \l__enumext_standar_first_bool
1558   {
1559     \tl_gclear:N \g__enumext_standar_series_tl
1560     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1561   }
1562   \bool_if:NT \l__enumext_starred_first_bool
1563   {
1564     \tl_gclear:N \g__enumext_starred_series_tl
1565     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1566   }
1567 }

```

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

### 11.22.2 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 (§11.34) and the `enumext*` environment definition (§11.37).

```

1568 \cs_new_protected:Npn \__enumext_resume_save_counter:
1569 {
1570   \bool_if:NT \g__enumext_standar_bool
1571   {
1572     \tl_if_empty:NF \l__enumext_series_str
1573     {
1574       \int_gset_eq:cN

```

```

1575         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1576     }
1577     \tl_if_empty:NTF \l__enumext_resume_name_tl
1578     {
1579         \str_if_empty:NT \l__enumext_series_str
1580         {
1581             \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1582         }
1583     }
1584     {
1585         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1586         {
1587             \int_gset_eq:cN
1588             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1589         }
1590     }
1591     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1592     {
1593         \int_gset_eq:cN
1594         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1595     }
1596 }
1597 \bool_if:NT \g__enumext_starred_bool
1598 {
1599     \tl_if_empty:NF \l__enumext_series_str
1600     {
1601         \int_gset_eq:cN
1602         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1603     }
1604     \tl_if_empty:NTF \l__enumext_resume_name_tl
1605     {
1606         \str_if_empty:NT \l__enumext_series_str
1607         {
1608             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1609         }
1610     }
1611     {
1612         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1613         {
1614             \int_gset_eq:cN
1615             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1616         }
1617     }
1618     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1619     {
1620         \int_gset_eq:cN
1621         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1622     }
1623 }
1624 }

```

(End of definition for \\_\_enumext\_resume\_save\_counter:.)

### 11.22.3 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.

```

1625 \cs_new_protected:Npn \__enumext_resume_series:n #1
1626 {
1627     \tl_if_empty:nTF {#1}
1628     {
1629         \__enumext_resume_counter:n { }
1630     }
1631     {
1632         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1633         {
1634             \__enumext_resume_counter:n {#1}

```

```

1635         \bool_if:NT \g__enumext_standar_bool
1636         {
1637             \keys_set:nv { enumext / level-1 }
1638             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1639         }
1640         \bool_if:NT \g__enumext_starred_bool
1641         {
1642             \keys_set:nv { enumext / enumext* }
1643             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1644         }
1645     }
1646     {
1647         \bool_if:NT \g__enumext_standar_bool
1648         {
1649             \msg_error:nnn { enumext } { unknown-series } {#1}
1650         }
1651         \bool_if:NT \g__enumext_starred_bool
1652         {
1653             \msg_error:nnn { enumext } { unknown-series } {#1}
1654         }
1655     }
1656 }
1657 }

```

(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`.

```

1658 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1659 {
1660     \bool_set_true:N \l__enumext_resume_active_bool
1661     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1662     \tl_if_empty:NTF \l__enumext_resume_name_tl
1663     {
1664         \__enumext_resume_counter:
1665     }
1666     {
1667         \__enumext_resume_counter_series:
1668     }
1669     \__enumext_resume_counter_save_ans:
1670 }

```

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.

```

1671 \cs_new_protected:Nn \__enumext_resume_counter:
1672 {
1673     \bool_if:NT \g__enumext_standar_bool
1674     {
1675         \int_gincr:N \g__enumext_resume_int
1676         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1677     }
1678     \bool_if:NT \g__enumext_starred_bool
1679     {
1680         \int_gincr:N \g__enumext_resume_vii_int
1681         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1682     }
1683 }

```

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.

```

1684 \cs_new_protected:Nn \__enumext_resume_counter_series:
1685 {
1686     \bool_if:NT \g__enumext_standar_bool
1687     {
1688         \int_set:Nn \l__enumext_start_i_int

```

```

1689         {
1690             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1691         }
1692     }
1693     \bool_if:NT \g__enumext_starred_bool
1694     {
1695         \int_set:Nn \l__enumext_start_vii_int
1696         {
1697             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1698         }
1699     }
1700 }

```

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.

```

1701 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1702 {
1703     \bool_lazy_and:nnT
1704     { \bool_if_p:N \l__enumext_standar_first_bool }
1705     { \bool_if_p:N \l__enumext_store_active_bool }
1706     {
1707         \int_set:Nn \l__enumext_start_i_int
1708         {
1709             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1710         }
1711     }
1712     \bool_lazy_and:nnT
1713     { \bool_if_p:N \l__enumext_starred_first_bool }
1714     { \bool_if_p:N \l__enumext_store_active_bool }
1715     {
1716         \int_set:Nn \l__enumext_start_vii_int
1717         {
1718             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1719         }
1720     }
1721 }

```

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

#### 11.22.4 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.

```

1722 \cs_new_protected:Nn \__enumext_resume_starred:
1723 {
1724     \bool_if:NT \g__enumext_standar_bool
1725     {
1726         \tl_if_empty:NF \g__enumext_standar_series_tl
1727         {
1728             \__enumext_resume_counter:n { }
1729             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1730         }
1731     }
1732     \bool_if:NT \g__enumext_starred_bool
1733     {
1734         \tl_if_empty:NF \g__enumext_starred_series_tl
1735         {
1736             \__enumext_resume_counter:n { }
1737             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1738         }
1739     }
1740 }

```

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

#### 11.23 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.

### 11.23.1 Setting save-ans key

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

```

1741 \cs_set_protected:Npn \__enumext_tmp:n #1
1742 {
1743   \keys_define:nn { enumext / #1 }
1744   {
1745     save-ans .code:n = \__enumext_storing_set:n {##1},
1746     save-ans .value_required:n = true,
1747   }
1748 }
1749 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} }

```

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

### 11.23.2 Internal functions for save-ans key

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

```

1750 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1751 {
1752   \msg_term:nnVV { enumext } { save-ans-log }
1753   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1754 }
1755 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1756 {
1757   \msg_term:nnVV { enumext } { save-ans-log-hook }
1758   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1759 }

```

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

`\__enumext_storing_set:n` 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.

```

1760 \cs_new_protected:Npn \__enumext_storing_set:n #1
1761 {
1762   \tl_set:Nx \l__enumext_store_name_tl {#1}
1763   \tl_if_empty:NTF \l__enumext_store_name_tl
1764   {
1765     \bool_lazy_or:nnT
1766     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1767     {
1768       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1769     }
1770   }
1771   {
1772     \bool_lazy_or:nnT
1773     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1774     {
1775       \__enumext_start_save_ans_msg:
1776       \__enumext_storing_exec:
1777     }
1778   }
1779 }

```

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*` (§11.27). 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*`.

```

1780 \cs_new_protected:Nn \__enumext_storing_exec:

```

```

1781 {
1782   \bool_set_true:N \__enumext_store_active_bool
1783   \bool_set_true:N \__enumext_check_answers_bool
1784   \tl_gset:NV \g__enumext_store_name_tl \__enumext_store_name_tl
1785   \__enumext_anskey_env_make:V \__enumext_store_name_tl
1786   \prop_if_exist:cF { g__enumext_ \__enumext_store_name_tl _prop }
1787   {
1788     \msg_log:nnV { enumext } { store-prop } \__enumext_store_name_tl
1789     \prop_new:c { g__enumext_ \__enumext_store_name_tl _prop }
1790   }
1791   \seq_if_exist:cF { g__enumext_ \__enumext_store_name_tl _seq }
1792   {
1793     \msg_log:nnV { enumext } { store-seq } \__enumext_store_name_tl
1794     \seq_new:c { g__enumext_ \__enumext_store_name_tl _seq }
1795   }
1796   \int_if_exist:cF { g__enumext_resume_ \__enumext_store_name_tl _int }
1797   {
1798     \msg_log:nnV { enumext } { store-int } \__enumext_store_name_tl
1799     \int_new:c { g__enumext_resume_ \__enumext_store_name_tl _int }
1800   }
1801 }

```

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

### 11.23.3 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.

### 11.23.4 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
1802 \cs_set_protected:Npn \__enumext_tmp:n #1
1803 {
1804   \keys_define:nn { enumext / #1 }
1805   {
1806     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1807     check-ans .initial:n = false,
1808     check-ans .value_required:n = true,
1809     no-store .code:n = {
1810       \bool_set_false:N \__enumext_check_answers_bool
1811       \bool_set_false:N \__enumext_check_ans_key_bool
1812     },
1813     no-store .value_forbidden:n = true,

```

```

1814     }
1815 }
1816 \clist_map_inline:nn
1817 {
1818     level-1, level-2, level-3, level-4, enumext*
1819 }
1820 { \__enumext_tmp:n {#1} }

```

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

### 11.23.5 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.

```

1821 \cs_new_protected:Nn \__enumext_check_ans_active:
1822 {
1823     \tl_if_empty:NF \l__enumext_store_name_tl
1824     {
1825         \bool_if:NT \l__enumext_check_answers_bool
1826         {
1827             \__enumext_check_ans_level:
1828         }
1829     }
1830 }

```

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.

```

1831 \cs_new_protected:Nn \__enumext_check_ans_level:
1832 {
1833     \int_case:nn { \l__enumext_level_int }
1834     {
1835         { 1 }{
1836             \bool_lazy_all:nT
1837             {
1838                 { \bool_if_p:N \g__enumext_starred_bool }
1839                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1840             }
1841             {
1842                 \int_gdecr:N \g__enumext_item_number_int
1843             }
1844         }
1845         { 2 }{
1846             \int_gdecr:N \g__enumext_item_number_int
1847         }
1848         { 3 }{
1849             \int_gdecr:N \g__enumext_item_number_int
1850         }
1851         { 4 }{
1852             \int_gdecr:N \g__enumext_item_number_int
1853         }
1854     }

```

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.

```

1855     \int_case:nn { \l__enumext_level_h_int }
1856     {
1857         { 1 }{
1858             \bool_lazy_all:nT
1859             {
1860                 { \bool_if_p:N \g__enumext_standar_bool }
1861                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1862             }
1863             {
1864                 \int_gdecr:N \g__enumext_item_number_int
1865             }
1866         }
1867     }
1868 }

```



(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.

```

1869 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1870 {
1871   \bool_lazy_and:nnT
1872     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1873     { \bool_if_p:N \g__enumext_standar_bool }
1874     {
1875       \bool_gset_true:N \g__enumext_check_ans_key_bool
1876     }
1877   \bool_lazy_and:nnT
1878     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1879     { \bool_if_p:N \g__enumext_starred_bool }
1880     {
1881       \bool_gset_true:N \g__enumext_check_ans_key_bool
1882     }
1883 }

```

(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:`.

```

1884 \cs_new_protected:Nn \__enumext_item_answer_diff:
1885 {
1886   \int_gset:Nn \g__enumext_item_answer_diff_int
1887     {
1888       \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1889     }
1890 }

```

(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:`.

```

1891 \cs_new_protected:Nn \__enumext_check_ans_show:
1892 {
1893   \int_case:nn { \g__enumext_item_answer_diff_int }
1894     {
1895       { -1 } { \__enumext_check_ans_msg_less: }
1896       { 0 } { \__enumext_check_ans_msg_same_ok: }
1897       { 1 } { \__enumext_check_ans_msg_greater: }
1898     }
1899 }
1900 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1901 {
1902   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1903     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1904 }
1905 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1906 {
1907   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1908     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1909 }
1910 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1911 {
1912   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1913     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1914 }

```

(End of definition for `\__enumext_check_ans_show:` and others.)

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

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

```

1915 \cs_new_protected:Nn \__enumext_check_ans_log:
1916 {
1917   \int_case:nn { \g__enumext_item_answer_diff_int }
1918   {
1919     { -1 } { \__enumext_check_ans_log_msg_less: }
1920     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1921     { 1 } { \__enumext_check_ans_log_msg_greater: }
1922   }
1923 }
1924 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1925 {
1926   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1927   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1928 }
1929 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1930 {
1931   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1932   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1933 }
1934 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1935 {
1936   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1937   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1938 }

```

(End of definition for `\__enumext_check_ans_log:` and others.)

### 11.23.6 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.

```

1939 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1940 {
1941   \int_compare:nNnT
1942   { \g__enumext_check_starred_cmd_int } = { 0 }
1943   {
1944     \msg_warning:nnnV
1945     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1946   }
1947   \int_compare:nNnT
1948   { \g__enumext_check_starred_cmd_int } > { 1 }
1949   {
1950     \msg_warning:nnnV
1951     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1952   }
1953   \int_gzero:N \g__enumext_check_starred_cmd_int
1954   \tl_clear:N \l__enumext_check_start_line_env_tl
1955 }

```

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

### 11.24 Writing .log, executing check-ans key and more

`\__enumext_execute_after_env:`

The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `\__enumext_item_answer_diff:` function and then will write the values of the global variables used to the .log file. If the key `check-ans` is active it will execute the function `\__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `\__enumext_check_ans_log:` and write the results in the .log file, undefine the environment `anskey*` (§11.27) 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.

```

1956 \cs_new_protected:Nn \__enumext_execute_after_env:
1957 {
1958   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1959   {
1960     \tl_if_empty:NF \g__enumext_store_name_tl

```

```

1961     {
1962         \__enumext_stop_save_ans_msg:
1963         \__enumext_item_answer_diff:
1964         \__enumext_log_global_vars:
1965         \__enumext_log_answer_vars:
1966         \bool_if:NTF \__enumext_check_ans_key_bool
1967         {
1968             \__enumext_check_ans_show:
1969         }
1970         { \__enumext_check_ans_log: }
1971         \__enumext_undefine_anskey_env:
1972     }
1973     \__enumext_reset_global_vars:
1974 }
1975 }

```

• This function is passed to the function `\__enumext_after_env:nn` for the environments `enumext` (§11.34) and `enumext*` (§11.37) and it is executed only when the environments are not nested or at some level of these..

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

## 11.25 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*`.

```

1976 \cs_set_protected:Npn \__enumext_tmp:n #1
1977 {
1978     \keys_define:nn { enumext / #1 }
1979     {
1980         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1981         wrap-ans .initial:n = \fbox{##1},
1982         wrap-ans .value_required:n = true,
1983         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1984         wrap-opt .initial:n = [{##1}],
1985         wrap-opt .value_required:n = true,
1986         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1987         save-sep .initial:n = {, ~ },
1988         save-sep .value_required:n = true,
1989         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1990         mark-ans .initial:n = \textasteriskcentered,
1991         mark-ans .value_required:n = true,
1992         mark-pos .choice:,
1993         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1994         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1995         mark-pos / unknown .code:n =
1996             \msg_error:nnee { enumext } { unknown-choice }
1997             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
1998         mark-pos .initial:n = right,
1999         mark-pos .value_required:n = true,
2000         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2001         show-ans .initial:n = false,
2002         show-ans .value_required:n = true,
2003         show-pos .bool_set:N = \l__enumext_show_position_bool,
2004         show-pos .initial:n = false,
2005         show-pos .value_required:n = true,
2006         mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2007         mark-ref .initial:n = \textasteriskcentered,
2008         mark-ref .value_required:n = true,
2009         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2010         save-ref .initial:n = false,
2011         save-ref .value_required:n = true,
2012     }
2013 }
2014 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {##1} }

```

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

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

```

2015 \cs_set_protected:Npn \__enumext_tmp:n #1
2016 {

```

```

2017 \keys_define:nn { enumext / #1 }
2018 {
2019   mark-pos .choice:,
2020   mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2021   mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2022   mark-pos .initial:n = right,
2023   mark-pos .value_required:n = true,
2024   show-ans .bool_set:N = \l__enumext_show_answer_bool,
2025   show-ans .initial:n = false,
2026   show-ans .value_required:n = true,
2027   show-pos .bool_set:N = \l__enumext_show_position_bool,
2028   show-pos .initial:n = false,
2029   show-pos .value_required:n = true,
2030 }
2031 }
2032 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

### 11.25.1 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⟩*.

```

2033 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2034 {
2035   \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2036   {
2037     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2038     \tl_set:ce
2039       { l__enumext_store_save_key_ \__enumext_level: _tl }
2040       { \__enumext_filter_save_key:n {#1} }
2041   }
2042 }
2043 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2044 {
2045   \bool_if:NF \l__enumext_store_save_key_vii_bool
2046   {
2047     \tl_clear:N \l__enumext_store_save_key_vii_tl
2048     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2049   }
2050 }

```

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

### 11.25.2 Setting save-key key

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

save-key

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

```

2051 \cs_set_protected:Npn \__enumext_tmp:n #1
2052 {
2053   \keys_define:nn { enumext / enumext* }
2054   {
2055     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2056     save-key .value_required:n = true,
2057   }
2058   \keys_define:nn { enumext / #1 }
2059   {
2060     save-key .code:n = \__enumext_parse_save_key:n {##1},

```

```

2061         save-key .value_required:n = true,
2062     }
2063 }
2064 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

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 `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2065 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2066 {
2067     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2068     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2069     \tl_set:ce
2070     { l__enumext_store_save_key_ \__enumext_level: _tl }
2071     { \__enumext_filter_save_key:n {#1} }
2072 }
2073 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2074 {
2075     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2076     \tl_clear:N \l__enumext_store_save_key_vii_tl
2077     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2078 }

```

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

### 11.25.3 Internal functions to store optional arguments

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

```

\__enumext_filter_save_key:n
\__enumext_filter_save_key_key:n
\__enumext_filter_save_key_pair:nn
2079 \cs_new:Npn \__enumext_filter_save_key:n #1
2080 {
2081     \use:e
2082     {
2083         \keyval_parse:NNn
2084         \__enumext_filter_save_key_key:n
2085         \__enumext_filter_save_key_pair:nn {#1}
2086     }
2087 }

```

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*` and `no-store` keys.

```

2088 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2089 {
2090     \str_case:nnF {#1}
2091     {
2092         { resume } {} { resume* } {} { no-store } {}
2093     }
2094     { , { \exp_not:n {#1} } }
2095 }

```

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.

```

2096 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2097 {
2098     \str_case:nnF {#1}
2099     {
2100         { series } {} { resume } {} { save-ans } {}
2101         { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2102         { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2103         { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2104         { mini-right } {} { mini-right* } {}
2105     }
2106     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2107 }

```

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

#### 11.25.4 Function for storing content in prop list

```
\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V
```

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

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

```
2108 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2109 {
2110   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2111   {
2112     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2113   }
2114   { #1 }
2115 }
2116 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

#### 11.25.5 Function for storing content in sequence

```
\__enumext_store_addto_seq:n
\__enumext_store_addto_seq:v
\__enumext_store_addto_seq:V
```

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

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

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

```
2117 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2118 {
2119   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2120 }
2121 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

#### 11.25.6 Functions for storing the list structure in the sequence

```
\__enumext_store_level_open:
\__enumext_store_level_close:
```

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

```
2122 \cs_new_protected:Nn \__enumext_store_level_open:
2123 {
2124   \bool_if:NT \l__enumext_check_answers_bool
2125   {
2126     \tl_if_empty:cTF { l__enumext_store_save_key_ \l__enumext_level: _tl }
2127     {
2128       \__enumext_store_addto_seq:n
2129       {
2130         \item \begin{enumext}
2131       }
2132     }
2133     {
2134       \tl_put_left:cn { l__enumext_store_save_key_ \l__enumext_level: _tl }
2135       {
2136         \item \begin{enumext} [
2137       }
2138       \tl_put_right:cn { l__enumext_store_save_key_ \l__enumext_level: _tl }
2139       {
2140       ]
2141     }
2142     \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \l__enumext_level: _tl }
2143   }
2144 }
2145 }
2146 \cs_new_protected:Nn \__enumext_store_level_close:
2147 {
2148   \bool_if:NT \l__enumext_check_answers_bool
2149   {
2150     \__enumext_store_addto_seq:n { \end{enumext} }
2151   }
2152 }
```

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

```

\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:

```

When nesting the `enumext*` environment in `enumext` starting right after `\item` (without material between them) there is a problem with the alignment of the labels with the baseline between the two environments. One way to get around this problem is to place `\mode_leave_vertical:` and then apply `\vspace` taking into account `\baselineskip`, the value of `\parsep` of the current level of `enumext` and the value of `\topsep` of the `enumext*` environment.

```

2153 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2154 {
2155   \bool_if:NT \l__enumext_check_answers_bool
2156   {
2157     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2158     {
2159       \__enumext_store_addto_seq:n
2160       {
2161         \item \mode_leave_vertical:
2162         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2163         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2164       }
2165     }
2166     {
2167       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2168       {
2169         \item \mode_leave_vertical:
2170         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2171         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2172       }
2173       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2174       {
2175       ]
2176     }
2177     \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2178   }
2179 }
2180 }
2181 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2182 {
2183   \bool_if:NT \l__enumext_check_answers_bool
2184   {
2185     \__enumext_store_addto_seq:n { \end{enumext*} }
2186   }
2187 }

```

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

### 11.25.7 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`

```

2188 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2189 {
2190   \mode_leave_vertical:
2191   \skip_horizontal:n { -\dim_use:N #2 }
2192   \makebox[\opt][ r ]
2193   {
2194     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2195     {
2196       \tl_use:N \l__enumext_mark_answer_sym_tl
2197     }
2198   }
2199   \skip_horizontal:n { \dim_use:N #2 }
2200 }
2201 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for `\__enumext_print_keyans_box:NN.`)



## 11.26 The command `\anskey` and internal label and ref

Since we will be “*storing content*” in a list environment within `\sequences` and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing. The `\anskey` command will cover this point and give it similar behaviour to that of `\item` in the `enumext` and `enumext*` environments executed as follows: `\anskey[⟨key = val⟩]{⟨content⟩}` so first we’ll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```
2202 \keys_define:nn { enumext / anskey }
2203 {
2204     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2205     break-col .default:n = true,
2206     break-col .value_forbidden:n = true,
2207     item-join .int_set:N = \l__enumext_store_item_join_int,
2208     item-join .value_required:n = true,
2209     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2210     item-star .default:n = true,
2211     item-star .value_forbidden:n = true,
2212     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2213     item-sym* .value_required:n = true,
2214     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2215     item-pos* .value_required:n = true,
2216 }
```

🔗 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 we call the function `\__enumext_store_anskey_code:nn`.

```
2217 \NewDocumentCommand \anskey { o +m }
2218 {
2219     \__enumext_anskey_safe_outer:
2220     \group_begin:
2221         \bool_if:NT \l__enumext_check_answers_bool
2222         {
2223             \int_gincr:N \g__enumext_item_anskey_int
2224             \__enumext_anskey_safe_inner:n {#2}
2225             \__enumext_store_anskey_code:nn {#1} {#2}
2226         }
2227     \group_end:
2228 }
```

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

### 11.26.1 Internal functions for the command

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

`\__enumext_anskey_safe_inner:n`

```
2229 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2230 {
2231     \bool_if:NF \l__enumext_store_active_bool
2232     {
2233         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2234     }
2235     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2236     {
2237         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2238     }
2239     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2240     {
2241         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2242     }
2243     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2244     {
2245         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2246     }
2247 }
```

The `\__enumext_anskey_safe_inner:n` function will first check to see if the passed argument is empty and then check to see if the command is nested by returning the appropriate messages.

```

2248 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2249 {
2250   \tl_if_empty:nT {#1}
2251   {
2252     \msg_error:nn { enumext } { anskey-empty-arg }
2253   }
2254   \int_incr:N \__enumext_anskey_level_int
2255   \int_compare:nNnT { \__enumext_anskey_level_int } > { 1 }
2256   {
2257     \msg_error:nn { enumext } { anskey-nested }
2258   }
2259 }

```

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

`\__enumext_store_anskey_code:nn`

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

```

2260 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2261 {
2262   \__enumext_store_addto_prop:n {#2}
2263   \bool_if:NT \l__enumext_store_ref_key_bool
2264   {
2265     \__enumext_store_internal_ref:
2266   }
2267   \__enumext_anskey_show_wrap_left:n { #2 }

```

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

```

2268   \tl_if_novalue:nF {#1}
2269   {
2270     \keys_set:nn { enumext / anskey } {#1}
2271   }
2272   \tl_clear:N \l__enumext_store_anskey_arg_tl
2273   \bool_lazy_and:nnT
2274   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2275   { \bool_not_p:n { \l__enumext_starred_bool } }
2276   {
2277     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2278   }
2279   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2280   \bool_lazy_and:nnT
2281   { \bool_not_p:n { \l__enumext_starred_bool } }
2282   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2283   {
2284     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2285     {
2286       ( \exp_not:V \l__enumext_store_item_join_int )
2287     }
2288   }

```

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

```

2289   \bool_if:NFT \l__enumext_store_item_star_bool
2290   {
2291     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2292     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2293     {
2294       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2295       {
2296         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2297       }

```

```

2298     }
2299     \dim_compare:nT
2300     {
2301         \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2302     }
2303     {
2304         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2305         {
2306             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2307         }
2308     }
2309     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2310 }
2311 {
2312     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2313 }

```

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

```

2314     \bool_lazy_and:nnT
2315     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2316     { \bool_if_p:N \l__enumext_hyperref_bool }
2317     {
2318         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2319         {
2320             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2321             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2322         }
2323     }
2324     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2325 }

```

(End of definition for `\__enumext_store_anskey_code:nn`.)

`\__enumext_store_internal_ref:`

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

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

```

2326 \cs_new_protected:Nn \__enumext_store_internal_ref:
2327 {
2328     \cs_set_protected:Npn \__enumext_tmp:n ##1
2329     {
2330         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2331         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2332         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2333         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2334     }
2335     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2336     \cs_set:Npn \__enumext_tmp:n ##1
2337     { . \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` is running alone in it or if it is running in a nested `enumext` environment within the starting environment.

```

2338     \bool_lazy_all:nT
2339     {
2340         { \bool_if_p:N \g__enumext_starred_bool }
2341         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2342     }
2343     {
2344         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2345         { \tl_use:N \l__enumext_label_copy_vii_tl }
2346     }
2347     \bool_lazy_all:nT
2348     {
2349         { \bool_if_p:N \l__enumext_standar_bool }
2350         { \bool_if_p:N \g__enumext_starred_bool }
2351         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2352     }
2353     {
2354         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

2355         {
2356             \tl_use:N \l__enumext_label_copy_vii_tl
2357             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2358         }
2359     }

```

If started with `enumext` and if `\anskey` is running alone in it or if it is running in a nested `enumext*` environment within the starting environment.

```

2360     \bool_lazy_all:nT
2361     {
2362         { \bool_if_p:N \l__enumext_standar_bool }
2363         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2364         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2365         { \bool_not_p:n { \l__enumext_starred_bool } }
2366     }
2367     {
2368         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2369         {
2370             \tl_use:N \l__enumext_label_copy_i_tl
2371             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2372         }
2373     }
2374     \cs_set:Npn \__enumext_tmp:n ##1
2375     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2376     \bool_lazy_all:nT
2377     {
2378         { \bool_if_p:N \l__enumext_standar_bool }
2379         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2380         { \bool_not_p:n { \g__enumext_starred_bool } }
2381         { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2382     }
2383     {
2384         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2385         {
2386             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2387             . \tl_use:N \l__enumext_label_copy_vii_tl
2388         }
2389     }

```

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

```

2390     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2391     {
2392         \l__enumext_store_name_tl \c_colon_str
2393         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2394     }

```

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

```

2395     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2396     {
2397         \__enumext_newlabel:nn
2398         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2399         { \l__enumext_newlabel_arg_two_tl }
2400     }
2401     \l__enumext_write_aux_file_tl
2402 }

```

(End of definition for `\__enumext_store_internal_ref:`)

`\__enumext_anskey_show_wrap_arg:n`

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

```

2403 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2404 {
2405     \par
2406     \bool_if:NT \l__enumext_starred_bool
2407     {
2408         \cs_set:Nn \__enumext_level: { vii }
2409     }
2410     \__enumext_print_keyans_box:cc
2411     { \l__enumext_labelwidth_ \__enumext_level: _dim }
2412     { \l__enumext_labelsep_ \__enumext_level: _dim }

```

```

2413   \__enumext_anskey_wrapper:n { #1 }
2414 }

```

(End of definition for \\_\_enumext\_anskey\_show\_wrap\_arg:n.)

\\_\_enumext\_anskey\_show\_wrap\_left:n

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

```

2415 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2416 {
2417   \bool_if:NT \__enumext_show_answer_bool
2418   {
2419     \__enumext_anskey_show_wrap_arg:n { #1 }
2420   }
2421   \bool_if:NT \__enumext_show_position_bool
2422   {
2423     \tl_set:Nx \__enumext_mark_answer_sym_tl
2424     {
2425       \group_begin:
2426       \exp_not:N \normalfont
2427       \exp_not:N \footnotesize [ \int_eval:n
2428       {
2429         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
2430       }
2431       ]
2432       \group_end:
2433     }
2434     \__enumext_anskey_show_wrap_arg:n { #1 }
2435   }
2436 }

```

(End of definition for \\_\_enumext\_anskey\_show\_wrap\_left:n.)

## 11.27 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: (§11.24) which is executed after the environment in which the key `save-ans` is active.

```

2437 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2438 {
2439   \cs_undefine:c { anskey* }
2440   \cs_undefine:c { endanskey* }
2441   \cs_undefine:c { __scontents_anskey*_env_begin: }
2442   \cs_undefine:c { __scontents_anskey*_env_end: }
2443 }

```

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

```

2444 \__enumext_before_env:nn { enumext }
2445 {
2446   \int_compare:nNt { \__enumext_level_int } = { 0 }
2447   {
2448     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2449     {
2450       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2451     }
2452   }

```

```

2453     }
2454     \__enumext_before_env:nn { enumext* }
2455     {
2456         \int_compare:nNt { \__enumext_level_int } = { 0 }
2457         {
2458             \cs_if_free:cF { __scontents_anskey*_env_begin: }
2459             {
2460                 \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2461             }
2462         }
2463     }

```

Detection of the `anskey*` environment inside the `keyans`, `keyans*` and `keyanspic` environments.

```

2464     \__enumext_before_env:nn { anskey* }
2465     {
2466         \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2467         {
2468             \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2469         }
2470         \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2471         {
2472             \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2473         }
2474         \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2475         {
2476             \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2477         }
2478     }

```

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

**anskey\***

The function `\__enumext_anskey_env_make:n` creates the environment `anskey*` (custom version of `scontents` environment) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`. To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `\__enumext_storing_exec: ($11.23.1)`.

```

2479 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2480 {
2481     \newenvvsc{anskey*}[store-env=#1,print-env=false]
2482     \__enumext_anskey_env_exec:
2483 }
2484 \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`.

```

2485 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2486 {
2487     \keys_define:nn { scontents / scontents }
2488     {
2489         break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2490         break-col .default:n = true,
2491         break-col .value_forbidden:n = true,
2492         item-join .int_gset:N = \g__enumext_store_item_join_int,
2493         item-join .value_required:n = true,
2494         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2495         item-star .default:n = true,
2496         item-star .value_forbidden:n = true,
2497         item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2498         item-sym* .value_required:n = true,
2499         item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2500         item-pos* .value_required:n = true,
2501         print-env .undefine:,
2502         store-env .undefine:,
2503         write-out .undefine:,
2504     }
2505 }

```

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

```

2506 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2507 {
2508   \keys_define:nn { scontents / scontents }
2509   {
2510     break-col .undefine:,
2511     item-join .undefine:,
2512     item-star .undefine:,
2513     item-sym* .undefine:,
2514     item-pos* .undefine:,
2515     write-out .code:n = {
2516       \bool_set_false:N \l__scontents_storing_bool
2517       \bool_set_true:N \l__scontents_writing_bool
2518       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2519     },
2520     write-out .value_required:n = true,
2521     print-env .meta:nn = { scontents } { print-env = ##1 },
2522     print-env .default:n = true,
2523     store-env .meta:nn = { scontents } { store-env = ##1 },
2524     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} }
2525   }
2526 }

```

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*.

```

2527 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2528 {
2529   \group_begin:
2530   \int_set:Nn \tex_newlinechar:D { `^^J }
2531   \__scontents_rescan_tokens:x
2532   {
2533     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2534     #1
2535   }
2536 }

```

(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*.

```

2537 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2538 {
2539   \__enumext_before_env:nn { anskey* }
2540   {
2541     \__enumext_anskey_env_define_keys:
2542   }

```

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.

```

2543   \hook_if_empty:nF {env/anskey*/after}
2544   {
2545     \hook_gremove_code:nn {env/anskey*/after} { * }
2546   }
2547   \__enumext_after_env:nn { anskey* }
2548   {
2549     \tl_clear:N \l__enumext_store_anskey_env_tl
2550     \tl_clear:N \l__enumext_store_anskey_opt_tl
2551     \tl_gset:Ne \l__enumext_store_anskey_env_tl
2552     {
2553       \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2554     }
2555     \__enumext_anskey_env_keys:
2556     \__enumext_anskey_env_store:
2557     \__enumext_anskey_env_clean:
2558     \__enumext_anskey_env_undefine_keys:
2559   }
2560 }

```

🔗 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_undefine_keys:` is necessary so as not to hinder any *scontents* environment running within *enumext* or *enumext\**.



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

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

```

2561 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2562 {
2563   \bool_lazy_and:nnT
2564     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2565     { \bool_not_p:n { \l__enumext_starred_bool } }
2566     {
2567       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2568     }

```

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

```

2569   \bool_lazy_and:nnT
2570     { \bool_not_p:n { \l__enumext_starred_bool } }
2571     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2572     {
2573       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2574         {
2575           ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2576         }
2577     }

```

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

```

2578   \bool_if:NT \g__enumext_store_item_star_bool
2579   {
2580     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2581       {
2582         ,item-star,
2583       }
2584     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2585     {
2586       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2587         {
2588           ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2589         }
2590     }
2591     \dim_compare:nT
2592     {
2593       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2594     }
2595     {
2596       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2597         {
2598           ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2599         }
2600     }
2601   }
2602 }

```

The function `\__enumext_anskey_env_store:` will be responsible for storing the content of the environment, we will execute the code within a group and only if the variable `\l__enumext_store_anskey_env_tl` is not empty using the function `\__enumext_rescan_anskey_env:n` from package `scontents`.

```

2603 \cs_new_protected:Nn \__enumext_anskey_env_store:
2604 {
2605   \group_begin:
2606     \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2607     {
2608       \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2609       {
2610         \exp_args:Ne
2611         \anskey
2612         {
2613           \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2614         }
2615       }
2616     }

```

```

2617         \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2618         \exp_args:Ne
2619         \anskey
2620         {
2621             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2622         }
2623     }
2624 }
2625 \group_end:
2626 }

```

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

```

2627 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2628 {
2629     \bool_gset_false:N \g__enumext_store_columns_break_bool
2630     \int_gzero:N       \g__enumext_store_item_join_int
2631     \bool_gset_false:N \g__enumext_store_item_star_bool
2632     \tl_gclear:N       \g__enumext_store_item_symbol_tl
2633     \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2634 }

```

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

## 11.28 Common functions for keyans, keyans\* and keyanspic

### 11.28.1 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`.

```

2635 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2636 {
2637     \tl_clear:N \l__enumext_store_current_label_tl
2638     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2639     {
2640         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2641     }
2642     {
2643         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2644     }
2645     \tl_if_novalue:nF { #1 }
2646     {
2647         % Set save-sep
2648         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2649         {
2650             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2651         }
2652         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2653     }
2654     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2655 }

```

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

### 11.28.2 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.

```

2656 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2657 {
2658     \bool_if:NT \l__enumext_store_ref_key_bool
2659     {

```

```

2660     \cs_set_protected:Npn \__enumext_tmp:n #1
2661     {
2662         \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2663         \tl_reverse:c { \__enumext_label_copy_##1_tl }
2664         \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2665         \tl_reverse:c { \__enumext_label_copy_##1_tl }
2666     }
2667     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2668     \__enumext_keyans_store_ref_aux_i:
2669 }
2670 }

```

The auxiliary function `\__enumext_keyans_store_ref_aux_i:` set the variable `\l__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`.

```

2671 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2672 {
2673     \bool_if:NT \g__enumext_starred_bool
2674     {
2675         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2676     }
2677     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2678     {
2679         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2680         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2681     }
2682     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2683     {
2684         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2685         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2686     }
2687     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2688     {
2689         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2690         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2691     }
2692     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2693     {
2694         \l__enumext_store_name_tl \c_colon_str
2695         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2696     }
2697     \__enumext_keyans_store_ref_aux_ii:
2698 }

```

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

```

2699 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2700 {
2701     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2702     {
2703         \__enumext_newlabel:nn
2704         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2705         { \l__enumext_newlabel_arg_two_tl }
2706     }
2707     \l__enumext_write_aux_file_tl
2708 }

```

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

### 11.28.3 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  $\langle label \rangle$  `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the  $\langle contents \rangle$  of the optional argument of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

2709 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2710 {
2711     \tl_clear:N \l__enumext_store_current_label_tl
2712     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2713     {

```

```

2714     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2715   }
2716   {
2717     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2718   }
2719   \tl_if_novalue:nF { #1 }
2720   {
2721     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2722     {
2723       \tl_put_right:Ne \l__enumext_store_current_label_tl
2724       {
2725         \l__enumext_store_keyans_item_opt_sep_tl
2726       }
2727     }
2728     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2729   }
2730   \__enumext_keyans_addto_seq_link:
2731 }

```

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.

```

2732 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2733 {
2734   \bool_lazy_and:nnT
2735   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2736   { \bool_if_p:N \l__enumext_hyperref_bool }
2737   {
2738     \tl_put_right:Ne \l__enumext_store_current_label_tl
2739     {
2740       \hfill \exp_not:N \hyperlink
2741       {
2742         \exp_not:V \l__enumext_newlabel_arg_one_tl
2743       }
2744       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2745     }
2746   }
2747   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2748   \bool_if:NT \l__enumext_check_answers_bool
2749   {
2750     \int_gincr:N \g__enumext_item_anskey_int
2751   }
2752 }

```

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

#### 11.28.4 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.

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.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:
2753 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2754 {
2755   \tl_if_novalue:nF { #1 }
2756   {
2757     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2758   }
2759   \bool_if:NT \l__enumext_show_answer_bool
2760   {
2761     \__enumext_keyans_show_ans:
2762   }
2763   \bool_if:NT \l__enumext_show_position_bool
2764   {
2765     \__enumext_keyans_show_pos:
2766   }
2767 }
2768 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:

```

```

2769 {
2770   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2771   {
2772     \bool_lazy_or:nnT
2773     { \bool_if_p:N \l__enumext_show_answer_bool }
2774     { \bool_if_p:N \l__enumext_show_position_bool }
2775     {
2776       \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2777     }
2778   }
2779 }
2780 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2781 {
2782   \tl_put_left:Nn \l__enumext_label_v_tl
2783   {
2784     \__enumext_print_keyans_box:NN
2785     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2786   }
2787 }
2788 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2789 {
2790   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2791   {
2792     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2793     {
2794       \group_begin:
2795       \exp_not:N \normalfont
2796       \exp_not:N \footnotesize [ \int_eval:n
2797       {
2798         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2799       }
2800       ]
2801       \group_end:
2802     }
2803   }
2804   {
2805     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2806     {
2807       \group_begin:
2808       \exp_not:N \normalfont
2809       \exp_not:N \footnotesize [ \int_eval:n
2810       {
2811         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2812       }
2813       ]
2814       \group_end:
2815     }
2816   }
2817   \tl_put_left:Nn \l__enumext_label_v_tl
2818   {
2819     \__enumext_print_keyans_box:NN
2820     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2821   }
2822 }

```

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

## 11.29 Setting `item-sym*` and `item-pos*` keys

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

<code>item-sym*</code> <code>item-pos*</code>	Define and set <code>item-sym*</code> and <code>item-pos*</code> keys for <code>enumext</code> and <code>enumext*</code> .
--------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------

```

2823 \cs_set_protected:Npn \__enumext_tmp:n #1 #2
2824 {
2825   \keys_define:nn { enumext / #1 }
2826   {
2827     item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2828     item-sym* .value_required:n = true,
2829     item-sym* .initial:n = { $\star$ },
2830     item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2831     item-pos* .value_required:n = true,

```

```

2832     }
2833   }
2834   \clist_map_inline:nn
2835   {
2836     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2837   }
2838   { \__enumext_tmp:nn #1 }

```

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

### 11.30 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 with the `mini-env` key and in the `enumext*` and `keyans*` environments, it is necessary to redefine the command. This implementation is adapted from the answer given by Clea F. Rees (@cfr) in [footnotes in boxes compatible with hyperref](#).

```

2839 \cs_new_protected:Nn \__enumext_footnotetext:nn
2840 {
2841   \footnotetext[#1]{#2}
2842 }
2843 \cs_new_protected:Nn \__enumext_renew_footnote:
2844 {
2845   \seq_gclear:N \g__enumext_footnote_arg_seq
2846   \seq_gclear:N \g__enumext_footnote_int_seq
2847   \RenewDocumentCommand \footnote { o +m }
2848   {
2849     \tl_if_novalue:nTF {##1}
2850     {
2851       \stepcounter{footnote}
2852       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2853     }
2854     {
2855       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2856     }
2857     \footnotemark [ \g__enumext_footnote_int ]
2858     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2859     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2860   }
2861 }
2862 \cs_new_protected:Nn \__enumext_print_footnote:
2863 {
2864   \seq_if_empty:NF \g__enumext_footnote_int_seq
2865   {
2866     \seq_map_pairwise_function:NNN
2867     \g__enumext_footnote_int_seq
2868     \g__enumext_footnote_arg_seq
2869     \__enumext_footnotetext:nn
2870   }
2871 }

```

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

### 11.31 Redefining `\item` command

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

#### 11.31.1 The `\item` command in `enumext`

```

\__enumext_default_item:n

```

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

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `\l__enumext_check_ans_key_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `\__enumext_item_std:w`.

Otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `\__enumext_item_std:w` with the optional argument.

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `\__enumext_make_label:` (§11.32).

```

2872 \cs_new_protected:Npn \__enumext_default_item:n #1
2873 {
2874   \tl_if_novalue:nTF {#1}
2875   {
2876     \bool_if:NT \l__enumext_check_answers_bool

```

```

2877     {
2878         \int_gincr:N \g__enumext_item_number_int
2879     }
2880     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2881     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2882 }
2883 {
2884     \bool_set_eq:cc
2885     { l__enumext_wrap_label_ \__enumext_level: _bool }
2886     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2887     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2888 }
2889 }

```

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

`\__enumext_starred_item:nn`

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as optional argument in the global variable `\g__enumext_item_symbol_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item*-sep` or by the second optional argument.

Then we will see the state of the variable `\l__enumext_check_ans_key_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `\__enumext_item_std:w`.

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `\__enumext_make_label:` (§11.32).

```

2890 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2891 {
2892     \tl_if_novalue:nF {#1}
2893     {
2894         \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2895     }
2896     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2897     \tl_if_novalue:nTF {#2}
2898     {
2899         \dim_set_eq:cc
2900         { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2901         { l__enumext_labelsep_ \__enumext_level: _dim }
2902     }
2903     {
2904         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2905     }
2906     \bool_if:NT \l__enumext_check_answers_bool
2907     {
2908         \int_gincr:N \g__enumext_item_number_int
2909     }
2910     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2911     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2912 }

```

(End of definition for `\__enumext_starred_item:nn`.)

`\__enumext_redefine_item:`

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

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.33.2).

```

2913 \cs_new_protected:Nn \__enumext_redefine_item:
2914 {
2915     \RenewDocumentCommand \item { s o o }
2916     {

```



```

2917     \bool_if:nTF {##1}
2918     {
2919         \__enumext_starred_item:n {##2} {##3}
2920     }
2921     { \__enumext_default_item:n {##2} }
2922 }
2923 }

```

(End of definition for `\__enumext_redefine_item:`)

### 11.31.2 The `\item` command in keyans

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

`\__enumext_keyans_default_item:n`

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

```

2924 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2925 {
2926     \tl_if_novalue:nTF { #1 }
2927     {
2928         \bool_set_true:N \l__enumext_wrap_label_v_bool
2929         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2930     }
2931     {
2932         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2933         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2934     }
2935 }

```

(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  $\langle label \rangle$ , 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  $\langle label \rangle$ .

```

2936 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2937 {
2938     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
2939     \__enumext_keyans_show_left:n { #1 }
2940     \bool_set_true:N \l__enumext_wrap_label_v_bool
2941     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

2942     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
2943     \__enumext_keyans_addto_prop:n { #1 }
2944     \__enumext_keyans_store_ref:
2945     \__enumext_keyans_addto_seq:n { #1 }
2946     \int_gincr:N \g__enumext_check_starred_cmd_int
2947 }

```

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

`\item*`

`\__enumext_keyans_redefine_item:`

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

This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.33.2).

```

2948 \cs_new_protected:Npn \__enumext_keyans_redefine_item:
2949 {
2950     \RenewDocumentCommand \item { s o }
2951     {
2952         \bool_if:nTF {##1}
2953         {
2954             \peek_remove_spaces:n
2955             {
2956                 \__enumext_keyans_starred_item:n {##2}
2957             }
2958         }

```

```

2959         {
2960             \__enumext_keyans_default_item:n {##2}
2961         }
2962     }
2963 }

```

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

### 11.32 Redefining `\makelabel` command

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

#### 11.32.1 Redefining `\makelabel` for `enumext`

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

```

2964 \cs_new_protected:Nn \__enumext_item_starred:
2965 {
2966     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2967     {
2968         \mode_leave_vertical:
2969         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2970         \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
2971         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2972     }
2973 }

```

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

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

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.33.2).

```

2974 \cs_new_protected:Nn \__enumext_make_label:
2975 {
2976     \RenewDocumentCommand \makelabel { m }
2977     {
2978         \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2979         \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2980         \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2981         {
2982             \__enumext_item_starred:
2983             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2984         }
2985         { ##1 }
2986         \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2987         \tl_gclear:N \g__enumext_item_symbol_tl
2988     }
2989 }

```

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

#### 11.32.2 Redefining `\makelabel` for `keyans`

`\__enumext_keyans_make_label:` The function `\__enumext_keyans_make_label:` redefine `\makelabel` for `keyans` environment.

This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.33.2).

```

2990 \cs_new_protected:Nn \__enumext_keyans_make_label:
2991 {
2992     \RenewDocumentCommand \makelabel { m }
2993     {
2994         \tl_use:N \l__enumext_label_fill_left_v_tl
2995         \tl_use:N \l__enumext_label_font_style_v_tl
2996         \bool_if:NTF \l__enumext_wrap_label_v_bool
2997         {
2998             \__enumext_wrapper_label_v:n { ##1 }
2999         }
3000         { ##1 }
3001         \tl_use:N \l__enumext_label_fill_right_v_tl
3002     }
3003 }

```

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

### 11.33 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`.

#### 11.33.1 Calculation of `\leftmargin` and `\itemindent`

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

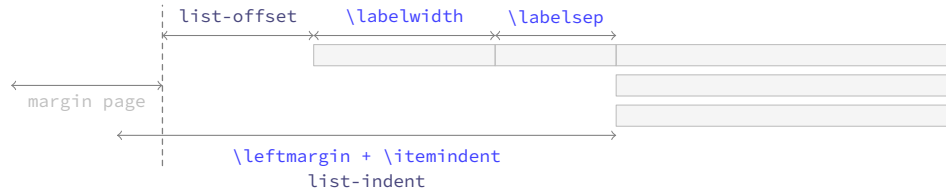


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

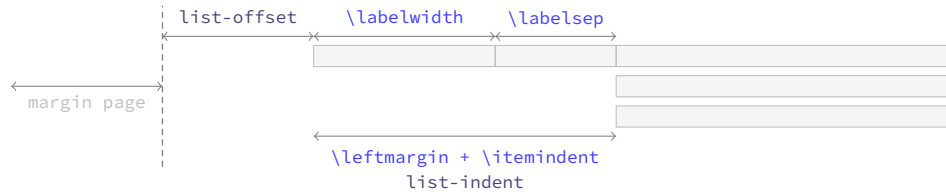


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

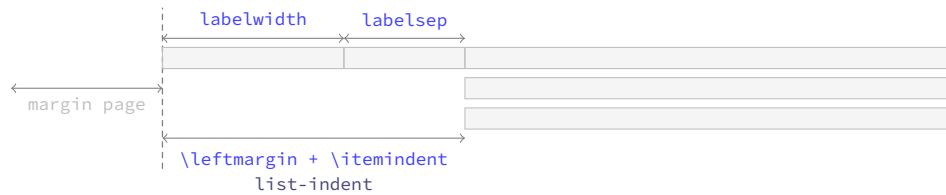


Figure 11: Default horizontal lengths in `enumext`.

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc
```

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

```
#1: \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 (§11.33.2).

```
3004 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3005 {
3006   \dim_compare:nNt { #1 } < { \c_zero_dim }
3007   {
3008     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3009     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3010   }
3011   \dim_compare:nNt { #2 } < { \c_zero_dim }
3012   {
3013     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3014     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3015   }
3016 }
```

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

```
3016 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```

3017 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3018 {
3019   \dim_set:Nn #6 { #1 + #2 - #4 }
3020   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3021 }
3022 {
3023   \dim_compare:nNnT { #4 } = { #1 + #2 }
3024   { \dim_set:Nn #6 { \c_zero_dim } }
3025   \dim_compare:nNnT { #4 } < { #1 + #2 }
3026   { \dim_set:Nn #6 { #1 + #2 - #4 } }
3027   \dim_compare:nNnT { #4 } > { #1 + #2 }
3028   {
3029     \dim_set:Nn #6 { -#1 - #2 + #4 }
3030     \dim_set:Nn #6 { #6*-1 }
3031   }
3032   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3033 }
3034 }
3035 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

### 11.33.2 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.

```

3036 \cs_set_protected:Npn \__enumext_tmp:n #1
3037 {
3038   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
3039   {
3040     \__enumext_calc_hspace:ccccc
3041     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3042     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3043     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3044     { \__enumext_leftmargin_tmp_#1_bool }
3045     \clist_map_inline:nn
3046       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3047       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3048     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3049       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3050     \usecounter { enumX#1 }
3051     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3052     \str_if_eq:nnTF { #1 } { v }
3053     {
3054       \__enumext_keyans_redefine_item:
3055       \__enumext_keyans_make_label:
3056       \__enumext_keyans_ref:
3057       \__enumext_keyans_fake_item:
3058       \bool_if:cT { \__enumext_show_length_#1_bool }
3059       {
3060         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3061       }
3062     }
3063     {
3064       \__enumext_redefine_item:
3065       \__enumext_make_label:
3066       \__enumext_standar_ref:
3067       \__enumext_fake_item:
3068       \bool_if:cT { \__enumext_show_length_#1_bool }
3069       {
3070         \msg_term:nnne { enumext } { list-lengths } { #1 } { \int_use:N \__enumext_level_int }
3071       }
3072     }
3073   }
3074 }
3075 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n { #1 } }

```

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

`\__enumext_list_arg_two_vii:` For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `0pt`. 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.

```

3076 \cs_set_protected:Npn \__enumext_tmp:n #1
3077 {
3078   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
3079   {
3080     \__enumext_calc_hspace:ccccc
3081     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3082     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3083     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3084     { \__enumext_leftmargin_tmp_#1_bool }
3085     \clist_map_inline:nn
3086     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3087     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3088     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3089     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3090     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3091     \skip_zero:N \partopsep
3092     \usecounter { enumX#1 }
3093     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3094     \__enumext_starred_ref:
3095     \str_if_eq:nnTF {#1} { vii }
3096     {
3097       \__enumext_fake_item_vii:
3098       \bool_if:cT { \__enumext_show_length_vii_bool }
3099       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3100     }
3101     {
3102       \__enumext_fake_item_viii:
3103       \bool_if:cT { \__enumext_show_length_#1_bool }
3104       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3105     }
3106   }
3107 }
3108 \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:`)

## 11.34 The environment `enumext`

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

```

3109 \NewDocumentEnvironment{enumext}{0}{ }
3110 {
3111   \__enumext_safe_exec:
3112   \__enumext_parse_keys:n {#1}
3113   \__enumext_before_list:
3114   \__enumext_start_store_level:
3115   \__enumext_start_list:nn
3116   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3117   {
3118     \use:c { \__enumext_list_arg_two_ \__enumext_level: : }
3119     \__enumext_before_keys_exec:
3120   }
3121   \__enumext_after_args_exec:
3122 }
3123 {
3124   \__enumext_stop_list:
3125   \__enumext_stop_store_level:
3126   \__enumext_after_list:
3127 }

```

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

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

```

3128 \cs_new_protected:Nn \__enumext_safe_exec:
3129 {

```

```

3130     \__enumext_internal_mini_page:
3131     \__enumext_is_not_nested:
3132     \int_incr:N \__enumext_level_int
3133     \int_compare:nNnT { \__enumext_level_int } > { 4 }
3134     { \msg_fatal:nn { enumext } { list-too-deep } }
3135     \bool_set_true:N \__enumext_standar_bool
3136     \__enumext_is_on_first_level:
3137 }

```

(End of definition for \\_\_enumext\_safe\_exec:.)

\\_\_enumext\_parse\_keys:n The \\_\_enumext\_parse\_store\_keys:n function will parse the *⟨keys⟩* passed to the optional environment argument *enumext* by levels only if present. First we clear the variable \\_\_enumext\_series\_str and then we check if we are at the first level, if so we process the *⟨keys⟩* and then execute the function \\_\_enumext\_parse\_series:n used by the key *series*, otherwise we will pass the *⟨keys⟩* to the inner levels of the environment and finally if the variable \\_\_enumext\_store\_active\_bool established by the key *save-ans* is true we execute \\_\_enumext\_parse\_store\_keys:n used by the key *save-key*.

```

3138 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3139 {
3140     \tl_if_novalue:nF {#1}
3141     {
3142         \str_clear:N \__enumext_series_str
3143         \int_compare:nNnTF { \__enumext_level_int } = { 1 }
3144         {
3145             \keys_set:nn { enumext / level-1 } {#1}
3146             \__enumext_parse_series:n {#1}
3147         }
3148         {
3149             \exp_args:Ne \keys_set:nn
3150             { enumext / level-\int_use:N \__enumext_level_int } {#1}
3151         }
3152         \__enumext_store_active_keys:n {#1}
3153     }
3154 }

```

(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⟩* of the \anskey command. \\_\_enumext\_stop\_store\_level: If *enumext* are nested in *enumext\** add \\_\_enumext\_store\_level\_open: to preserve the stored structure.

```

3155 \cs_new_protected:Nn \__enumext_start_store_level:
3156 {
3157     \bool_lazy_all:nT
3158     {
3159         { \bool_if_p:N \__enumext_store_active_bool }
3160         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3161         { \bool_not_p:n { \g__enumext_starred_bool } }
3162     }
3163     {
3164         \int_compare:nNnT { \__enumext_level_int } > { 1 }
3165         {
3166             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3167             \__enumext_store_level_open:
3168         }
3169     }
3170     \bool_lazy_all:nT
3171     {
3172         { \bool_if_p:N \__enumext_store_active_bool }
3173         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3174         { \bool_if_p:N \g__enumext_starred_bool }
3175     }
3176     {
3177         \int_compare:nNnT { \__enumext_level_int } > { 0 }
3178         {
3179             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3180             \__enumext_store_level_open:
3181         }
3182     }
3183 }

```

```

3184 \cs_new_protected:Nn \__enumext_stop_store_level:
3185 {
3186   \bool_if:cT { \__enumext_store_upper_level_ \__enumext_level: _bool }
3187   {
3188     \__enumext_store_level_close:
3189   }
3190 }

```

(End of definition for \\_\_enumext\_start\_store\_level: and \\_\_enumext\_stop\_store\_level:.)

`\__enumext_before_list:` The function `\__enumext_before_list:` will add the vertical spacing on the environment if the `above` key is active next to the `{\code}` defined by the `before*` key if it is active.

```

3191 \cs_new_protected:Nn \__enumext_before_list:
3192 {
3193   \__enumext_vspace_above:
3194   \__enumext_before_args_exec:

```

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

```

3195   \__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.

```

3196   \dim_compare:nNnT
3197   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3198   {
3199     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3200     {
3201       \linewidth
3202       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3203       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3204     }

```

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*.

```

3205   \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3206   \int_gincr:N \g__enumext_minipage_stat_int
3207   \__enumext_mini_addvspace:
3208   \nointerlineskip\noindent
3209   \begin{\__enumext_mini_env*}
3210   { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3211   }
3212   \__enumext_multicols_start:
3213   }

```

(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.

```

3214 \cs_new_protected:Nn \__enumext_multicols_start:
3215 {
3216   \int_compare:nNnT
3217   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3218   {
3219     \dim_compare:nNnT
3220     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3221     {
3222       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
3223       {
3224         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }

```



```

3225         + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
3226       ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
3227       - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3228     }
3229   }
3230   \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
3231   \skip_zero:N \multicolsep
3232   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3233   {
3234     \dim_zero:N \columnseprule
3235   }

```

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.

```

3236   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3237   {
3238     \__enumext_multi_addvspace:
3239   }
3240   \raggedcolumns
3241   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
3242 }
3243 }

```

(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.

```

3244 \cs_new_protected:Nn \__enumext_multicols_stop:
3245 {
3246   \int_compare:nNnT
3247   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3248   {
3249     \end{multicols}
3250     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3251     {
3252       \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3253     }
3254   }
3255 }

```

(End of definition for `\__enumext_multicols_stop:`)

`\__enumext_after_list:` The function `\__enumext_after_list:` will 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* `\__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3256 \cs_new_protected:Nn \__enumext_after_list:
3257 {
3258   \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
3259   {
3260     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3261     {
3262       \msg_warning:nn { enumext } { missing-miniright }
3263       \miniright
3264     }
3265     \int_gzero:N \g__enumext_minipage_stat_int
3266     \end{\__enumext_mini_env*}
3267     \par\addvspace { \__enumext_minipage_after_skip }
3268   }
3269   { \__enumext_multicols_stop: }

```

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

```

3270   \__enumext_check_ans_key_hook:

```

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

```

3271     \__enumext_after_stop_list:
3272     \__enumext_vspace_below:
3273     \bool_set_false:N \l__enumext_standar_bool
3274     \__enumext_resume_save_counter:
3275 }

```

(End of definition for \\_\_enumext\_after\_list:.)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext` environment using the “hook” function `\__enumext_after_env:nn`.

```

3276 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

### 11.35 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.

```

3277 \NewDocumentEnvironment{keyans}{0}{}
3278 {
3279     \__enumext_keyans_safe_exec:
3280     \__enumext_keyans_parse_keys:n {#1}
3281     \__enumext_before_list_v:
3282     \__enumext_start_list:nn
3283     { \tl_use:N \l__enumext_label_v_tl }
3284     {
3285         \__enumext_list_arg_two_v:
3286         \__enumext_before_keys_exec_v:
3287     }
3288     \__enumext_after_args_exec_v:
3289 }
3290 {
3291     \__enumext_check_starred_cmd:n { item }
3292     \__enumext_stop_list:
3293     \__enumext_after_list_v:
3294 }

```

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

\\_\_enumext\_keyans\_safe\_exec:

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

```

3295 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3296 {
3297     \bool_if:NF \l__enumext_store_active_bool
3298     {
3299         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3300     }
3301     \int_incr:N \l__enumext_keyans_level_int
3302     \bool_set_true:N \l__enumext_keyans_env_bool
3303     \__enumext_keyans_start_line:
3304     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3305     \bool_set_false:N \l__enumext_store_active_bool
3306     \int_compare:nNtT { \l__enumext_keyans_level_int } > { 1 }
3307     {
3308         \msg_error:nn { enumext } { keyans-nested }
3309     }
3310     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
3311     {
3312         \msg_error:nn { enumext } { keyans-wrong-level }
3313     }
3314 }

```

(End of definition for \\_\_enumext\_keyans\_safe\_exec:.)

\\_\_enumext\_keyans\_parse\_keys:n

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

```

3315 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3316 {
3317     \keys_set:nn { enumext / keyans } {#1}
3318 }

```

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

`\__enumext_before_list_v:` The function `\__enumext_before_list_v:` will add the *vertical spacing above* the environment if the *above* key is active next to the *code* defined by the *before* key if it is active.

```

3319 \cs_new_protected:Nn \__enumext_before_list_v:
3320 {
3321     \__enumext_vspace_above_v:
3322     \__enumext_before_args_exec_v:

```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_v_dim` to be the *width* of the `\__enumext_mini_env*` environment on the *left side*, using this value together with the value of the `\l__enumext_minipage_hsep_v_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_v_dim` will be set, which will be the *width* of `\__enumextt_mini_env*` environment on the *right side*, always having `\linewidth` as the maximum width between them.

```

3323     \dim_compare:nNtT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3324     {
3325         \dim_set:Nn \l__enumext_minipage_left_v_dim
3326         {
3327             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3328         }

```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\mini-right` command will be incremented, then the function `\__enumext_keyans_mini_addvspace:` is called and the `\__enumext_mini_env*` environment on *left side* will be initialized followed by the *vertical spacing* `\l__enumext_minipage_left_skip`. Here we use the plain TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3329         \bool_set_true:N \l__enumext_minipage_active_v_bool
3330         \int_gincr:N \g__enumext_minipage_stat_int
3331         \__enumext_keyans_mini_addvspace:
3332         \nointerlineskip\noindent
3333         \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3334     }

```

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

```

3335     \__enumext_keyans_multicols_start:
3336 }

```

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

`\__enumext_keyans_multicols_start:` The function `\__enumext_keyans_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key.

```

3337 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3338 {
3339     \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3340     {

```

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

```

3341         \dim_compare:nNtT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3342         {
3343             \dim_set:Nn \l__enumext_columns_sep_v_dim
3344             {
3345                 (
3346                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3347                 ) / \l__enumext_columns_v_int
3348                 - \l__enumext_listoffset_v_dim
3349             }
3350         }
3351         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim

```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```

3352         \skip_zero:N \multicolsep
3353         \dim_zero:N \columnseprule

```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `\__enumext_keyans_multi_addvspace:` and apply our “*vertical adjust spacing*”, then start the `multicols` environment.

```

3354         \bool_if:NF \l__enumext_minipage_active_v_bool
3355         {
3356             \__enumext_keyans_multi_addvspace:

```

```

3357     }
3358     \raggedcolumns
3359     \begin{multicols}{\l__enumext_columns_v_int }
3360 }
3361 }

```

(End of definition for \\_\_enumext\_keyans\_multicols\_start:.)

\\_\_enumext\_keyans\_multicols\_stop: The function \\_\_enumext\_keyans\_multicols\_stop: will stop the `multicols` environment. If the boolean variable \l\_\_enumext\_minipage\_active\_v\_bool is false (not nested in `__enumext_mini_env*`) we will apply our vertical “adjust” spacing.

```

3362 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3363 {
3364   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3365   {
3366     \end{multicols}
3367     \bool_if:NF \l__enumext_minipage_active_v_bool
3368     {
3369       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3370     }
3371   }
3372 }

```

(End of definition for \\_\_enumext\_keyans\_multicols\_stop:.)

\\_\_enumext\_after\_list\_v: The function \\_\_enumext\_after\_list\_v: will check the state of the boolean variable \l\_\_enumext\_minipage\_active\_v\_bool, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_env*` environment has not been closed), then close `__enumext_mini_env*` and add the vertical adjustment space \l\_\_enumext\_minipage\_after\_skip, otherwise we will close the `multicols` environment.

```

3373 \cs_new_protected:Nn \__enumext_after_list_v:
3374 {
3375   \bool_if:NTF \l__enumext_minipage_active_v_bool
3376   {
3377     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3378     {
3379       \msg_warning:nn { enumext } { missing-miniright }
3380       \miniright
3381     }
3382     \int_gzero:N \g__enumext_minipage_stat_int
3383     \end{__enumext_mini_env*}
3384     \par\addvspace{ \l__enumext_minipage_after_skip }
3385   }
3386   { \__enumext_keyans_multicols_stop: }

```

Finally we will apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

3387   \bool_set_false:N \l__enumext_keyans_env_bool
3388   \__enumext_after_stop_list_v:
3389   \__enumext_vspace_below_v:
3390 }

```

(End of definition for \\_\_enumext\_after\_list\_v:.)

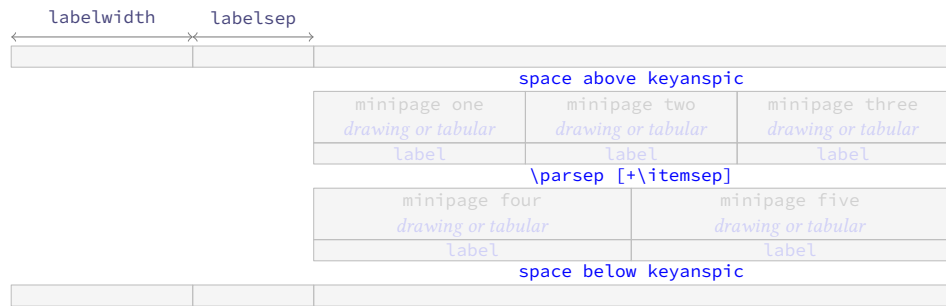
### 11.36 The environment `keyanspic` and `\anspic`

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.

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?](#).

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

### 11.36.1 The command `\anspic`

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

```
3391 \NewDocumentCommand \anspic { s o +m }
3392 {
```

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

```
3393   \bool_if:NF \__enumext_store_active_bool
3394   {
3395     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3396   }
3397   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3398   {
3399     \msg_error:nn { enumext } { keyanspic-wrong-level }
3400   }
3401   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3402   {
3403     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3404   }
```

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

```
3405   \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3406   {
3407     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3408   }
3409 }
```

(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.

```
3410 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3411 {
3412   \stepcounter { enumXvi }
3413   #3 \\\
3414   \bool_if:nT { #1 }
3415   {
3416     \__enumext_keyans_addto_prop:n { #2 }
3417     \__enumext_keyans_store_ref:
3418     \__enumext_keyans_addto_seq:n { #2 }
3419     \int_gincr:N \__enumext_check_starred_cmd_int
3420     \bool_lazy_or:nnT
3421     { \bool_if_p:N \__enumext_show_answer_bool }
3422     { \bool_if_p:N \__enumext_show_position_bool }
3423     {
3424       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3425       \__enumext_keyans_show_left:n { #2 }
3426       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3427     }
3428   }
3429   \tl_use:N \__enumext_label_font_style_v_tl
3430   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3431 }
```

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

### 11.36.2 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.

```

3432 \NewDocumentEnvironment{keyanspic}{ o }
3433 {
3434   \__enumext_keyans_pic_safe_exec:
3435   \__enumext_start_list:nn
3436   { }
3437   {
3438     \__enumext_keyans_pic_arg_two:
3439   }

```

We apply the “adjusted” vertical spacing above the environment

```

3440   \vspace { \__enumext_keyans_pic_above_skip }
3441 }

```

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.

```

3442 {
3443   \tl_if_novalue:nTF { #1 }
3444   {
3445     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3446   }
3447   { \__enumext_keyans_pic_do:n { #1 } }
3448   \__enumext_stop_list:
3449   \__enumext_check_starred_cmd:n { anspic }
3450   \setcounter { enumXvi } { 0 }
3451   \vspace { \__enumext_topsep_v_skip }
3452   %\bool_set_false:N \l__enumext_store_active_bool
3453 }

```

(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.

```

3454 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3455 {
3456   \int_incr:N \l__enumext_keyans_pic_level_int
3457   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3458   {
3459     \msg_error:nn { enumext } { keyanspic-nested }
3460   }
3461   \__enumext_keyans_start_line:
3462 }

```

(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`.

```

3463 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3464 {
3465   \dim_compare:nNnT { #1 } < { 0pt }
3466   { \skip_set:Nn #1 { -#1 } }
3467 }

```

(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.

```

3468 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3469 {

```

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.

```

3470   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3471   \__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.

```

3472 \skip_add:Nn \parsep { \itemsep }
3473 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3474 \dim_zero:N \labelwidth
3475 \dim_zero:N \listparindent
3476 \dim_zero:N \labelsep
3477 \skip_zero:N \partopsep
3478 \skip_zero:N \itemsep

```

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

```

3479 \__enumxt_keyans_pic_skip_abs:N \parsep
3480 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
3481 {
3482   \box_dp:N \strutbox
3483   + \l__enumxt_topsep_v_skip
3484   - \parsep
3485 }
3486 \__enumxt_item_std:w \scan_stop:
3487 }

```

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

```

\__enumxt_keyans_pic_do:n
\__enumxt_keyans_pic_do:e

```

The optional argument is split by comma and is handled directly by the function `\__enumxt_keyans_pic_do:n` and passed to the function `\__enumxt_keyans_pic_row:n`.

```

3488 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
3489 {
3490   \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
3491 }
3492 \cs_generate_variant:Nn \__enumxt_keyans_pic_do:n { e }

```

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

```
\__enumxt_keyans_pic_row:n
```

The function `\__enumxt_keyans_pic_row:n` will set the widths for the `minipage` environments and place the content  $\langle stored \rangle$  by `\anspic*` in the `\l__enumxt_keyans_pic_body_seq` sequence inside them.

```

3493 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
3494 {
3495   \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
3496   \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
3497   \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
3498   \int_step_inline:nnn
3499     { \l__enumxt_keyans_pic_above_int + 1 }
3500     { \l__enumxt_keyans_pic_below_int }
3501     {
3502       \__enumxt_minipage:w [ b ]{ \l__enumxt_keyans_pic_width_dim }
3503       \centering
3504       \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
3505       \__enumxt_endminipage:
3506     }
3507   \par
3508 }

```

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

### 11.37 The environment `enumxt*`

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

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



### 11.37.1 Functions for item box width

\\_enumext\_starred\_columns\_set\_vii:

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3509 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3510 {
3511   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3512   {
3513     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3514     {
3515       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3516       / \l__enumext_columns_vii_int
3517     }
3518   }
3519   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3520   \dim_set:Nn \l__enumext_item_width_vii_dim
3521   {
3522     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3523     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3524     - \l__enumext_labelsep_vii_dim
3525   }
3526   \dim_zero_new:N \itemwidth
3527 }

```

(End of definition for \\_enumext\_starred\_columns\_set\_vii:.)

\\_enumext\_starred\_joined\_item\_vii:n

The function `\_enumext_starred_joined_item_vii:n` will set the *width* of the box in which the content passed to `\item(<number>)` will be stored together with the value of `\itemwidth`.

```

3528 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3529 {
3530   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3531   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3532   {
3533     \msg_warning:nnee { enumext } { item-joined }
3534     { \int_use:N \l__enumext_joined_item_vii_int }
3535     { \int_use:N \l__enumext_columns_vii_int }
3536     \int_set:Nn \l__enumext_joined_item_vii_int
3537     {
3538       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3539     }
3540   }
3541   \int_compare:nNnT
3542   { \l__enumext_joined_item_vii_int }
3543   >
3544   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3545   {
3546     \msg_warning:nnee { enumext } { item-joined-columns }
3547     { \int_use:N \l__enumext_joined_item_vii_int }
3548     {
3549       \int_eval:n
3550       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3551     }
3552     \int_set:Nn \l__enumext_joined_item_vii_int
3553     {
3554       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3555     }
3556   }

```

Only need if #1 >> 1 (default are set before).

```

3557   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3558   {
3559     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3560     \int_decr:N \l__enumext_joined_item_aux_vii_int
3561     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3562     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3563     \dim_set:Nn \l__enumext_joined_width_vii_dim
3564     {
3565       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3566       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3567         + \l__enumext_columns_sep_vii_dim
3568         ) * \l__enumext_joined_item_aux_vii_int

```

```

3569     }
3570     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3571   }
3572   {
3573     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3574     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3575   }
3576 }

```

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

`\__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`.

```

3577 \cs_new_protected:Nn \__enumext_start_mini_vii:
3578 {
3579   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3580   {
3581     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3582     {
3583       \linewidth
3584       - \l__enumext_minipage_right_vii_dim
3585       - \l__enumext_minipage_hsep_vii_dim
3586     }
3587     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3588     \dim_gset_eq:NN
3589       \g__enumext_minipage_right_vii_dim
3590       \l__enumext_minipage_right_vii_dim
3591     \__enumext_mini_addvspace_vii:
3592     \nointerlineskip\noindent
3593     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3594   }
3595 }

```

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

`\__enumext_stop_mini_vii:` 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_star_env:n` to execute the `\__enumext_mini_env*` on the “*right side*”.

```

3596 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3597 {
3598   \bool_if:NT \l__enumext_minipage_active_vii_bool
3599   {
3600     \end{__enumext_mini_env*}
3601     \hfill
3602     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3603   }
3604 }

```

Finally we execute 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*”.

```

3605 \__enumext_after_env:n {enumext*}
3606 {
3607   \bool_if:NT \g__enumext_minipage_active_vii_bool
3608   {
3609     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3610     \par\addvspace { \g__enumext_minipage_right_skip }
3611     \bool_if:NF \g__enumext_minipage_center_vii_bool
3612     {
3613       \centering
3614     }
3615     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3616     \end{__enumext_mini_env*}
3617     \par\addvspace{ \g__enumext_minipage_after_skip }
3618   }
3619   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3620   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3621   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3622   \dim_gzero:N \g__enumext_minipage_right_vii_dim

```

```

3623     \bool_gset_false:N \g__enumext_starred_bool
3624 }

```

(End of definition for \\_\_enumext\_stop\_mini\_vii:.)

**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.

```

3625 \NewDocumentEnvironment{enumext*}{o}
3626 {
3627     \__enumext_safe_exec_vii:
3628     \__enumext_parse_keys_vii:n {#1}
3629     \__enumext_before_list_vii:
3630     \__enumext_start_store_level_vii:
3631     \__enumext_start_list:nn { }
3632     {
3633         \__enumext_list_arg_two_vii:
3634         \__enumext_before_keys_exec_vii:
3635     }
3636     \__enumext_starred_columns_set_vii:
3637     \item[] \scan_stop:
3638     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3639     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3640 }
3641 {
3642     \__enumext_stop_item_tmp_vii:
3643     \__enumext_remove_extra_parsep_vii:
3644     \__enumext_stop_list:
3645     \__enumext_stop_store_level_vii:
3646     \__enumext_after_list_vii:
3647 }

```

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

\\_\_enumext\_safe\_exec\_vii: First check the maximum nesting level for the **enumext\*** environment then set the vars \l\_\_enumext\_starred\_bool and \g\_\_enumext\_starred\_bool.

```

3648 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3649 {
3650     \__enumext_internal_mini_page:
3651     \__enumext_is_not_nested:
3652     \int_incr:N \l__enumext_level_h_int
3653     \int_compare:nNt { \l__enumext_level_h_int } > { 1 }
3654     {
3655         \msg_error:nn { enumext } { nested }
3656     }
3657     \bool_set_true:N \l__enumext_starred_bool
3658     \__enumext_is_on_first_level:
3659 }

```

(End of definition for \\_\_enumext\_safe\_exec\_vii:.)

\\_\_enumext\_parse\_keys\_vii:n Parse [*key = val*] for **enumext\***. If the variable \l\_\_enumext\_store\_active\_bool is true it will call the functions \\_\_enumext\_parse\_series:n and \\_\_enumext\_store\_active\_keys\_vii:n and reprocess the *keys* to pass them to the storage *sequence*.

```

3660 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3661 {
3662     \tl_if_novalue:nF {#1}
3663     {
3664         \str_clear:N \l__enumext_series_str
3665         \keys_set:nn { enumext / enumext* } {#1}
3666         \__enumext_parse_series:n {#1}
3667         \__enumext_store_active_keys_vii:n {#1}
3668     }
3669 }

```

(End of definition for \\_\_enumext\_parse\_keys\_vii:n.)

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

```

3670 \cs_new_protected:Nn \__enumext_before_list_vii:
3671 {
3672   \__enumext_vspace_above_vii:
3673   \__enumext_check_ans_active:
3674   \__enumext_before_args_exec_vii:
3675   \__enumext_start_mini_vii:
3676 }

```

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

`\__enumext_after_list_vii:` The function `\__enumext_after_list:` first call the function `\__enumext_stop_mini_vii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present. Finally set false the vars `\g__enumext_starred_bool` and `\l__enumext_starred_bool`, save the *current value* of the counter in `\g__enumext_resume_vii_int` for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

3677 \cs_new_protected:Nn \__enumext_after_list_vii:
3678 {
3679   \__enumext_stop_mini_vii:
3680   \__enumext_after_stop_list_vii:
3681   \__enumext_check_ans_key_hook:
3682   \__enumext_vspace_below_vii:
3683   \bool_set_false:N \l__enumext_starred_bool
3684   \__enumext_resume_save_counter:
3685 }

```

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

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

```

3686 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3687 {
3688   \bool_if:NT \l__enumext_store_active_bool
3689   {
3690     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3691     {
3692       \__enumext_store_level_open_vii:
3693     }
3694   }
3695 }
3696 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3697 {
3698   \bool_if:NT \l__enumext_store_active_bool
3699   {
3700     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3701     {
3702       \__enumext_store_level_close_vii:
3703     }
3704   }
3705 }

```

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

### 11.37.2 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`.

```

3706 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3707 {
3708   \__enumext_stop_item_tmp_vii:
3709   \int_incr:N \l__enumext_item_column_pos_vii_int
3710   \int_gincr:N \g__enumext_item_count_all_vii_int
3711   \__enumext_item_peek_args_vii:
3712 }

```

(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).

```
3713 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3714 {
3715     \peek_meaning:NTF (
3716         { \__enumext_joined_item_vii:w }
3717         { \__enumext_joined_item_vii:w (1) }
3718     }
```

(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`.

```
3719 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3720 {
3721     \__enumext_starred_joined_item_vii:n {#1}
3722     \peek_meaning_remove:NTF *
3723     { \__enumext_starred_item_vii:w }
3724     { \__enumext_standar_item_vii:w }
3725 }
```

(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 ]`.

```
3726 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3727 {
3728     \bool_set_false:N \l__enumext_item_starred_vii_bool
3729     \peek_meaning:NTF [
3730     {
3731         \bool_set_eq:NN
3732         \l__enumext_wrap_label_vii_bool
3733         \l__enumext_wrap_label_opt_vii_bool
3734         \__enumext_start_item_vii:w
3735     }
3736     {
3737         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3738         \legacy_if_set_true:n { @noitemarg }
3739         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3740     }
3741 }
```

(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`

```
3742 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3743 {
3744     \bool_set_true:N \l__enumext_item_starred_vii_bool
3745     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3746     \peek_meaning:NTF [
3747         { \__enumext_starred_item_vii_aux_i:w }
3748         { \__enumext_starred_item_vii_aux_ii:w }
3749     }
3750 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3751 {
3752     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3753     \__enumext_starred_item_vii_aux_ii:w
```

```

3754 }
3755 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3756 {
3757   \peek_meaning:NTF [
3758     { \__enumext_starred_item_vii_aux_iii:w }
3759     {
3760       \dim_set_eq:NN
3761         \l__enumext_item_symbol_sep_vii_dim
3762         \l__enumext_labelsep_vii_dim
3763       \legacy_if_set_true:n { @noitemarg }
3764       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3765     }
3766   }
3767 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3768 {
3769   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3770   \legacy_if_set_true:n { @noitemarg }
3771   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3772 }

```

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

### 11.37.3 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.

```

3773 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3774 {
3775   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3776   \legacy_if:nT { @noitemarg }
3777   {
3778     \legacy_if_set_false:n { @noitemarg }
3779     \legacy_if:nT { @nmbrlist }
3780     {
3781       \bool_if:NT \l__enumext_hyperref_bool
3782       {
3783         \legacy_if_set_true:n { @hyper@item }
3784       }
3785       \refstepcounter{enumXvii}
3786       \bool_if:NT \l__enumext_check_answers_bool
3787       {
3788         \int_gincr:N \g__enumext_item_number_int
3789       }
3790     }
3791   }

```

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.

```

3792 \group_begin:
3793   \lrbox{ \l__enumext_item_text_vii_box }
3794   \bool_if:NF \l__enumext_footnotes_key_bool
3795   {
3796     \__enumext_renew_footnote:
3797   }
3798   \bool_if:NT \l__enumext_item_starred_vii_bool
3799   {
3800     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3801     {
3802       \tl_gset_eq:NN
3803         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3804     }

```

```

3805         \mode_leave_vertical:
3806         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3807         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3808         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3809         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3810     }
3811     \group_begin:
3812     \tl_use:N \l__enumext_label_font_style_vii_tl
3813     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3814     {
3815         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3816         { \__enumext_wrapper_label_vii:n {#1} }
3817     }
3818     {
3819         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3820     }
3821     \group_end:
3822     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3823     \tl_use:N \l__enumext_after_list_args_vii_tl
3824     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3825     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3826     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3827     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3828 }

```

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

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

```

3829 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3830 {
3831     \__enumext_endminipage:
3832     \endlrbox
3833     \group_end:
3834     \box_set_wd:Nn \l__enumext_item_text_vii_box
3835     {
3836         \l__enumext_joined_width_vii_dim
3837         + \l__enumext_labelwidth_vii_dim
3838         + \l__enumext_labelsep_vii_dim
3839     }
3840     \int_set:Nn \hbadness { 10000 }
3841     \box_use:N \l__enumext_item_text_vii_box
3842     \bool_if:NF \l__enumext_footnotes_key_bool
3843     {
3844         \__enumext_print_footnote:
3845     }
3846     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3847     {
3848         \par\noindent
3849         \int_zero:N \l__enumext_item_column_pos_vii_int
3850     }
3851     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3852 }

```

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

`\__enumext_remove_extra_parsep_vii:` Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

3853 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3854 {
3855     \int_compare:nNnT
3856     {
3857         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3858     }
3859     =
3860     { \c_zero_int }
3861     {
3862         \par
3863         \vspace{ -\l__enumext_itemsep_vii_skip }
3864         \int_zero:N \g__enumext_item_count_all_vii_int

```



```

3865     }
3866 }

```

(End of definition for `\__enumext_remove_extra_parsep_viii:`)

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`.

```

3867 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

## 11.38 The environment keyans\*

### 11.38.1 Functions for item box width

```
\__enumext_starred_columns_set_viii:
```

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3868 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3869 {
3870   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3871   {
3872     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3873     {
3874       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3875       / \l__enumext_columns_viii_int
3876     }
3877   }
3878   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3879   \dim_set:Nn \l__enumext_item_width_viii_dim
3880   {
3881     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3882     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3883     - \l__enumext_labelsep_viii_dim
3884   }
3885   \dim_zero_new:N \itemwidth
3886 }

```

(End of definition for `\__enumext_starred_columns_set_viii:`)

```
\__enumext_starred_joined_item_viii:n
```

The function `\__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the content passed to `\item(<number>)` will be stored together with the value of `\itemwidth`.

```

3887 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3888 {
3889   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3890   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3891   {
3892     \msg_warning:nnee { enumext } { item-joined }
3893     { \int_use:N \l__enumext_joined_item_viii_int }
3894     { \int_use:N \l__enumext_columns_viii_int }
3895     \int_set:Nn \l__enumext_joined_item_viii_int
3896     {
3897       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3898     }
3899   }
3900   \int_compare:nNnT
3901   { \l__enumext_joined_item_viii_int }
3902   >
3903   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3904   {
3905     \msg_warning:nnee { enumext } { item-joined-columns }
3906     { \int_use:N \l__enumext_joined_item_viii_int }
3907     {
3908       \int_eval:n
3909       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3910     }
3911     \int_set:Nn \l__enumext_joined_item_viii_int
3912     {
3913       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3914     }
3915   }
3916   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3917   {
3918     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3919     \int_decr:N \l__enumext_joined_item_aux_viii_int

```

```

3920 \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3921 \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3922 \dim_set:Nn \l__enumext_joined_width_viii_dim
3923 {
3924   \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3925   + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3926       + \l__enumext_columns_sep_viii_dim
3927       )*\l__enumext_joined_item_aux_viii_int
3928 }
3929 \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3930 }
3931 {
3932   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3933   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3934 }
3935 }

```

(End of definition for `\__enumext_starred_joined_item_viii:n`)

`\__enumext_start_mini_viii:` The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.  
`\__enumext_stop_mini_viii:`

```

3936 \cs_new_protected:Nn \__enumext_start_mini_viii:
3937 {
3938   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3939   {
3940     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3941     {
3942       \linewidth
3943       - \l__enumext_minipage_right_viii_dim
3944       - \l__enumext_minipage_hsep_viii_dim
3945     }
3946     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3947     \dim_gset_eq:NN
3948       \g__enumext_minipage_right_viii_dim
3949       \l__enumext_minipage_right_viii_dim
3950     \__enumext_mini_addvspace_viii:
3951     \nointerlineskip\noindent
3952     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3953   }
3954 }
3955 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3956 {
3957   \bool_if:NT \l__enumext_minipage_active_viii_bool
3958   {
3959     \end{__enumext_mini_env*}
3960     \hfill
3961     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3962   }
3963 }
3964 \__enumext_after_env:nn {keyans*}
3965 {
3966   \bool_if:NT \g__enumext_minipage_active_viii_bool
3967   {
3968     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3969     \par\addvspace { \g__enumext_minipage_right_skip }
3970     \bool_if:NF \g__enumext_minipage_center_viii_bool
3971     {
3972       \centering
3973     }
3974     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3975     \end{__enumext_mini_env*}
3976     \par\addvspace{ \g__enumext_minipage_after_skip }
3977   }
3978   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3979   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3980   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3981   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3982 }

```

(End of definition for `\__enumext_start_mini_viii:` and `\__enumext_stop_mini_viii:`)

**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.

```

3983 \NewDocumentEnvironment{keyans*}{ o }
3984 {
3985   \__enumext_safe_exec_viii:
3986   \__enumext_parse_keys_viii:n {#1}
3987   \__enumext_before_list_viii:
3988   \__enumext_start_list:nn { }
3989   {
3990     \__enumext_list_arg_two_viii:
3991     \__enumext_before_keys_exec_viii:
3992   }
3993   \__enumext_starred_columns_set_viii:
3994   \item[] \scan_stop:
3995   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3996   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3997 }
3998 {
3999   \__enumext_stop_item_tmp_viii:
4000   \__enumext_remove_extra_parsep_viii:
4001   \__enumext_check_starred_cmd:n { item }
4002   \__enumext_stop_list:
4003   \__enumext_after_list_viii:
4004 }

```

(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.

```

4005 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4006 {
4007   \int_incr:N \__enumext_keyans_level_h_int
4008   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4009   {
4010     \msg_error:nn { enumext } { nested }
4011   }
4012   \__enumext_keyans_start_line:
4013   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4014   \bool_set_false:N \__enumext_store_active_bool
4015   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4016   {
4017     \msg_error:nn { enumext } { keyans-wrong-level }
4018   }
4019 }

```

(End of definition for `\__enumext_safe_exec_viii:`)

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

```

4020 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4021 {
4022   \tl_if_novalue:nF {#1}
4023   {
4024     \keys_set:nn { enumext / keyans* } {#1}
4025   }
4026 }

```

(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`.

```

4027 \cs_new_protected:Nn \__enumext_before_list_viii:
4028 {
4029   \__enumext_vspace_above_viii:
4030   \__enumext_before_args_exec_viii:
4031   \__enumext_start_mini_viii:
4032 }

```

(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.

```
4033 \cs_new_protected:Nn \__enumext_after_list_viii:
4034 {
4035     \__enumext_stop_mini_viii:
4036     \__enumext_after_stop_list_viii:
4037     \__enumext_vspace_below_viii:
4038 }
```

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

### 11.38.2 The command `\item` in `keyans*`

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

`\__enumext_start_item_tmp_viii:` First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
4039 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4040 {
4041     \__enumext_stop_item_tmp_viii:
4042     \int_incr:N \l__enumext_item_column_pos_viii_int
4043     \int_gincr:N \g__enumext_item_count_all_viii_int
4044     \__enumext_item_peek_args_viii:
4045 }
```

(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).

```
4046 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4047 {
4048     \peek_meaning:NTF (
4049         { \__enumext_joined_item_viii:w }
4050         { \__enumext_joined_item_viii:w (1) }
4051 }
```

(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`.

```
4052 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4053 {
4054     \__enumext_starred_joined_item_viii:n {#1}
4055     \peek_meaning_remove:NTF *
4056         { \__enumext_starred_item_viii:w }
4057         { \__enumext_standar_item_viii:w }
4058 }
```

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

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

```

4059 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4060 {
4061   \bool_set_false:N \l__enumext_item_starred_viii_bool
4062   \peek_meaning:NTF [
4063     {
4064       \bool_set_eq:NN
4065       \l__enumext_wrap_label_viii_bool
4066       \l__enumext_wrap_label_opt_viii_bool
4067       \__enumext_start_item_viii:w
4068     }
4069     {
4070       \bool_set_true:N \l__enumext_wrap_label_viii_bool
4071       \legacy_if_set_true:n { @noitemarg }
4072       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4073     }
4074   }

```

(End of definition for \\_\_enumext\_standar\_item\_viii:w.)

```

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

```

The function \\_\_enumext\_starred\_item\_viii:w together with the specified auxiliary functions **aux\_i:w** and **aux\_ii:w** execute `\item*` and `\item*[\langle content \rangle]`.

```

4075 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4076 {
4077   \bool_set_true:N \l__enumext_item_starred_viii_bool
4078   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4079   \peek_meaning:NTF [
4080     { \__enumext_starred_item_viii_aux_i:w }
4081     { \__enumext_starred_item_viii_aux_ii:w }
4082   }

```

The function \\_\_enumext\_starred\_item\_viii\_aux\_i:w will save the optional argument to `\item*` in `\l__enumext_store_current_opt_arg_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `\l__enumext_store_current_label_tl` if present, then call the function `\__enumext_starred_item_viii_aux_ii:w`.

```

4083 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4084 {
4085   \tl_clear:N \l__enumext_store_current_label_tl
4086   \tl_if_no_value:nF { #1 }
4087   {
4088     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4089     {
4090       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4091       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4092     }
4093     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4094   }
4095   \__enumext_starred_item_viii_aux_ii:w
4096 }
4097 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4098 {
4099   \legacy_if_set_true:n { @noitemarg }
4100   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4101 }

```

(End of definition for \\_\_enumext\_starred\_item\_viii:w, \\_\_enumext\_starred\_item\_viii\_aux\_i:w, and \\_\_enumext\_starred\_item\_viii\_aux\_ii:w.)

```
\__enumext_starred_item_exec:
```

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

```

4102 \cs_new_protected:Nn \__enumext_starred_item_exec:
4103 {
4104   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4105   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4106   \__enumext_keyans_store_ref:
4107   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4108   \__enumext_keyans_addto_seq_link:
4109   \int_gincr:N \g__enumext_check_starred_cmd_int
4110   \bool_if:NT \l__enumext_show_answer_bool
4111   {

```

```

4112     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4113   }
4114   \bool_if:NT \l__enumext_show_position_bool
4115   {
4116     \tl_set:Nx \l__enumext_mark_answer_sym_tl
4117     {
4118       \group_begin:
4119       \exp_not:N \normalfont
4120       \exp_not:N \footnotesize [ \int_eval:n
4121       {
4122         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4123       }
4124     ]
4125     \group_end:
4126   }
4127   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4128 }
4129 }

```

(End of definition for `\__enumext_starred_item_exec:`)

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

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

```

4130 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4131 {
4132   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4133   \legacy_if:nT { @noitemarg }
4134   {
4135     \legacy_if_set_false:n { @noitemarg }
4136     \legacy_if:nT { @nmbrrlist }
4137     {
4138       \bool_if:NT \l__enumext_hyperref_bool
4139       {
4140         \legacy_if_set_true:n { @hyper@item }
4141       }
4142       \refstepcounter{enumXviii}
4143     }
4144   }

```

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

```

4145   \group_begin:
4146   \lrbox{ \l__enumext_item_text_viii_box }
4147   \bool_if:NF \l__enumext_footnotes_key_bool
4148   {
4149     \__enumext_renew_footnote:
4150   }
4151   \bool_if:NT \l__enumext_item_starred_viii_bool
4152   {
4153     \__enumext_starred_item_exec:
4154   }
4155   \group_begin:
4156   \tl_use:N \l__enumext_label_font_style_viii_tl
4157   \bool_if:NTF \l__enumext_wrap_label_viii_bool
4158   {
4159     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4160     { \__enumext_wrapper_label_viii:n {#1} }
4161   }
4162   {
4163     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
4164   }
4165   \group_end:
4166   \skip_horizontal:N \l__enumext_labelsep_viii_dim
4167   \tl_use:N \l__enumext_after_list_args_viii_tl
4168   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4169   \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4170   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4171   \bool_if:NT \l__enumext_item_starred_viii_bool
4172   {
4173     \tl_use:N \l__enumext_fake_item_indent_viii_tl
4174     \__enumext_keyans_show_item_opt:
4175     \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }

```

```

4176         }
4177         {
4178             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4179         }
4180     }

```

(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.

```

4181 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4182 {
4183     \__enumext_endminipage:
4184     \endlrbox
4185     \group_end:
4186     \box_set_wd:Nn \l__enumext_item_text_viii_box
4187     {
4188         \l__enumext_joined_width_viii_dim
4189         + \l__enumext_labelwidth_viii_dim
4190         + \l__enumext_labelsep_viii_dim
4191     }
4192     \int_set:Nn \hbadness { 10000 }
4193     \box_use:N \l__enumext_item_text_viii_box
4194     \bool_if:NF \l__enumext_footnotes_key_bool
4195     {
4196         \__enumext_print_footnote:
4197     }
4198     \int_compare:nNnTF
4199     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4200     {
4201         \par\noindent
4202         \int_zero:N \l__enumext_item_column_pos_viii_int
4203     }
4204     { \hspace{ \l__enumext_columns_sep_viii_dim } }
4205 }

```

(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.

```

4206 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4207 {
4208     \int_compare:nNnT
4209     {
4210         \int_mod:nn
4211         { \g__enumext_item_count_all_viii_int }
4212         { \l__enumext_columns_viii_int }
4213     }
4214     =
4215     { \c_zero_int }
4216     {
4217         \par
4218         \vspace{ -\l__enumext_itemsep_viii_skip }
4219         \int_gzero:N \g__enumext_item_count_all_viii_int
4220     }
4221 }

```

(End of definition for \\_\_enumext\_remove\_extra\_parsep\_viii:.)

### 11.39 The command `\getkeyans`

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

```

4222 \NewDocumentCommand \getkeyans { m }
4223 {
4224     \exp_args:Ne \__enumext_getkeyans_aux:n
4225     { \tl_to_str:e { \text_expand:n {#1} } }
4226 }

```

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



`\__enumext_getkeyans_aux:n` The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the *⟨argument⟩* using “:”. If “:” is omitted it will return an error.

```

4227 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4228 {
4229   \str_if_in:nnTF {#1} { : }
4230   {
4231     \use:e
4232     {
4233       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4234       { {##1} {##2} }
4235     }
4236     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4237   }
4238   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4239 }

```

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

`\__enumext_getkeyans:nn` The internal function `\__enumext_getkeyans:nn` will check for the existence of the *⟨prop list⟩*, if it does not exist it will return an error message, then it will fetch the content specified by the second *⟨argument⟩* from *⟨prop list⟩*.

```

4240 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4241 {
4242   \prop_if_exist:cF { g__enumext_#1_prop }
4243   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4244   \group_begin:
4245   \prop_item:cn { g__enumext_#1_prop }{#2}
4246   \group_end:
4247 }

```

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

## 11.40 The command `\printkeyans`

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

The variable `\__enumext_print_keyans_starred_tl` will have the default *⟨keys⟩* for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `\__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *⟨sequence⟩* and will be set by `\setenumext[⟨print,*⟩]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[⟨print,level⟩]`

```

4248 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4249 \keys_define:nn { enumext / print }
4250 {
4251   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4252                 { \__enumext_filter_save_key:n {#1} }
4253                 \__enumext_print_keyans_starred_tl, % starred cmd
4254   print* .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4255   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4256                 { \__enumext_filter_save_key:n {#1} }
4257                 \__enumext_print_keyans_i_tl,
4258   print-1 .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4259   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4260                 { \__enumext_filter_save_key:n {#1} }
4261                 \__enumext_print_keyans_ii_tl,
4262   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4263   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4264                 { \__enumext_filter_save_key:n {#1} }
4265                 \__enumext_print_keyans_iii_tl,
4266   print-3 .initial:n = { nosep, label=\roman*, first=\small, font=\small },
4267   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4268                 { \__enumext_filter_save_key:n {#1} }
4269                 \__enumext_print_keyans_iv_tl,
4270   print-4 .initial:n = { nosep, label=\Alph*, first=\small, font=\small },
4271   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4272                 { \__enumext_filter_save_key:n {#1} }
4273                 \__enumext_print_keyans_vii_tl, % starred nested
4274   print-* .initial:n = { nosep, label=\arabic*, first=\small, font=\small },
4275 }

```

- 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`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `\__enumext_printkeyans:nnn`.

```

4276 \NewDocumentCommand \printkeyans { s O{} m }
4277 {
4278   \group_begin:
4279     \tl_use:N \__enumext_print_keyans_i_tl
4280     \tl_use:N \__enumext_print_keyans_ii_tl
4281     \tl_use:N \__enumext_print_keyans_iii_tl
4282     \tl_use:N \__enumext_print_keyans_iv_tl
4283     \tl_use:N \__enumext_print_keyans_vii_tl
4284     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4285   \group_end:
4286 }

```

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

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

```

4287 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4288 {
4289   \seq_if_exist:cTF { g__enumext_#3_seq }
4290   {
4291     \seq_if_empty:cF { g__enumext_#3_seq }
4292     {
4293       %%\seq_show:c { g__enumext_#3_seq }

```

If the starred if it is present we will check that the environment `enumext*` is not saved in the  $\langle sequence \rangle$ , then execute the variable `\__enumext_print_keyans_starred_tl` that contains the default  $\langle keys \rangle$  for the environment `enumext*`, it will open the environment `enumext*` passing the optional argument to the first level and then will map the  $\langle sequence \rangle$

```

4294     \bool_if:nTF {#1}
4295     {
4296       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4297       {
4298         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4299       }
4300       {
4301         \tl_use:N \__enumext_print_keyans_starred_tl
4302         \begin{enumext*}[#2]
4303           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4304           \end{enumext*}
4305       }
4306     }

```

Otherwise it will open the environment `enumext` passing the optional argument to the first level and then map the  $\langle sequence \rangle$ .

```

4307     {
4308       \begin{enumext}[#2]
4309       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4310       \end{enumext}
4311     }
4312   }
4313   {
4314     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4315   }
4316 }
4317

```

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

### 11.41 The command \setenumext

First we define a “meta families” of  $\langle keys \rangle$  to access from \setenumext.

```

4318 \keys_define:nn { enumext / meta-families }
4319 {
4320   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4321   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4322   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4323   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4324   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4325   enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4326   keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4327   print*    .code:n = { \keys_set:nn { enumext / print   } { print* = {#1} } } ,
4328   print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
4329   print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
4330   print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
4331   print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
4332   print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
4333   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4334 }

```

We store them in the constant sequence \c\_\_enumext\_all\_families\_seq separated by commas.

```

4335 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4336 {
4337   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4338   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4339 }

```

\setenumext Now we define the user command \setenumext.

```

4340 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4341 {
4342   \tl_if_novalue:nTF {#1}
4343   {
4344     \seq_map_inline:Nn \c__enumext_all_families_seq
4345   }
4346   {
4347     \seq_clear:N \l__enumext_setkey_tmpa_seq
4348     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4349     \int_set:Nn \l__enumext_setkey_tmpa_int
4350     {
4351       \seq_count:N \l__enumext_setkey_tmpb_seq
4352     }
4353     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4354     {
4355       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4356       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4357       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4358       {
4359         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4360       }
4361     }
4362     {
4363       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4364     }
4365     \seq_if_empty:NnTF \l__enumext_setkey_tmpa_seq
4366     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4367     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4368   }
4369   {
4370     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4371   }
4372 }

```

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

\l\_\_enumext\_set\_parse:n  
\l\_\_enumext\_set\_error:n

Internal functions used by the \setenumext command.

```

4373 \cs_new_protected:Npn \l__enumext_set_parse:n #1
4374 {
4375   \tl_set:Nc \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4376   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4377   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }

```

```

4378 \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4379 {
4380   \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4381   { \tl_trim_spaces:n {#1} }
4382 }
4383 { \__enumext_set_error:nn {#1} { } }
4384 }
4385 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4386 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

## 11.42 Messages

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

```

4387 \msg_new:nnn { enumext } { package-load }
4388 {
4389   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4390 }
4391 \msg_new:nnn { enumext } { package-not-load }
4392 {
4393   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4394 }
4395 \msg_new:nnn { enumext } { package-load-foot }
4396 {
4397   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4398 }

```

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

```

4399 \msg_new:nnn { enumext } { counters }
4400 {
4401   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4402   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4403 }

```

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

```

4404 \msg_new:nnn { enumext } { unknown-choice }
4405 {
4406   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4407 }
4408 % \begin{macrocode}
4409 %
4410 % Message used by reserved \myenv*{anskey*} environment by \mypkg*{enumext} package.
4411 % \begin{macrocode}
4412 \msg_new:nnnn { enumext } { anskey-env-error }
4413 {
4414   The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
4415   'enumext' ~ package, ~ It~ is~ already~ defined.
4416 }
4417 {
4418   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4419   for ~ the ~ 'save-ans' ~ key.\\
4420 }

```

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

```

4421 \msg_new:nnn { enumext } { store-prop }
4422 {
4423   * ~ Package ~ enumext: ~ Creating ~
4424   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4425 }
4426 \msg_new:nnn { enumext } { store-seq }
4427 {
4428   * ~ Package ~ enumext: ~ Creating ~
4429   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4430 }
4431 \msg_new:nnn { enumext } { store-int }
4432 {
4433   * ~ Package ~ enumext: ~ Creating ~
4434   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4435 }
4436 \msg_new:nnn { enumext } { prop-seq-int-hook }
4437 {
4438   * ~ Package ~ enumext: ~ Elements ~ in ~

```

```

4439     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4440     * ~ Package ~ enumext: ~ Elements ~ in ~
4441     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4442     * ~ Package ~ enumext: ~ Value ~ off ~
4443     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4444 }
4445 \msg_new:nnn { enumext } { item-answer-hook }
4446 {
4447     * ~ Package ~ enumext: ~ Value ~ off ~
4448     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4449     * ~ Package ~ enumext: ~ Value ~ off ~
4450     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4451     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4452 }

```

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

```

4453 \msg_new:nnn { enumext } { invalid-key }
4454 {
4455     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4456 }
4457 \msg_new:nnn { enumext } { unknown-key-family }
4458 {
4459     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4460 }

```

Messages used in length calculation.

```

4461 \msg_new:nnn { enumext } { width-negative }
4462 {
4463     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4464     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4465 }
4466 \msg_new:nnn { enumext } { width-zero }
4467 {
4468     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4469     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4470 }

```

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

```

4471 \msg_new:nnn { enumext } { list-lengths }
4472 {
4473     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4474     \__enumext_show_length:nnn { dim } { labelsep } { #1}
4475     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
4476     \__enumext_show_length:nnn { dim } { itemindent } { #1}
4477     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
4478     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
4479     \__enumext_show_length:nnn { dim } { listparindent } { #1}
4480     \__enumext_show_length:nnn { skip } { topsep } { #1}
4481     \__enumext_show_length:nnn { skip } { parsep } { #1}
4482     \__enumext_show_length:nnn { skip } { partopsep } { #1}
4483     \__enumext_show_length:nnn { skip } { itemsep } { #1}
4484     ****
4485 }

```

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

```

4486 \msg_new:nnn { enumext } { list-lengths-not-nested }
4487 {
4488     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4489     \__enumext_show_length:nnn { dim } { labelsep } { #1}
4490     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
4491     \__enumext_show_length:nnn { dim } { itemindent } { #1}
4492     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
4493     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
4494     \__enumext_show_length:nnn { dim } { listparindent } { #1}
4495     \__enumext_show_length:nnn { skip } { topsep } { #1}
4496     \__enumext_show_length:nnn { skip } { parsep } { #1}
4497     \__enumext_show_length:nnn { skip } { partopsep } { #1}
4498     \__enumext_show_length:nnn { skip } { itemsep } { #1}
4499     ****
4500 }

```

Messages used by `ref` key.

```

4501 \msg_new:nnn { enumext } { key-ref-empty }

```

```

4502 {
4503     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4504 }

```

Messages used by `save-ans` key.

```

4505 \msg_new:nnn { enumext } { save-ans-empty }
4506 {
4507     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4508 }
4509 \msg_new:nnn { enumext } { save-ans-log }
4510 {
4511     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4512 }
4513 \msg_new:nnn { enumext } { save-ans-log-hook }
4514 {
4515     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4516 }
4517 \msg_new:nnn { enumext } { save-ans-hook }
4518 {
4519     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4520 }

```

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

```

4521 \msg_new:nnn { enumext } { need-save-ans }
4522 {
4523     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4524 }
4525 \msg_new:nnn { enumext } { items-same-answer }
4526 {
4527     *****\
4528     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4529     for ~ \c_left_brace_str #2 \c_right_brace_str\
4530     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4531     'OK', ~ all ~ items ~ with ~ answer.\
4532     *****
4533 }
4534 \msg_new:nnn { enumext } { item-greater-answer }
4535 {
4536     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4537     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4538     Items ~ > ~ Answers.
4539 }
4540 \msg_new:nnn { enumext } { item-less-answer }
4541 {
4542     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4543     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4544     Items ~ < ~ Answers.
4545 }

```

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

```

4546 \msg_new:nnn { enumext } { missing-starred }
4547 {
4548     Missing ~ '\c_backslash_str #1*' ~ #2.
4549 }
4550 \msg_new:nnn { enumext } { many-starred }
4551 {
4552     Many ~ '\c_backslash_str #1*' ~ #2.
4553 }

```

Messages used by `\printkeyans*` command.

```

4554 \msg_new:nnn { enumext } { print-starred }
4555 {
4556     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4557     #2 ~ environment ~ \msg_line_context:.
4558 }

```

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

```

4559 \msg_new:nnn { enumext } { list-too-deep }
4560 {
4561     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
4562     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4563 }

```

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

```

4564 \msg_new:nnn { enumext } { anskey-empty-arg }
4565 {
4566   Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4567 }
4568 \msg_new:nnn { enumext } { anskey-wrong-place }
4569 {
4570   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4571   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4572 }
4573 \msg_new:nnn { enumext } { anskey-nested }
4574 {
4575   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4576 }
4577 \msg_new:nnn { enumext } { anskey-env-wrong }
4578 {
4579   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4580 }
4581 \msg_new:nnn { enumext } { anspic-wrong-place }
4582 {
4583   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4584   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4585 }
4586 \msg_new:nnn { enumext } { command-wrong-place }
4587 {
4588   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4589   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4590 }

```

Messages used by `keyans` and `keyanspic` environment.

```

4591 \msg_new:nnn { enumext } { keyans-nested }
4592 {
4593   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4594 }
4595 \msg_new:nnn { enumext } { keyans-wrong-level }
4596 {
4597   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4598   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4599 }
4600 \msg_new:nnn { enumext } { wrong-place }
4601 {
4602   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4603   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4604 }
4605 \msg_new:nnn { enumext } { keyanspic-nested }
4606 {
4607   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4608 }
4609 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4610 {
4611   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4612   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4613 }

```

Messages used by `\getkeyans` command.

```

4614 \msg_new:nnn { enumext } { undefined-storage-anskey }
4615 {
4616   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4617 }

```

Messages used by `\miniright` command.

```

4618 \msg_new:nnn { enumext } { missing-miniright }
4619 {
4620   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4621   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4622 }
4623 \msg_new:nnn { enumext } { wrong-miniright-place }
4624 {
4625   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4626   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4627 }
4628 \msg_new:nnn { enumext } { wrong-miniright-use }

```



```

4629 {
4630     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4631     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4632 }

```

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

```

4633 \msg_new:nnn { enumext } { nested }
4634 {
4635     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4636 }
4637 \msg_new:nnn { enumext } { item-joined }
4638 {
4639     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4640 }
4641 \msg_new:nnn { enumext } { item-joined-columns }
4642 {
4643     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4644 }

```

## 11.43 Finish package

Finish package implementation.

```

4645 \file_input_stop:
4646 </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>	208
<code>\+</code>	200
<code>\-</code>	200
<code>\\</code>	216, 2533, 3413, 4401, 4414, 4419, 4439, 4441, 4448, 4450, 4463, 4468, 4473, 4488, 4527, 4529, 4531, 4536, 4537, 4542, 4543, 4561, 4570, 4583, 4588, 4597, 4602, 4611, 4620, 4625, 4630
A	
<code>above</code>	1383
<code>above*</code>	1383
<code>\addvspace</code>	1037, 1065, 1181, 1260, 1323, 1329, 1357, 1374, 3252, 3267, 3369, 3384, 3610, 3617, 3969, 3976
<code>after</code>	876
<code>align</code>	487
<code>\Alph</code>	36, 40, 41
<code>\Alph</code>	439, 554, 599, 667, 4270
<code>\alph</code>	36, 40, 41
<code>\alph</code>	440, 552, 4262
<code>\anskey</code>	12, 72, 2217, 2611, 2619
<code>anskey*</code>	13, 2479
<code>\anspic</code>	15, 96, 97, 3391
<code>\anspic*</code>	66
<code>\arabic</code>	30, 36
<code>\arabic</code>	438, 551, 598, 4254, 4258, 4274
B	
<code>\baselineskip</code>	48
<code>\baselineskip</code>	2162, 2170
<code>before</code>	876
<code>before*</code>	876
<code>below</code>	1383
<code>below*</code>	1383
bool commands:	
<code>\bool_gset_false:N</code>	314, 315, 316, 2629, 2631, 3619, 3623, 3978
<code>\bool_gset_true:N</code>	228, 237, 979, 1875, 1881, 3602, 3620, 3961, 3979
<code>\bool_if:NTF</code>	379, 391, 408, 1405, 1419, 1432, 1443, 1454, 1465, 1476, 1487, 1540, 1557, 1562, 1570, 1597, 1635, 1640, 1647, 1651, 1673, 1678, 1686, 1693, 1724, 1732, 1825, 1966, 2035, 2045, 2124, 2148, 2155, 2183, 2221, 2231, 2263, 2289, 2406, 2417, 2421, 2578, 2658, 2673, 2748, 2759, 2763, 2876, 2906, 2980, 2996, 3058, 3068, 3098, 3103, 3186, 3236, 3250, 3258, 3297, 3354, 3367, 3375, 3393, 3598, 3607, 3611, 3688, 3698, 3781, 3786, 3794, 3798, 3813, 3842, 3957, 3966, 3970, 4110, 4114, 4138, 4147, 4151, 4157, 4171, 4194
<code>\bool_if:nTF</code>	1358, 1375, 2917, 2952, 3016, 3414, 4294
<code>\bool_if_p:N</code>	246, 260, 1704, 1705, 1713, 1714, 1838, 1860, 1872, 1873, 1878, 1879, 2274, 2315, 2316, 2340, 2349, 2350, 2362, 2378, 2564, 2735, 2736, 2773, 2774, 3159, 3172, 3174, 3421, 3422
<code>\bool_lazy_all:nTF</code>	244, 258, 1836, 1858, 2338, 2347, 2360, 2376, 3157, 3170
<code>\bool_lazy_and:nnTF</code>	224, 233, 1703, 1712, 1871, 1877, 2273, 2280, 2314, 2563, 2569, 2734
<code>\bool_lazy_or:nnTF</code>	1765, 1772, 2772, 3420
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 62, 71, 92, 97, 98, 103, 104, 107, 128, 129, 137, 138, 144, 145, 159, 170, 172
<code>\bool_not_p:n</code>	225, 234, 2275, 2281, 2365, 2380, 2565, 2570, 3160, 3161, 3173
<code>\bool_set_eq:NN</code>	2884, 2932, 3731, 4064
<code>\bool_set_false:N</code>	388, 1810, 1811, 2516, 3273, 3305, 3387, 3452, 3470, 3683, 3728, 4014, 4061
<code>\bool_set_true:N</code>	251, 265, 370, 374, 480, 804, 1389, 1394, 1660, 1782, 1783, 2067, 2075, 2517, 2880, 2910, 2928, 2940, 3135, 3166, 3179, 3205, 3302, 3329, 3587, 3657, 3737, 3744, 3745, 3946, 4070, 4077, 4078
box commands:	
<code>\box_dp:N</code>	1077, 1081, 1085, 1096, 1100, 1111, 1120, 1126, 1136, 1149, 1155, 1161, 1192, 1193, 1194, 1197, 1207, 1211, 1220, 1227, 1232, 1240, 1269, 1270, 1273, 1280, 1293, 1301, 1307, 1315, 3482
<code>\box_new:N</code>	68, 165
<code>\box_set_wd:Nn</code>	3834, 4186
<code>\box_use:N</code>	3841, 4193
<code>\box_wd:N</code>	446
C	
<code>\c</code>	208, 209, 704, 706, 718, 720
<code>\catcode</code>	2533
<code>\cB</code>	209
<code>\cE</code>	209
<code>\centering</code>	1360, 1377, 3503, 3613, 3972
<code>check-ans</code>	1802
Document class:	
<code>article</code>	42
clist commands:	
<code>\clist_const:Nn</code>	177
<code>\clist_map_function:nN</code>	3490
<code>\clist_map_inline:Nn</code>	486, 746, 809, 875, 890, 971, 1399
<code>\clist_map_inline:nn</code>	47, 58, 76, 82, 94, 106, 131, 153, 176, 514, 534, 814, 985, 1505, 1749, 1816, 2014, 2032, 2064, 2335, 2667, 2834, 3045, 3048, 3075, 3085, 3088, 3108, 4376
<code>\columnbreak</code>	73
<code>\columnbreak</code>	2277
<code>columns</code>	955
<code>columns-sep</code>	955
<code>\columnsep</code>	92, 95
<code>\columnsep</code>	3230, 3351
<code>\columnseprule</code>	92, 95
<code>\columnseprule</code>	3234, 3353
Commands provide by enumext:	
<code>\anskey</code>	28, 62, 63, 68, 70, 72, 74–76, 80, 82, 91, 103, 113, 115, 120
<code>\anspic*</code>	28, 66, 70, 80, 81, 97–99, 113, 115
<code>\anspic</code>	70, 96–99, 120
<code>\getkeyans</code>	70, 113, 120
<code>\item*</code>	28, 66, 70, 80, 81, 85, 86, 104, 111, 113, 115
<code>\itemwidth</code>	100, 107
<code>\item</code>	84, 86, 100, 103–105, 107, 110
<code>\miniright</code>	27, 46, 53, 54, 92, 93, 95, 96, 120
<code>\printkeyans*</code>	114
<code>\printkeyans</code>	28, 70, 114, 115

`\setenumext` . . . . . 28, 115, 116, 118

Counters defined by `enumext`:

`enumXiii` . . . . . 26, 35

`enumXii` . . . . . 26, 35

`enumXiv` . . . . . 26, 35

`enumXi` . . . . . 26, 35

`enumXviii` . . . . . 26, 35

`enumXvii` . . . . . 26, 35, 105

`enumXvi` . . . . . 26, 35

`enumXv` . . . . . 26, 35

cs commands:

`\cs_generate_variant:Nn` 448, 464, 710, 726, 2116, 2121, 2201, 2484, 3035, 3492, 4248

`\cs_if_exist:NTF` . . . . . 418

`\cs_if_free:NTF` . . . . . 2448, 2458

`\cs_new:Nn` . . . . . 194

`\cs_new:Npn` . 212, 1506, 1515, 1524, 2079, 2088, 2096

`\cs_new_eq:NN` 341, 342, 343, 347, 348, 393, 394, 397, 398

`\cs_new_protected:Nn` . 204, 218, 242, 273, 300, 306, 312, 318, 324, 332, 350, 365, 575, 638, 690, 891, 895, 899, 903, 907, 911, 915, 919, 923, 927, 931, 935, 939, 943, 947, 951, 986, 998, 1022, 1039, 1050, 1067, 1142, 1166, 1183, 1245, 1262, 1284, 1319, 1325, 1400, 1414, 1428, 1439, 1450, 1461, 1472, 1483, 1568, 1671, 1684, 1701, 1722, 1750, 1755, 1780, 1821, 1831, 1869, 1884, 1891, 1900, 1905, 1910, 1915, 1924, 1929, 1934, 1956, 2122, 2146, 2153, 2181, 2188, 2229, 2326, 2437, 2485, 2506, 2537, 2561, 2603, 2627, 2656, 2671, 2699, 2732, 2768, 2780, 2788, 2839, 2843, 2862, 2913, 2948, 2964, 2974, 2990, 3128, 3155, 3184, 3191, 3214, 3244, 3256, 3295, 3319, 3337, 3362, 3373, 3410, 3454, 3468, 3488, 3493, 3509, 3577, 3596, 3648, 3670, 3677, 3686, 3696, 3713, 3853, 3868, 3936, 3955, 4005, 4027, 4033, 4046, 4102, 4206

`\cs_new_protected:Npn` 182, 186, 190, 401, 416, 433, 443, 449, 555, 600, 672, 697, 711, 1347, 1366, 1536, 1555, 1625, 1658, 1760, 1939, 2033, 2043, 2065, 2073, 2108, 2117, 2248, 2260, 2403, 2415, 2479, 2527, 2635, 2709, 2753, 2872, 2890, 2924, 2936, 3004, 3038, 3078, 3138, 3315, 3463, 3528, 3660, 3719, 3726, 3742, 3750, 3755, 3767, 3887, 4020, 4052, 4059, 4075, 4083, 4097, 4227, 4240, 4287, 4373, 4385

`\cs_new_protected_nopar:Nn` . . . 3706, 3829, 4039, 4181

`\cs_new_protected_nopar:Npn` . . . . . 3773, 4130

`\cs_set:Nn` . . . . . 2408

`\cs_set:Npn` . . . . . 2336, 2374, 4233

`\cs_set_eq:NN` . . 3638, 3639, 3775, 3995, 3996, 4132

`\cs_set_protected:Nn` . . . . . 815, 831, 843, 855

`\cs_set_protected:Npn` . 43, 52, 69, 77, 89, 95, 124, 149, 157, 465, 487, 519, 535, 582, 727, 747, 791, 810, 867, 876, 955, 972, 1383, 1494, 1741, 1802, 1976, 2015, 2051, 2328, 2660, 2823, 3036, 3076

`\cs_to_str:N` . . . . . 435, 458

`\cs_undefine:N` . . . . . 2439, 2440, 2441, 2442

## D

`\d` . . . . . 200

`\DeclareDocumentEnvironment` . . . . . 354

dim commands:

`\dim_abs:n` . . . . . 3009, 3014

`\dim_add:Nn` . . . . . 3473

`\dim_compare:nNTF` . 817, 833, 845, 857, 1349, 1368, 3006, 3011, 3017, 3023, 3025, 3027, 3196, 3219, 3323, 3341, 3465, 3511, 3579, 3870, 3938

`\dim_compare:nTF` . . . . . 2299, 2591

`\dim_gset_eq:NN` . . . . . 3588, 3947

`\dim_gzero:N` . . . . . 2633, 3622, 3981

`\dim_new:N` . . . . 65, 72, 73, 74, 91, 133, 166, 167, 173

`\dim_set:Nn` . . 446, 805, 2904, 3009, 3014, 3016, 3019, 3020, 3024, 3026, 3029, 3030, 3032, 3199, 3222, 3325, 3343, 3495, 3513, 3520, 3563, 3581, 3769, 3872, 3879, 3922, 3940

`\dim_set_eq:NN` 542, 589, 660, 664, 2899, 3047, 3087, 3230, 3351, 3570, 3573, 3574, 3760, 3929, 3932, 3933

`\dim_use:N` 818, 826, 1350, 1356, 2191, 2194, 2199, 2969, 2971, 3197, 3202, 3203, 3210, 3220, 3224, 3225, 3227

`\dim_zero:N` . . . . . 3234, 3353, 3474, 3475, 3476

`\dim_zero_new:N` . . . . . 3526, 3885

`\c_zero_dim` 820, 834, 846, 858, 1350, 1368, 2301, 2593, 3006, 3011, 3017, 3024, 3197, 3220, 3323, 3341, 3511, 3579, 3870, 3938

## E

`\end` . . 1353, 1371, 2150, 2185, 3249, 3266, 3366, 3383, 3600, 3616, 3959, 3975, 4296, 4304, 4310

`\endgroup` . . . . . 2533

`\endlist` . . . . . 33

`\endlist` . . . . . 342

`\endlrbox` . . . . . 3832, 4184

`\endminipage` . . . . . 33

`\endminipage` . . . . . 348

`enumext` . . . . . 5, 3109

enumext internal commands:

`\l__enumext_l_check_start_line_env_tl` . . . 31

`\l__enumext_l_ref_the_count_tl` . . . . . 38

`\l__enumext__resume_name_tl` . . . . . 58

`\__enumext_add_pre_parsep:` . . . 47, 996, 998, 998

`\__enumext_after_args_exec:` . 44, 891, 903, 3121

`\__enumext_after_args_exec_v:` . 45, 46, 907, 919, 3288

`\__enumext_after_args_exec_vii:` . . . 923, 947

`\__enumext_after_args_exec_viii:` . . . . . 951

`\__enumext_after_env:nn` . 67, 77, 78, 94, 107, 186, 186, 2547, 3276, 3605, 3867, 3964

`\__enumext_after_hyperref:` . . . 34, 363, 365, 365

`\__enumext_after_list:` . 93, 103, 110, 3126, 3256, 3256

`\l__enumext_after_list_args_v_tl` . . . . . 921

`\l__enumext_after_list_args_vii_tl` 949, 3823

`\l__enumext_after_list_args_viii_tl` 953, 4167

`\__enumext_after_list_v:` . . 96, 3293, 3373, 3373

`\__enumext_after_list_vii:` . . . 3646, 3677, 3677

`\__enumext_after_list_viii:` . . 4003, 4033, 4033

`\__enumext_after_star_env:nn` . . . . . 101

`\__enumext_after_stop_list:` . . . 44, 45, 891, 899, 3271

`\__enumext_after_stop_list_v:` 45, 907, 915, 3388

`\l__enumext_after_stop_list_v_tl` . . . . . 917

`\__enumext_after_stop_list_vii:` 923, 939, 3680

`\l__enumext_after_stop_list_vii_tl` . . . 941

`\__enumext_after_stop_list_viii:` . 943, 4036

`\l__enumext_after_stop_list_viii_tl` . . . 945

`\l__enumext_align_label_vii_str` . . 3815, 3819

`\l__enumext_align_label_viii_str` . 4159, 4163

`\l__enumext_align_label_X_str` . . . . . 157

`\c__enumext_all_envs_clist` . . 177, 486, 746, 809, 875, 890, 971, 1399

`\c__enumext_all_families_seq` . . 116, 4335, 4344, 4366

```

\__enumext_anskey__env_keys: . . . . . 79
\__enumext_anskey_env_clean: .. 80, 2557, 2561,
    2627
\__enumext_anskey_env_define_keys: 77, 2479,
    2485, 2541
\__enumext_anskey_env_exec: 78, 2482, 2537, 2537
\__enumext_anskey_env_keys: .. 2555, 2561, 2561
\__enumext_anskey_env_make:n 62, 77, 1785, 2479,
    2479, 2484
\__enumext_anskey_env_store: .. 79, 2556, 2561,
    2603
\__enumext_anskey_env_undefine_keys: . 77, 78,
    2506, 2558
\__enumext_anskey_env_undefine_keys:\__-
    enumext_rescan_anskey_env:n . . . . . 2479
\l__enumext_anskey_level_int .. 28, 2254, 2255
\__enumext_anskey_safe_inner:n .. 72, 73, 2224,
    2229, 2248
\__enumext_anskey_safe_outer: . 72, 2219, 2229,
    2229
\__enumext_anskey_show_wrap_arg:n . 75, 2403,
    2403, 2419, 2434
\__enumext_anskey_show_wrap_left:n 76, 2267,
    2415, 2415
\__enumext_anskey_wrapper:n . . . . . 1980, 2413
\__enumext_at_begin_document:n .. 33, 182, 182,
    339, 345
\__enumext_before_args_exec: 44, 891, 891, 3194
\__enumext_before_args_exec_v: .. 45, 907, 907,
    3322
\__enumext_before_args_exec_vii: .. 923, 923,
    3674
\__enumext_before_args_exec_viii: 927, 4030
\__enumext_before_env:nn 77, 186, 190, 2444, 2454,
    2464, 2539
\__enumext_before_keys_exec: 44, 891, 895, 3119
\__enumext_before_keys_exec_v: .. 45, 907, 911,
    3286
\__enumext_before_keys_exec_vii . . . . . 923
\__enumext_before_keys_exec_vii: 45, 931, 3634
\__enumext_before_keys_exec_viii: .. 45, 935,
    3991
\__enumext_before_list: ... 92, 3113, 3191, 3191
\__enumext_before_list_v: . 95, 3281, 3319, 3319
\__enumext_before_list_vii: ... 103, 3629, 3670,
    3670
\__enumext_before_list_viii: .. 109, 3987, 4027,
    4027
\l__enumext_before_no_starred_key_v_tl 913
\l__enumext_before_no_starred_key_vii_-
    tl . . . . . 933
\l__enumext_before_no_starred_key_viii_-
    tl . . . . . 937
\l__enumext_before_starred_key_v_tl ... 909
\l__enumext_before_starred_key_vii_tl . 925
\l__enumext_before_starred_key_viii_tl 929
\__enumext_calc_hspace:NNNNNNN 88, 3004, 3004,
    3035, 3040, 3080
\__enumext_check_ans_active: 64, 92, 1821, 1821,
    3195, 3673
\g__enumext_check_ans_item_tl . . . . . 82
\g__enumext_check_ans_key_bool 65, 66, 137, 314,
    1875, 1881, 1966
\l__enumext_check_ans_key_bool 65, 84, 85, 1806,
    1811, 1872, 1878
\__enumext_check_ans_key_hook: 65, 1869, 1869,
    3270, 3681
\__enumext_check_ans_level: 64, 1821, 1827, 1831
\__enumext_check_ans_log: 65, 66, 1915, 1915, 1970
\__enumext_check_ans_log_msg_greater: 1915,
    1921, 1934
\__enumext_check_ans_log_msg_less: 1915, 1919,
    1924
\__enumext_check_ans_log_msg_same_ok: 1915,
    1920, 1929
\__enumext_check_ans_msg_greater: 1891, 1897,
    1910
\__enumext_check_ans_msg_less: 1891, 1895, 1900
\__enumext_check_ans_msg_same_ok: 1891, 1896,
    1905
\__enumext_check_ans_show: .. 65, 66, 1891, 1891,
    1968
\g__enumext_check_ans_show_bool . . . . . 93
\l__enumext_check_answers_bool 62, 64, 72, 137,
    1783, 1810, 1825, 2124, 2148, 2155, 2183, 2221, 2748,
    2876, 2906, 3786
\__enumext_check_starred_cmd:n 31, 66, 82, 1939,
    1939, 3291, 3449, 4001
\g__enumext_check_starred_cmd_int 137, 1942,
    1948, 1953, 2946, 3419, 4109
\l__enumext_check_start_line_env_tl 137, 279,
    286, 293, 1945, 1951, 1954
\l__enumext_columns_sep_v_dim 3341, 3343, 3351
\l__enumext_columns_sep_vii_dim .. 3511, 3513,
    3522, 3567, 3851
\l__enumext_columns_sep_viii_dim . 3870, 3872,
    3881, 3926, 4204
\l__enumext_columns_v_int 1188, 3339, 3347, 3359,
    3364
\l__enumext_columns_vii_int .. 3516, 3519, 3523,
    3531, 3535, 3538, 3544, 3550, 3554, 3846, 3857
\l__enumext_columns_viii_int . 3875, 3878, 3882,
    3890, 3894, 3897, 3903, 3909, 3913, 4199, 4212
\l__enumext_counter_i_tl . . . . . 43, 425
\l__enumext_counter_ii_tl . . . . . 43, 426
\l__enumext_counter_iii_tl . . . . . 43, 427
\l__enumext_counter_iv_tl . . . . . 43, 428
\c__enumext_counter_style_tl . . . . . 30, 48, 206
\g__enumext_counter_styles_tl . 27, 36, 65, 436,
    454
\l__enumext_counter_v_tl . . . . . 43, 429, 680
\l__enumext_counter_vi_tl . . . . . 43, 430
\l__enumext_counter_vii_tl . . . . . 43, 431, 610
\l__enumext_counter_viii_tl . . . . . 43, 432, 627
\l__enumext_current_widest_dim 27, 65, 460, 543,
    590, 661, 665
\__enumext_default_item:n ... 2872, 2872, 2921
\__enumext_define_counters:Nn 26, 416, 416, 425,
    426, 427, 428, 429, 430, 431, 432
\__enumext_endminipage: . 33, 345, 348, 360, 3505,
    3831, 4183
\g__enumext_envir_name_tl 31, 34, 252, 266, 322,
    1753, 1758, 1768, 1903, 1908, 1913, 1927, 1932, 1937
\__enumext_execute_after_env: 32, 33, 62, 65, 66,
    76, 1956, 1956, 3276, 3867
\__enumext_fake_item: . . . . . 815, 815, 3067
\l__enumext_fake_item_indent_v_dim 834, 839
\l__enumext_fake_item_indent_v_tl 836, 2929,

```

[2933](#), [2941](#)  
`\l__enumext_fake_item_indent_vii_dim` [846](#), [851](#)  
`\l__enumext_fake_item_indent_vii_tl` [848](#), [3827](#)  
`\l__enumext_fake_item_indent_viii_dim` . [858](#),  
[863](#), [4175](#)  
`\l__enumext_fake_item_indent_viii_tl` .. [860](#),  
[4173](#), [4178](#)  
`\l__enumext_fake_item_indent_X_tl` ..... [95](#)  
`\__enumext_fake_item_vii:` .... [815](#), [843](#), [3097](#)  
`\__enumext_fake_item_viii:` .... [815](#), [855](#), [3102](#)  
`\__enumext_filter_save_key:n` .. [69](#), [2040](#), [2048](#),  
[2071](#), [2077](#), [2079](#), [2079](#), [4252](#), [4256](#), [4260](#), [4264](#), [4268](#),  
[4272](#)  
`\__enumext_filter_save_key_key:n` .. [69](#), [2079](#),  
[2084](#), [2088](#)  
`\__enumext_filter_save_key_pair:nn` [69](#), [2079](#),  
[2085](#), [2096](#)  
`\__enumext_filter_series:n` [57](#), [1506](#), [1506](#), [1548](#),  
[1560](#), [1565](#)  
`\__enumext_filter_series_key:n` [57](#), [1506](#), [1511](#),  
[1515](#)  
`\__enumext_filter_series_pair:nn` .. [57](#), [1506](#),  
[1512](#), [1524](#)  
`\g__enumext_footnote_arg_seq` . [154](#), [2845](#), [2858](#),  
[2868](#)  
`\g__enumext_footnote_int` . [154](#), [2852](#), [2855](#), [2857](#),  
[2859](#)  
`\g__enumext_footnote_int_seq` . [154](#), [2846](#), [2859](#),  
[2864](#), [2867](#)  
`\__enumext_footnotes_key_bool` ..... [34](#)  
`\l__enumext_footnotes_key_bool` [29](#), [34](#), [105](#), [144](#),  
[374](#), [379](#), [388](#), [3794](#), [3842](#), [4147](#), [4194](#)  
`\__enumext_footnotetext:nn` ... [2839](#), [2839](#), [2869](#)  
`\__enumext_getkeyans:nn` .. [114](#), [4236](#), [4240](#), [4240](#)  
`\__enumext_getkeyans_aux:n` [114](#), [4224](#), [4227](#), [4227](#)  
`\l__enumext_hyperref_bool` [29](#), [34](#), [144](#), [370](#), [391](#),  
[408](#), [2316](#), [2736](#), [3781](#), [4138](#)  
`\__enumext_hypertarget:nn` [34](#), [365](#), [393](#), [397](#), [413](#)  
`\__enumext_if_is_int:n` ..... [198](#)  
`\__enumext_if_is_int:nTF` ..... [198](#), [699](#), [713](#)  
`\__enumext_internal_mini_page:` .. [33](#), [350](#), [350](#),  
[3130](#), [3650](#)  
`\__enumext_is_not_nested:` . [26](#), [31](#), [90](#), [218](#), [218](#),  
[3131](#), [3651](#)  
`\__enumext_is_on_first_level:` . [26](#), [31](#), [90](#), [218](#),  
[242](#), [3136](#), [3658](#)  
`\g__enumext_item_anskey_int` [72](#), [82](#), [137](#), [309](#), [336](#),  
[337](#), [1888](#), [2223](#), [2750](#)  
`\__enumext_item_answer_diff:` [65](#), [66](#), [1884](#), [1884](#),  
[1963](#)  
`\g__enumext_item_answer_diff_int` . [65](#), [66](#), [143](#),  
[310](#), [1886](#), [1893](#), [1917](#)  
`\l__enumext_item_column_pos_vii_int` [103](#), [3538](#),  
[3544](#), [3550](#), [3554](#), [3561](#), [3709](#), [3846](#), [3849](#)  
`\l__enumext_item_column_pos_viii_int` .. [110](#),  
[3897](#), [3903](#), [3909](#), [3913](#), [3920](#), [4042](#), [4199](#), [4202](#)  
`\l__enumext_item_column_pos_X_int` ..... [157](#)  
`\g__enumext_item_count_all_vii_int` [103](#), [3562](#),  
[3710](#), [3857](#), [3864](#)  
`\g__enumext_item_count_all_viii_int` [110](#), [3921](#),  
[4043](#), [4211](#), [4219](#)  
`\g__enumext_item_count_all_X_int` ..... [157](#)  
`\g__enumext_item_number_int` .. [64](#), [137](#), [308](#), [335](#),  
[337](#), [1842](#), [1846](#), [1849](#), [1852](#), [1864](#), [1888](#), [2878](#), [2908](#),  
[3788](#)  
`\__enumext_item_peek_args_vii:` [103](#), [104](#), [3711](#),  
[3713](#), [3713](#)  
`\__enumext_item_peek_args_viii:` .. [110](#), [4044](#),  
[4046](#), [4046](#)  
`\__enumext_item_starred:` .. [87](#), [2964](#), [2964](#), [2982](#)  
`\l__enumext_item_starred_vii_bool` [3728](#), [3744](#),  
[3798](#)  
`\l__enumext_item_starred_viii_bool` [4061](#), [4077](#),  
[4151](#), [4171](#)  
`\l__enumext_item_starred_X_bool` ..... [157](#)  
`\__enumext_item_std:w` [33](#), [84](#)–[86](#), [99](#), [339](#), [343](#), [2881](#),  
[2887](#), [2911](#), [2929](#), [2933](#), [2941](#), [3486](#)  
`\g__enumext_item_symbol_aux_vii_tl` [3752](#), [3800](#),  
[3803](#), [3807](#), [3809](#)  
`\g__enumext_item_symbol_aux_X_tl` ..... [157](#)  
`\l__enumext_item_symbol_sep_vii_dim` .. [3761](#),  
[3769](#), [3806](#), [3808](#)  
`\g__enumext_item_symbol_tl` ... [85](#), [59](#), [121](#), [2896](#),  
[2970](#), [2987](#)  
`\l__enumext_item_symbol_vii_tl` ..... [3803](#)  
`\l__enumext_item_text_vii_box` [3793](#), [3834](#), [3841](#)  
`\l__enumext_item_text_viii_box` [4146](#), [4186](#), [4193](#)  
`\l__enumext_item_text_X_box` ..... [157](#)  
`\l__enumext_item_width_vii_dim` ... [3520](#), [3565](#),  
[3573](#), [3574](#)  
`\l__enumext_item_width_viii_dim` .. [3879](#), [3924](#),  
[3932](#), [3933](#)  
`\l__enumext_item_width_X_dim` ..... [157](#)  
`\l__enumext_itemindent_X_dim` ..... [69](#)  
`\l__enumext_itemsep_vii_skip` ..... [3863](#)  
`\l__enumext_itemsep_viii_skip` ..... [4218](#)  
`\l__enumext_joined_item_aux_vii_int` .. [3559](#),  
[3560](#), [3561](#), [3562](#), [3568](#)  
`\l__enumext_joined_item_aux_viii_int` . [3918](#),  
[3919](#), [3920](#), [3921](#), [3927](#)  
`\l__enumext_joined_item_aux_X_int` .... [157](#)  
`\__enumext_joined_item_vii:w` . [104](#), [3716](#), [3717](#),  
[3719](#), [3719](#)  
`\l__enumext_joined_item_vii_int` .. [3530](#), [3531](#),  
[3534](#), [3536](#), [3542](#), [3547](#), [3552](#), [3557](#), [3559](#), [3565](#)  
`\__enumext_joined_item_viii:w` . [110](#), [4049](#), [4050](#),  
[4052](#), [4052](#)  
`\l__enumext_joined_item_viii_int` . [3889](#), [3890](#),  
[3893](#), [3895](#), [3901](#), [3906](#), [3911](#), [3916](#), [3918](#), [3924](#)  
`\l__enumext_joined_item_X_int` ..... [157](#)  
`\l__enumext_joined_width_vii_dim` . [3563](#), [3570](#),  
[3573](#), [3824](#), [3836](#)  
`\l__enumext_joined_width_viii_dim` [3922](#), [3929](#),  
[3932](#), [4168](#), [4188](#)  
`\l__enumext_joined_width_X_dim` ..... [157](#)  
`\__enumext_keyans_addto_prop:n` [80](#), [2635](#), [2635](#),  
[2943](#), [3416](#)  
`\__enumext_keyans_addto_seq:n` . [81](#), [2709](#), [2709](#),  
[2945](#), [3418](#)  
`\__enumext_keyans_addto_seq_link:` [2709](#), [2730](#),  
[2732](#), [4108](#)  
`\__enumext_keyans_anspic_code:nnn` . [97](#), [3407](#),  
[3410](#), [3410](#)  
`\__enumext_keyans_default_item:n` .. [86](#), [2924](#),  
[2924](#), [2960](#)  
`\l__enumext_keyans_env_bool` [34](#), [3160](#), [3173](#), [3302](#),  
[3387](#)  
`\__enumext_keyans_fake_item:` .. [815](#), [831](#), [3057](#)



`\l__enumext_keyans_level_h_int` .. [28](#), [620](#), [647](#),  
[2239](#), [2470](#), [2687](#), [4007](#), [4008](#)  
`\l__enumext_keyans_level_int` .. [28](#), [1341](#), [2235](#),  
[2466](#), [2682](#), [3301](#), [3306](#), [3401](#)  
`\__enumext_keyans_make_label`: [36](#), [87](#), [2990](#), [2990](#),  
[3055](#)  
`\__enumext_keyans_mini_addvspace`: [52](#), [95](#), [1245](#),  
[1245](#), [3331](#)  
`\__enumext_keyans_mini_right_cmd:n` [54](#), [1343](#),  
[1366](#), [1366](#)  
`\__enumext_keyans_mini_set_vskip`: .. [51](#), [1183](#),  
[1183](#), [1247](#)  
`\__enumext_keyans_multi_addvspace`: [95](#), [1039](#),  
[1050](#), [3356](#)  
`\__enumext_keyans_multi_set_vskip`: [48](#), [1039](#),  
[1039](#), [1052](#)  
`\__enumext_keyans_multicols_start`: [95](#), [3335](#),  
[3337](#), [3337](#)  
`\__enumext_keyans_multicols_stop`: .. [96](#), [1370](#),  
[3362](#), [3362](#), [3386](#)  
`\__enumext_keyans_parse_keys:n` [3280](#), [3315](#), [3315](#)  
`\l__enumext_keyans_pic_above_int` .. [132](#), [3496](#),  
[3497](#), [3499](#)  
`\l__enumext_keyans_pic_above_skip` .. [99](#), [132](#),  
[3440](#), [3480](#)  
`\__enumext_keyans_pic_arg_two`: [98](#), [3438](#), [3468](#),  
[3468](#)  
`\l__enumext_keyans_pic_below_int` .. [132](#), [3496](#),  
[3497](#), [3500](#)  
`\l__enumext_keyans_pic_body_seq` .. [97-99](#), [132](#),  
[3405](#), [3445](#), [3504](#)  
`\__enumext_keyans_pic_do:n` [99](#), [3445](#), [3447](#), [3488](#),  
[3488](#), [3492](#)  
`\l__enumext_keyans_pic_level_int` .. [28](#), [1333](#),  
[2243](#), [2474](#), [2638](#), [2677](#), [2712](#), [2790](#), [3456](#), [3457](#)  
`\__enumext_keyans_pic_row:n` [99](#), [3490](#), [3493](#), [3493](#)  
`\__enumext_keyans_pic_safe_exec`: .. [98](#), [3434](#),  
[3454](#), [3454](#)  
`\__enumext_keyans_pic_skip_abs:N` .. [98](#), [3463](#),  
[3463](#), [3479](#)  
`\l__enumext_keyans_pic_width_dim` .. [132](#), [3495](#),  
[3502](#)  
`\__enumext_keyans_redefine_item`: .. [86](#), [2948](#),  
[2948](#), [3054](#)  
`\__enumext_keyans_ref`: ..... [40](#), [672](#), [690](#), [3056](#)  
`\__enumext_keyans_ref:n` ..... [40](#), [669](#), [672](#), [672](#)  
`\__enumext_keyans_safe_exec`: .. [3279](#), [3295](#), [3295](#)  
`\__enumext_keyans_show_ans`: .. [2753](#), [2761](#), [2780](#)  
`\__enumext_keyans_show_item_opt`: .. [2753](#), [2768](#),  
[2941](#), [3430](#), [4174](#)  
`\__enumext_keyans_show_left:n` .. [86](#), [2753](#), [2753](#),  
[2939](#), [3425](#)  
`\__enumext_keyans_show_pos`: .. [2753](#), [2765](#), [2788](#)  
`\__enumext_keyans_starred_item:n` .. [86](#), [2936](#),  
[2936](#), [2956](#)  
`\__enumext_keyans_start_line`: .. [26](#), [31](#), [273](#), [273](#),  
[3303](#), [3461](#), [4012](#)  
`\__enumext_keyans_store_ref`: .. [80](#), [2656](#), [2656](#),  
[2944](#), [3417](#), [4106](#)  
`\__enumext_keyans_store_ref_aux_i`: [81](#), [2656](#),  
[2668](#), [2671](#)  
`\__enumext_keyans_store_ref_aux_ii`: [81](#), [2656](#),  
[2697](#), [2699](#)  
`\__enumext_keyans_wrapper_opt:n` .. [1983](#), [2776](#)

`\l__enumext_label_copy_i_tl` .. [2370](#), [2675](#), [2680](#),  
[2685](#), [2690](#)  
`\l__enumext_label_copy_v_tl` ..... [2685](#)  
`\l__enumext_label_copy_vi_tl` ..... [2680](#)  
`\l__enumext_label_copy_vii_tl` [2345](#), [2356](#), [2387](#),  
[2675](#)  
`\l__enumext_label_copy_viii_tl` ..... [2690](#)  
`\l__enumext_label_copy_X_tl` ..... [146](#)  
`\l__enumext_label_fill_left_v_tl` ..... [2994](#)  
`\l__enumext_label_fill_left_X_tl` ..... [95](#)  
`\l__enumext_label_fill_right_v_tl` .... [3001](#)  
`\l__enumext_label_fill_right_X_tl` ..... [95](#)  
`\l__enumext_label_font_style_v_tl` [2995](#), [3429](#)  
`\l__enumext_label_font_style_vii_tl` ... [3812](#)  
`\l__enumext_label_font_style_viii_tl` .. [4156](#)  
`\l__enumext_label_i_tl` ..... [535](#)  
`\l__enumext_label_ii_tl` ..... [535](#)  
`\l__enumext_label_iii_tl` ..... [535](#)  
`\l__enumext_label_iv_tl` ..... [535](#)  
`\__enumext_label_style:Nnn` [26](#), [36](#), [449](#), [449](#), [464](#),  
[540](#), [587](#), [658](#), [662](#)  
`\l__enumext_label_v_tl` .. [80](#), [81](#), [655](#), [2643](#), [2717](#),  
[2782](#), [2817](#), [2938](#), [2942](#), [3283](#), [3424](#), [3426](#)  
`\l__enumext_label_vi_tl` .. [80](#), [81](#), [655](#), [2640](#), [2714](#),  
[3424](#), [3426](#), [3430](#)  
`\l__enumext_label_vii_tl` .. [582](#), [3739](#), [3764](#), [3771](#)  
`\l__enumext_label_viii_tl` [582](#), [4072](#), [4100](#), [4104](#)  
`\l__enumext_label_width_by_box` .. [65](#), [445](#), [446](#)  
`\__enumext_label_width_by_box:Nn` [36](#), [443](#), [443](#),  
[448](#), [460](#), [723](#)  
`\l__enumext_labelsep_i_dim` ... [2785](#), [2820](#), [4112](#),  
[4127](#)  
`\l__enumext_labelsep_v_dim` ..... [3346](#)  
`\l__enumext_labelsep_vii_dim` .. [3515](#), [3524](#), [3566](#),  
[3762](#), [3822](#), [3838](#)  
`\l__enumext_labelsep_viii_dim` [3874](#), [3883](#), [3925](#),  
[4166](#), [4175](#), [4190](#)  
`\l__enumext_labelwidth_i_dim` .. [2785](#), [2820](#), [4112](#),  
[4127](#)  
`\l__enumext_labelwidth_v_dim` ..... [3346](#)  
`\l__enumext_labelwidth_vii_dim` ... [3515](#), [3523](#),  
[3566](#), [3815](#), [3819](#), [3837](#)  
`\l__enumext_labelwidth_viii_dim` .. [3874](#), [3882](#),  
[3925](#), [4159](#), [4163](#), [4189](#)  
`\l__enumext_leftmargin_tmp_v_bool` .. [98](#), [3470](#)  
`\l__enumext_leftmargin_tmp_X_bool` ..... [69](#)  
`\l__enumext_leftmargin_tmp_X_dim` ..... [69](#)  
`\l__enumext_leftmargin_X_dim` ..... [69](#)  
`\__enumext_level`: [194](#), [194](#), [564](#), [567](#), [568](#), [577](#), [579](#),  
[818](#), [822](#), [826](#), [893](#), [897](#), [901](#), [905](#), [988](#), [990](#), [992](#), [994](#),  
[1027](#), [1029](#), [1031](#), [1033](#), [1037](#), [1070](#), [1073](#), [1092](#), [1101](#),  
[1107](#), [1112](#), [1116](#), [1127](#), [1131](#), [1132](#), [1137](#), [1173](#), [1177](#),  
[1350](#), [1356](#), [1403](#), [1405](#), [1407](#), [1410](#), [1417](#), [1419](#), [1421](#),  
[1424](#), [2035](#), [2037](#), [2039](#), [2067](#), [2068](#), [2070](#), [2126](#), [2134](#),  
[2138](#), [2142](#), [2408](#), [2411](#), [2412](#), [2880](#), [2881](#), [2885](#), [2886](#),  
[2887](#), [2894](#), [2896](#), [2900](#), [2901](#), [2904](#), [2910](#), [2911](#), [2966](#),  
[2969](#), [2971](#), [2978](#), [2979](#), [2980](#), [2983](#), [2986](#), [3116](#), [3118](#),  
[3166](#), [3179](#), [3186](#), [3197](#), [3199](#), [3202](#), [3203](#), [3205](#), [3210](#),  
[3217](#), [3220](#), [3222](#), [3224](#), [3225](#), [3226](#), [3227](#), [3230](#), [3236](#),  
[3241](#), [3247](#), [3250](#), [3252](#), [3258](#)  
`\l__enumext_level_h_int` .. [28](#), [226](#), [248](#), [261](#), [603](#),  
[640](#), [1839](#), [1855](#), [2364](#), [2381](#), [3652](#), [3653](#)  
`\l__enumext_level_int` .. [90](#), [28](#), [196](#), [235](#), [247](#), [262](#),  
[352](#), [1000](#), [1144](#), [1337](#), [1833](#), [1861](#), [1958](#), [2341](#), [2351](#),

2357, 2363, 2371, 2379, 2386, 2446, 2456, 3070, 3132,  
 3133, 3143, 3150, 3164, 3177, 3232, 3310, 3397, 3690,  
 3700, 4015  
 \\_enumext\_list\_arg\_two\_i: . . . . . 3036  
 \\_enumext\_list\_arg\_two\_ii: . . . . . 3036  
 \\_enumext\_list\_arg\_two\_iii: . . . . . 3036  
 \\_enumext\_list\_arg\_two\_iv: . . . . . 3036  
 \\_enumext\_list\_arg\_two\_v: . 86, 3036, 3285, 3471  
 \\_enumext\_list\_arg\_two\_vii: . . . . 3076, 3633  
 \\_enumext\_list\_arg\_two\_viii: . . . 3076, 3990  
 \l\_enumext\_listoffset\_v\_dim . . . . . 3348  
 \l\_enumext\_listparindent\_vii\_dim . . . 3825  
 \l\_enumext\_listparindent\_viii\_dim . . . 4169  
 \\_enumext\_log\_answer\_vars: . 33, 324, 332, 1965  
 \\_enumext\_log\_global\_vars: . 32, 324, 324, 1964  
 \\_enumext\_make\_label: 36, 84, 85, 87, 2974, 2974,  
 3065  
 \l\_enumext\_mark\_answer\_sym\_tl 71, 1989, 2196,  
 2423, 2792, 2805, 4116  
 \l\_enumext\_mark\_position\_str 121, 1993, 1994,  
 2020, 2021, 2194  
 \l\_enumext\_mark\_ref\_sym\_tl . . 2006, 2321, 2744  
 \\_enumext\_mini\_addvspace: . . 50, 92, 1166, 1166,  
 3207  
 \\_enumext\_mini\_addvspace\_vii: 53, 1319, 1319,  
 3591  
 \\_enumext\_mini\_addvspace\_viii: 53, 1319, 1325,  
 3950  
 \_\_enumext\_mini\_env\* . . . . . 350  
 \\_enumext\_mini\_right\_cmd:n 54, 1345, 1347, 1347  
 \\_enumext\_mini\_set\_vskip: . 49, 1067, 1067, 1168  
 \\_enumext\_mini\_set\_vskip\_vii: 52, 1262, 1262,  
 1321  
 \\_enumext\_mini\_set\_vskip\_viii: 52, 1262, 1284,  
 1327  
 \\_enumext\_minipage:w 33, 345, 347, 356, 3502, 3824,  
 4168  
 \l\_enumext\_minipage\_active\_v\_bool . . 95, 96,  
 3329, 3354, 3367, 3375  
 \g\_enumext\_minipage\_active\_vii\_bool . . 101,  
 3602, 3607, 3619  
 \l\_enumext\_minipage\_active\_vii\_bool . 3587,  
 3598  
 \g\_enumext\_minipage\_active\_viii\_bool 3961,  
 3966, 3978  
 \l\_enumext\_minipage\_active\_viii\_bool 3946,  
 3957  
 \g\_enumext\_minipage\_active\_X\_bool . . . 157  
 \l\_enumext\_minipage\_active\_X\_bool . . . 83  
 \g\_enumext\_minipage\_after\_skip 83, 1266, 1278,  
 3617, 3976  
 \l\_enumext\_minipage\_after\_skip 49, 50, 93, 96,  
 83, 1083, 1098, 1118, 1134, 1149, 1155, 1161, 1175,  
 1185, 1194, 1197, 1209, 1227, 1238, 1254, 1286, 1299,  
 1313, 3267, 3384  
 \g\_enumext\_minipage\_center\_vii\_bool . 3611,  
 3620  
 \g\_enumext\_minipage\_center\_viii\_bool 3970,  
 3979  
 \g\_enumext\_minipage\_center\_X\_bool . . . 157  
 \l\_enumext\_minipage\_hsep\_v\_dim . . . 95, 3327  
 \l\_enumext\_minipage\_hsep\_vii\_dim . . . 3585  
 \l\_enumext\_minipage\_hsep\_viii\_dim . . . 3944  
 \l\_enumext\_minipage\_left\_skip 49, 95, 83, 1075,  
 1090, 1109, 1124, 1171, 1181, 1186, 1192, 1201, 1218,  
 1230, 1250, 1260, 1264, 1269, 1273, 1287, 1291, 1305,  
 1323, 1329  
 \l\_enumext\_minipage\_left\_v\_dim 95, 3325, 3333  
 \l\_enumext\_minipage\_left\_vii\_dim 3581, 3593  
 \l\_enumext\_minipage\_left\_viii\_dim 3940, 3952  
 \l\_enumext\_minipage\_left\_X\_dim . . . . . 83  
 \g\_enumext\_minipage\_right\_skip 83, 1265, 1270,  
 1274, 3610, 3969  
 \l\_enumext\_minipage\_right\_skip . 49, 83, 1079,  
 1094, 1114, 1129, 1187, 1193, 1205, 1223, 1234, 1288,  
 1295, 1309, 1357, 1374  
 \l\_enumext\_minipage\_right\_v\_dim . . 95, 1368,  
 1373, 3323, 3327  
 \g\_enumext\_minipage\_right\_vii\_dim 101, 3589,  
 3609, 3622  
 \l\_enumext\_minipage\_right\_vii\_dim 101, 3579,  
 3584, 3590  
 \g\_enumext\_minipage\_right\_viii\_dim . . 3948,  
 3968, 3981  
 \l\_enumext\_minipage\_right\_viii\_dim . . 3938,  
 3943, 3949  
 \g\_enumext\_minipage\_right\_X\_dim . . . . . 157  
 \g\_enumext\_minipage\_right\_X\_skip . . . . 157  
 \g\_enumext\_minipage\_stat\_int . 92, 95, 83, 1362,  
 1379, 3206, 3260, 3265, 3330, 3377, 3382  
 \g\_enumext\_miniright\_code\_vii\_tl 101, 3615,  
 3621  
 \g\_enumext\_miniright\_code\_viii\_tl 3974, 3980  
 \g\_enumext\_miniright\_code\_X\_tl . . . . . 157  
 \\_enumext\_multi\_addvspace: . 47, 93, 1022, 1022,  
 3238  
 \\_enumext\_multi\_set\_vskip: . 47, 986, 986, 1024  
 \l\_enumext\_multicols\_above\_ii\_skip . . . 1005  
 \l\_enumext\_multicols\_above\_iii\_skip . . 1011  
 \l\_enumext\_multicols\_above\_iv\_skip . . . 1017  
 \l\_enumext\_multicols\_above\_v\_skip 1041, 1055,  
 1065  
 \l\_enumext\_multicols\_above\_X\_skip . . . . 77  
 \l\_enumext\_multicols\_below\_v\_skip 1045, 1059,  
 3369  
 \l\_enumext\_multicols\_below\_X\_skip . . . . 77  
 \\_enumext\_multicols\_start: 92, 3212, 3214, 3214  
 \\_enumext\_multicols\_stop: 93, 1352, 3244, 3244,  
 3269  
 \\_enumext\_newlabel:nn 29, 34, 75, 401, 401, 2397,  
 2703  
 \l\_enumext\_newlabel\_arg\_one\_tl 29, 34, 75, 81,  
 146, 2320, 2390, 2398, 2692, 2704, 2742  
 \l\_enumext\_newlabel\_arg\_two\_tl 29, 34, 74, 146,  
 2344, 2354, 2368, 2384, 2399, 2679, 2684, 2689, 2705  
 \\_enumext\_parse\_keys:n . . . 58, 3112, 3138, 3138  
 \\_enumext\_parse\_keys\_vii:n 58, 3628, 3660, 3660  
 \\_enumext\_parse\_keys\_viii:n . 3986, 4020, 4020  
 \\_enumext\_parse\_save\_key:n 69, 2060, 2065, 2065  
 \\_enumext\_parse\_save\_key\_vii:n 69, 2055, 2065,  
 2073  
 \\_enumext\_parse\_serie:n . . . . . 102  
 \\_enumext\_parse\_series:n . . 58, 91, 1536, 1536,  
 3146, 3666  
 \\_enumext\_parse\_store\_keys:n . . . . . 91  
 \l\_enumext\_parsep\_i\_skip 1003, 1005, 1147, 1195  
 \l\_enumext\_parsep\_ii\_skip . . . 1009, 1011, 1153  
 \l\_enumext\_parsep\_iii\_skip . . 1015, 1017, 1159



```

\l__enumext_parsep_vii_skip . . . . . 3826
\l__enumext_parsep_viii_skip . . . . . 4170
\l__enumext_partopsep_v_skip . 1057, 1061, 1221,
    1225, 1232, 1236, 1252, 1256
\l__enumext_partopsep_viii_skip . . . . . 1297
\__enumext_phantomsection: 34, 365, 394, 398, 414
\__enumext_print_footnote: . . . 2839, 2862, 3844,
    4196
\__enumext_print_keyans_box:NN 71, 2188, 2188,
    2201, 2410, 2784, 2819, 4112, 4127
\l__enumext_print_keyans_i_tl . . . . 4257, 4279
\l__enumext_print_keyans_ii_tl . . . 4261, 4280
\l__enumext_print_keyans_iii_tl . . 4265, 4281
\l__enumext_print_keyans_iv_tl . . . 4269, 4282
\l__enumext_print_keyans_starred_tl 114, 115,
    121, 4253, 4301
\l__enumext_print_keyans_vii_tl 114, 4273, 4283
\l__enumext_print_keyans_X_tl . . . . . 121
\__enumext_printkeyans:nnn 115, 4284, 4287, 4287
\__enumext_redefine_item: . 85, 2913, 2913, 3064
\l__enumext_ref_key_arg_tl 38, 48, 209, 557, 558,
    571, 602, 605, 616, 622, 633, 674, 675, 686
\l__enumext_ref_the_count_tl . 38, 48, 564, 567,
    570, 610, 612, 615, 627, 629, 632, 680, 682, 685
\__enumext_regex_counter_style: . . 30, 38, 204,
    204, 565, 611, 628, 681
\__enumext_register_counter_style:Nn . . 433,
    433, 438, 439, 440, 441, 442
\__enumext_remove_extra_parsep_vii: . . 3643,
    3853, 3853
\__enumext_remove_extra_parsep_viii: . 4000,
    4206, 4206
\__enumext_renew_footnote: . . . 2839, 2843, 3796,
    4149
\l__enumext_renew_the_count_v_tl 683, 692, 694
\l__enumext_renew_the_count_vii_tl 613, 642,
    644
\l__enumext_renew_the_count_viii_tl 630, 649,
    651
\l__enumext_renew_the_count_X_tl . . . . . 48
\__enumext_rescan_anskey_env:n . . 78, 79, 2527,
    2613, 2621
\__enumext_reset_global_bool: . . 300, 303, 312
\__enumext_reset_global_int: . . . 300, 302, 306
\__enumext_reset_global_tl: . . . . 300, 304, 318
\__enumext_reset_global_vars: . 32, 66, 300, 300,
    1973
\l__enumext_resume_active_bool 58, 60, 59, 1540,
    1660
\__enumext_resume_counter: . . 59, 60, 1658, 1664,
    1671
\__enumext_resume_counter:n . 58, 60, 1629, 1634,
    1658, 1658, 1728, 1736
\__enumext_resume_counter_save_ans: . . 60, 61,
    1658, 1669, 1701
\__enumext_resume_counter_series: . 60, 1658,
    1667, 1684
\g__enumext_resume_int . . . 59, 1581, 1675, 1676
\__enumext_resume_last:n . . 58, 1536, 1542, 1555
\l__enumext_resume_name_tl 59, 1577, 1585, 1588,
    1604, 1612, 1615, 1661, 1662, 1690, 1697
\__enumext_resume_save_counter: 58, 1568, 1568,
    3274, 3684
\__enumext_resume_series:n . 59, 1500, 1625, 1625

\__enumext_resume_starred: . 61, 1501, 1722, 1722
\g__enumext_resume_vii_int . 103, 59, 1608, 1680,
    1681
\__enumext_safe_exec: . . 33, 90, 3111, 3128, 3128
\__enumext_safe_exec_vii: . 33, 3627, 3648, 3648
\__enumext_safe_exec_viii: . . 3985, 4005, 4005
\l__enumext_series_name_tl . . . . . 60
\l__enumext_series_str . 58, 91, 1498, 1538, 1546,
    1547, 1549, 1551, 1572, 1575, 1579, 1599, 1602, 1606,
    3142, 3664
\__enumext_set_error:nn . . . . . 4373, 4383, 4385
\__enumext_set_parse:n . . . . . 4356, 4373, 4373
\l__enumext_setkey_tmpa_int . . . 116, 4349, 4353
\l__enumext_setkey_tmpa_seq . . 116, 4347, 4357,
    4363, 4365, 4367, 4380
\l__enumext_setkey_tmpa_tl . . . . 116, 4355, 4359
\l__enumext_setkey_tmpb_seq . . 116, 4348, 4351,
    4355, 4356
\l__enumext_setkey_tmpb_tl 116, 4375, 4377, 4378
\l__enumext_show_answer_bool . 2000, 2024, 2417,
    2759, 2773, 3421, 4110
\__enumext_show_length:nnn . . 44, 212, 212, 4474,
    4475, 4476, 4477, 4478, 4479, 4480, 4481, 4482, 4483,
    4489, 4490, 4491, 4492, 4493, 4494, 4495, 4496, 4497,
    4498
\l__enumext_show_position_bool . . . 2003, 2027,
    2421, 2763, 2774, 3422, 4114
\g__enumext_standar_bool 31, 90, 34, 225, 228, 246,
    315, 1570, 1635, 1647, 1673, 1686, 1724, 1860, 1873
\l__enumext_standar_bool . 90, 93, 34, 2349, 2362,
    2378, 3135, 3273
\l__enumext_standar_first_bool 31, 90, 34, 251,
    1557, 1704, 1766, 1773
\__enumext_standar_item_vii:w 104, 3724, 3726,
    3726
\__enumext_standar_item_viii:w 110, 4057, 4059,
    4059
\__enumext_standar_ref: . . . . 38, 555, 575, 3066
\__enumext_standar_ref:n . . . . 38, 547, 555, 555
\g__enumext_standar_series_tl . 59, 1559, 1560,
    1726, 1729
\g__enumext_starred_bool 31, 102, 103, 34, 234, 237,
    260, 316, 1597, 1640, 1651, 1678, 1693, 1732, 1838,
    1879, 2340, 2350, 2380, 2673, 3161, 3174, 3623
\l__enumext_starred_bool 102, 103, 34, 2275, 2281,
    2365, 2406, 2565, 2570, 3657, 3683
\__enumext_starred_columns_set_vii: . . 3509,
    3509, 3636
\__enumext_starred_columns_set_viii: . 3868,
    3868, 3993
\l__enumext_starred_first_bool . . 31, 34, 265,
    1562, 1713, 1766, 1773
\__enumext_starred_item:nn . . . 2890, 2890, 2919
\__enumext_starred_item_exec: . 111, 4102, 4102,
    4153
\__enumext_starred_item_vii:w 104, 3723, 3742,
    3742
\__enumext_starred_item_vii_aux_i:w . . 3742,
    3747, 3750
\__enumext_starred_item_vii_aux_ii:w . 3742,
    3748, 3753, 3755
\__enumext_starred_item_vii_aux_iii:w 3742,
    3758, 3767
\__enumext_starred_item_viii:w 110, 111, 4056,

```

[4075](#), [4075](#)  
`__enumext_starred_item_viii_aux_i:w` . . . [111](#),  
[4075](#), [4080](#), [4083](#)  
`__enumext_starred_item_viii_aux_ii:w` . . . [111](#),  
[4075](#), [4081](#), [4095](#), [4097](#)  
`__enumext_starred_joined_item_vii:n` [100](#), [104](#),  
[3528](#), [3528](#), [3721](#)  
`__enumext_starred_joined_item_viii:n` . . . [107](#),  
[110](#), [3887](#), [3887](#), [4054](#)  
`__enumext_starred_ref:` . . . . [39](#), [600](#), [638](#), [3094](#)  
`__enumext_starred_ref:n` . . . . [39](#), [594](#), [600](#), [600](#)  
`g__enumext_starred_series_tl` . . . [59](#), [1564](#), [1565](#),  
[1734](#), [1737](#)  
`__enumext_start_from:NNn` [40](#), [697](#), [697](#), [710](#), [732](#)  
`l__enumext_start_i_int` . . . . . [1676](#), [1688](#), [1707](#)  
`__enumext_start_item_tmp_vii:` [102](#), [3639](#), [3706](#),  
[3706](#)  
`__enumext_start_item_tmp_viii:` . . . [109](#), [3996](#),  
[4039](#), [4039](#)  
`__enumext_start_item_vii:w` [104](#), [105](#), [3734](#), [3739](#),  
[3764](#), [3771](#), [3773](#), [3773](#)  
`__enumext_start_item_viii:w` . . . [110](#), [4067](#), [4072](#),  
[4100](#), [4130](#), [4130](#)  
`g__enumext_start_line_tl` [31](#), [34](#), [253](#), [267](#), [321](#),  
[1903](#), [1908](#), [1913](#), [1927](#), [1932](#), [1937](#)  
`__enumext_start_list:nn` [33](#), [88](#), [98](#), [339](#), [341](#), [3115](#),  
[3282](#), [3435](#), [3631](#), [3988](#)  
`__enumext_start_mini_vii:` [103](#), [3577](#), [3577](#), [3675](#)  
`__enumext_start_mini_viii:` . . . [109](#), [3936](#), [3936](#),  
[4031](#)  
`__enumext_start_save_ans_msg:` [62](#), [1750](#), [1750](#),  
[1775](#)  
`__enumext_start_store_level:` . . . [91](#), [3114](#), [3155](#),  
[3155](#)  
`__enumext_start_store_level_vii:` [103](#), [3630](#),  
[3686](#), [3686](#)  
`l__enumext_start_vii_int` . . . [1681](#), [1695](#), [1716](#)  
`l__enumext_start_X_int` . . . . . [95](#), [727](#)  
`__enumext_stop_item_tmp_vii:` . . . [102](#), [103](#), [105](#),  
[3638](#), [3642](#), [3708](#), [3775](#)  
`__enumext_stop_item_tmp_viii:` [109](#), [110](#), [3995](#),  
[3999](#), [4041](#), [4132](#)  
`__enumext_stop_item_vii:` [105](#), [106](#), [3775](#), [3829](#),  
[3829](#)  
`__enumext_stop_item_viii:` [113](#), [4132](#), [4181](#), [4181](#)  
`__enumext_stop_list:` . . . [33](#), [339](#), [342](#), [3124](#), [3292](#),  
[3448](#), [3644](#), [4002](#)  
`__enumext_stop_mini_vii:` [101](#), [103](#), [3596](#), [3596](#),  
[3679](#)  
`__enumext_stop_mini_viii:` [110](#), [3936](#), [3955](#), [4035](#)  
`__enumext_stop_save_ans_msg:` . . . [62](#), [1750](#), [1755](#),  
[1962](#)  
`__enumext_stop_store_level:` . . . [91](#), [3125](#), [3155](#),  
[3184](#)  
`__enumext_stop_store_level_vii:` . . . [103](#), [3645](#),  
[3686](#), [3696](#)  
`l__enumext_store_active_bool` . . . [28](#), [62](#), [91](#), [102](#),  
[107](#), [1705](#), [1714](#), [1782](#), [2231](#), [3159](#), [3172](#), [3297](#), [3305](#),  
[3393](#), [3452](#), [3688](#), [3698](#), [4014](#)  
`__enumext_store_active_keys:n` [68](#), [2033](#), [2033](#),  
[3152](#)  
`__enumext_store_active_keys_vii:n` . . . [68](#), [102](#),  
[2033](#), [2043](#), [3667](#)  
`__enumext_store_addto_prop:n` [70](#), [80](#), [2108](#), [2108](#),

[2116](#), [2262](#), [2654](#), [4105](#)  
`__enumext_store_addto_seq:n` [70](#), [82](#), [2117](#), [2117](#),  
[2121](#), [2128](#), [2142](#), [2150](#), [2159](#), [2177](#), [2185](#), [2324](#), [2747](#)  
`l__enumext_store_anskey_arg_tl` . . . [28](#), [73](#), [107](#),  
[2272](#), [2277](#), [2279](#), [2284](#), [2291](#), [2294](#), [2304](#), [2309](#), [2312](#),  
[2318](#), [2324](#)  
`__enumext_store_anskey_code:nn` . . . [72](#), [73](#), [2225](#),  
[2260](#), [2260](#)  
`l__enumext_store_anskey_env_tl` [28](#), [78](#), [79](#), [107](#),  
[2549](#), [2551](#), [2606](#), [2613](#), [2621](#)  
`l__enumext_store_anskey_opt_tl` . . . [28](#), [79](#), [107](#),  
[2550](#), [2567](#), [2573](#), [2580](#), [2586](#), [2596](#), [2608](#), [2617](#)  
`__enumext_store_anskey_safe_outer:` . . . . . [72](#)  
`g__enumext_store_columns_break_bool` . . . [2489](#),  
[2564](#), [2629](#)  
`l__enumext_store_columns_break_bool` . . . [2204](#),  
[2274](#)  
`l__enumext_store_current_label_tl` [28](#), [80](#)–[82](#),  
[111](#), [107](#), [2637](#), [2640](#), [2643](#), [2650](#), [2652](#), [2654](#), [2711](#),  
[2714](#), [2717](#), [2723](#), [2728](#), [2738](#), [2747](#), [4085](#), [4090](#), [4091](#),  
[4104](#), [4105](#), [4107](#)  
`l__enumext_store_current_label_tmp_tl` . . . [28](#),  
[107](#), [2938](#), [2942](#)  
`l__enumext_store_current_opt_arg_tl` [28](#), [111](#),  
[107](#), [2757](#), [2770](#), [2776](#), [4093](#)  
`__enumext_store_internal_ref:` . . . [73](#), [74](#), [2265](#),  
[2326](#), [2326](#)  
`g__enumext_store_item_join_int` . . . [2492](#), [2571](#),  
[2575](#), [2630](#)  
`l__enumext_store_item_join_int` . . . [2207](#), [2282](#),  
[2286](#)  
`g__enumext_store_item_star_bool` . . . [2494](#), [2578](#),  
[2631](#)  
`l__enumext_store_item_star_bool` . . . [2209](#), [2289](#)  
`g__enumext_store_item_symbol_sep_dim` [2499](#),  
[2593](#), [2598](#), [2633](#)  
`l__enumext_store_item_symbol_sep_dim` [2214](#),  
[2301](#), [2306](#)  
`g__enumext_store_item_symbol_tl` . . . [2497](#), [2584](#),  
[2588](#), [2632](#)  
`l__enumext_store_item_symbol_tl` . . . [2212](#), [2292](#),  
[2296](#)  
`l__enumext_store_keyans_item_opt_sep_-`  
`tl` . . . . . [1986](#), [2648](#), [2650](#), [2721](#), [2725](#), [4088](#), [4090](#)  
`__enumext_store_level_close:` . . . [70](#), [2122](#), [2146](#),  
[3188](#)  
`__enumext_store_level_close_vii:` [2153](#), [2181](#),  
[3702](#)  
`__enumext_store_level_open:` . . . [70](#), [2122](#), [2122](#),  
[3167](#), [3180](#)  
`__enumext_store_level_open_vii:` . . . [2153](#), [2153](#),  
[3692](#)  
`g__enumext_store_name_tl` . . . [28](#), [62](#), [93](#), [107](#), [320](#),  
[327](#), [328](#), [329](#), [330](#), [1758](#), [1784](#), [1902](#), [1907](#), [1912](#), [1926](#),  
[1931](#), [1936](#), [1960](#)  
`l__enumext_store_name_tl` [28](#), [62](#), [64](#), [107](#), [1591](#),  
[1594](#), [1618](#), [1621](#), [1709](#), [1718](#), [1753](#), [1762](#), [1763](#), [1784](#),  
[1785](#), [1786](#), [1788](#), [1789](#), [1791](#), [1793](#), [1794](#), [1796](#), [1798](#),  
[1799](#), [1823](#), [2110](#), [2112](#), [2119](#), [2392](#), [2393](#), [2429](#), [2553](#),  
[2694](#), [2695](#), [2798](#), [2811](#), [4122](#)  
`l__enumext_store_ref_key_bool` [73](#), [2009](#), [2263](#),  
[2315](#), [2658](#), [2735](#)  
`l__enumext_store_save_key_vii_bool` . . . [2045](#),  
[2075](#)  
`l__enumext_store_save_key_vii_tl` [2047](#), [2048](#),

2076, 2077, 2157, 2167, 2173, 2177

\l\_\_enumext\_store\_save\_key\_X\_bool .. 68, 121

\l\_\_enumext\_store\_save\_key\_X\_tl .... 68, 121

\l\_\_enumext\_store\_upper\_level\_X\_bool .. 121

\\_\_enumext\_storing\_exec: 62, 77, 1760, 1776, 1780

\\_\_enumext\_storing\_set:n .. 62, 1745, 1760, 1760

\l\_\_enumext\_the\_counter\_v\_tl ..... 682

\l\_\_enumext\_the\_counter\_vii\_tl ..... 612

\l\_\_enumext\_the\_counter\_viii\_tl ..... 629

\l\_\_enumext\_the\_counter\_X\_tl ..... 48

\\_\_enumext\_tmp:n 43, 47, 52, 58, 69, 76, 77, 82, 89, 94, 95, 106, 124, 131, 149, 153, 157, 176, 810, 814, 1494, 1505, 1741, 1749, 1802, 1820, 1976, 2014, 2015, 2032, 2051, 2064, 2328, 2335, 2336, 2357, 2371, 2374, 2386, 2660, 2667, 3036, 3075, 3076, 3108

\\_\_enumext\_tmp:nn 465, 486, 487, 518, 519, 534, 727, 746, 791, 809, 867, 875, 876, 890, 955, 971, 972, 985, 1383, 1399, 2823, 2838

\\_\_enumext\_tmp:nnn 535, 551, 552, 553, 554, 582, 598, 599

\\_\_enumext\_tmp:nnnnn 747, 772, 775, 778, 780, 782, 785, 788

\\_\_enumext\_tmp:w ..... 4233, 4236

\l\_\_enumext\_tmpa\_vii\_int ..... 3519, 3522

\l\_\_enumext\_tmpa\_viii\_int ..... 3878, 3881

\l\_\_enumext\_tmpa\_X\_int ..... 157

\l\_\_enumext\_topsep\_v\_skip 1043, 1047, 1190, 1203, 1211, 1216, 1236, 1240, 3451, 3483

\l\_\_enumext\_topsep\_vii\_skip .. 1267, 1276, 1280

\l\_\_enumext\_topsep\_viii\_skip . 1289, 1311, 1315

\\_\_enumext\_undefine\_anskey\_env: . 66, 76, 1971, 2437, 2437

\l\_\_enumext\_vspace\_a\_star\_v\_bool ..... 1432

\l\_\_enumext\_vspace\_a\_star\_vii\_bool ... 1454

\l\_\_enumext\_vspace\_a\_star\_viii\_bool ... 1465

\l\_\_enumext\_vspace\_a\_star\_X\_bool ..... 95

\\_\_enumext\_vspace\_above: .. 55, 1400, 1400, 3193

\\_\_enumext\_vspace\_above\_v: . 55, 1428, 1428, 3321

\l\_\_enumext\_vspace\_above\_v\_skip .. 1430, 1434, 1436

\\_\_enumext\_vspace\_above\_vii: .. 56, 1450, 1450, 3672

\l\_\_enumext\_vspace\_above\_vii\_skip 1452, 1456, 1458

\\_\_enumext\_vspace\_above\_viii: . 56, 1450, 1461, 4029

\l\_\_enumext\_vspace\_above\_viii\_skip 1463, 1467, 1469

\l\_\_enumext\_vspace\_b\_star\_v\_bool ..... 1443

\l\_\_enumext\_vspace\_b\_star\_vii\_bool ... 1476

\l\_\_enumext\_vspace\_b\_star\_viii\_bool ... 1487

\l\_\_enumext\_vspace\_b\_star\_X\_bool ..... 95

\\_\_enumext\_vspace\_below: .. 55, 1414, 1414, 3272

\\_\_enumext\_vspace\_below\_v: . 56, 1439, 1439, 3389

\l\_\_enumext\_vspace\_below\_v\_skip .. 1441, 1445, 1447

\\_\_enumext\_vspace\_below\_vii: .. 56, 1472, 1472, 3682

\l\_\_enumext\_vspace\_below\_vii\_skip 1474, 1478, 1480

\\_\_enumext\_vspace\_below\_viii: . 56, 1472, 1483, 4037

\l\_\_enumext\_vspace\_below\_viii\_skip 1485, 1489, 1491

\\_\_enumext\_widest\_from:nNn .. 41, 711, 711, 726, 738

\g\_\_enumext\_widest\_label\_tl 27, 36, 65, 453, 457, 461

\l\_\_enumext\_wrap\_label\_opt\_v\_bool ..... 2932

\l\_\_enumext\_wrap\_label\_opt\_vii\_bool 104, 3733

\l\_\_enumext\_wrap\_label\_opt\_viii\_bool .. 110, 4066

\l\_\_enumext\_wrap\_label\_opt\_X\_bool ..... 95

\l\_\_enumext\_wrap\_label\_v\_bool 2928, 2932, 2940, 2996

\l\_\_enumext\_wrap\_label\_vii\_bool .. 104, 3732, 3737, 3745, 3813

\l\_\_enumext\_wrap\_label\_viii\_bool . 110, 4065, 4070, 4078, 4157

\l\_\_enumext\_wrap\_label\_X\_bool ..... 95

\\_\_enumext\_wrapper\_label\_v:n ..... 2998, 3430

\\_\_enumext\_wrapper\_label\_vii:n ..... 3816

\\_\_enumext\_wrapper\_label\_viii:n ..... 4160

\l\_\_enumext\_write\_aux\_file\_tl . 29, 75, 81, 146, 2395, 2401, 2701, 2707

\\_\_enumext\_zero\_parsep: ... 50, 1087, 1142, 1142

enumext\* ..... 5, 3625

enumXi ..... 425

enumXii ..... 425

enumXiii ..... 425

enumXiv ..... 425

enumXv ..... 425

enumXvi ..... 425

enumXvii ..... 425

enumXviii ..... 425

Environments provide by enumext:

anskey\* ..... 28, 62, 66, 76–78

enumext\* .. 25, 26, 29–31, 33, 35, 38, 39, 41–46, 52, 53, 56–59, 61–64, 67–76, 78, 79, 81, 83, 84, 89–91, 102, 103, 105, 107, 108, 110, 112, 114, 115, 118, 121

enumext 25, 26, 30, 31, 33, 35–42, 44, 46–55, 57–59, 61–64, 67–76, 78, 79, 81, 83–85, 87, 88, 90, 91, 94, 98, 99, 101, 103, 114, 115, 118, 119

keyans\* 25, 26, 28–31, 35, 38–46, 52, 53, 56, 62, 63, 66, 67, 70, 77, 80, 84, 89, 109, 118, 121

keyanspic 25, 26, 28, 31, 35, 36, 40, 53, 62, 63, 66, 70, 77, 80–82, 96–99, 120

keyans 25, 26, 28, 30, 31, 35, 36, 40–42, 44–46, 48, 51–56, 62, 63, 66, 67, 70, 77, 80–82, 86–88, 94, 96–98, 101, 110, 118, 120

Environments:

list ..... 30, 33, 88, 90

lrbox ..... 99, 105, 106, 112, 113

minipage .. 30, 33, 46, 48, 49, 96, 98, 99, 105, 106, 113

multicols ..... 46–49, 54, 92, 93, 95, 96

scontents ..... 77, 78

exp commands:

\exp\_after:wN ..... 4236

\exp\_args:Ne ..... 2610, 2618, 3149, 4224

\exp\_not:N . 56, 456, 570, 615, 632, 685, 824, 838, 839, 850, 851, 862, 863, 2320, 2426, 2427, 2740, 2795, 2796, 2808, 2809, 4119, 4120, 4233

\exp\_not:n 255, 269, 281, 288, 295, 509, 529, 570, 571, 615, 616, 632, 633, 685, 686, 825, 1522, 1534, 1997, 2094, 2106, 2286, 2296, 2306, 2320, 2321, 2398, 2575, 2588, 2598, 2704, 2742, 2744

## F

\fbox ..... 1981

file commands:

\file_input_stop:	4645
first	876
font	465
\footnote	84
\footnote	84, 2847
\footnotemark	2857
\footnotesize	2427, 2796, 2809, 4120
\footnotetext	2841

G

\getkeyans	15, 113, 4222
group commands:	
\group_begin:	2220, 2425, 2529, 2605, 2794, 2807, 3792, 3811, 4118, 4145, 4155, 4244, 4278
\group_end:	2227, 2432, 2625, 2801, 2814, 3821, 3833, 4125, 4165, 4185, 4246, 4285

H

\hbadness	3840, 4192
hbox commands:	
\hbox_set:Nn	445
\hfill	495, 499, 504, 505, 1354, 1372, 2320, 2740, 3601, 3960
hook commands:	
\hook_gput_code:nnn	9, 184, 188, 192, 363
\hook_gremove_code:nn	78, 2545
\hook_gset_rule:nnnn	364
\hook_if_empty:nTF	2543
\hspace	3851, 4204
\hyperlink	74, 82
\hyperlink	2320, 2740
\hypertarget	34
\hypertarget	393

I

\IfHyperBoolean	371
\IfPackageLoadedTF	11, 19, 367, 381
\ignorespaces	827
\inputlineno	255, 269, 281, 288, 295
int commands:	
\int_add:Nn	3561, 3920
\int_case:nn	1000, 1144, 1833, 1855, 1893, 1917
\int_compare:nNnTF	352, 603, 620, 640, 647, 1069, 1188, 1333, 1337, 1341, 1941, 1947, 1958, 2235, 2239, 2243, 2255, 2446, 2456, 2466, 2470, 2474, 2638, 2677, 2682, 2687, 2712, 2790, 3133, 3143, 3164, 3177, 3216, 3232, 3246, 3260, 3306, 3310, 3339, 3364, 3377, 3397, 3401, 3457, 3531, 3541, 3557, 3653, 3690, 3700, 3846, 3855, 3890, 3900, 3916, 4008, 4015, 4198, 4208, 4353
\int_compare_p:nNn	226, 235, 247, 248, 261, 262, 1839, 1861, 2282, 2341, 2351, 2363, 2364, 2379, 2381, 2571
\int_decr:N	3560, 3919
\int_eval:n	337, 2112, 2393, 2427, 2695, 2796, 2809, 3051, 3093, 3549, 3908, 4120
\int_from_alph:n	705, 719
\int_from_roman:n	707, 721
\int_gadd:Nn	3562, 3921
\int_gdecr:N	1842, 1846, 1849, 1852, 1864
\int_gincr:N	1675, 1680, 2223, 2750, 2878, 2908, 2946, 3206, 3330, 3419, 3710, 3788, 4043, 4109
\int_gset:Nn	1886, 2855
\int_gset_eq:NN	1574, 1581, 1587, 1593, 1601, 1608, 1614, 1620, 2852
\int_gzero:N	308, 309, 310, 1362, 1379, 1953, 2630, 3265, 3382, 3864, 4219
\int_if_exist:NTF	1549, 1585, 1591, 1612, 1618, 1796

\int_incr:N	2254, 3132, 3301, 3456, 3652, 3709, 4007, 4042
\int_mod:nn	3857, 4210
\int_new:N	28, 29, 30, 31, 32, 33, 59, 60, 83, 99, 118, 134, 135, 140, 141, 142, 143, 154, 160, 161, 162, 163, 164, 1551, 1799
\int_set:Nn	701, 705, 707, 1688, 1695, 1707, 1716, 2530, 3496, 3497, 3519, 3530, 3536, 3552, 3840, 3878, 3889, 3895, 3911, 4192, 4349
\int_set_eq:NN	1676, 1681, 3559, 3918
\int_sign:n	1888
\int_step_function:nnN	2357, 2371, 2386
\int_step_inline:nnn	3498
\int_to_roman:n	196, 2337, 2375
\int_use:N	330, 335, 336, 1070, 1690, 1697, 1709, 1718, 3051, 3070, 3093, 3150, 3217, 3226, 3241, 3247, 3534, 3535, 3547, 3893, 3894, 3906
\int_zero:N	3849, 4202
\c_one_int	3519, 3538, 3544, 3550, 3554, 3557, 3878, 3897, 3903, 3909, 3913, 3916
\c_zero_int	2341, 2351, 2363, 2364, 2379, 2381, 3690, 3700, 3860, 4215
\item	33, 44-46, 71, 84, 96, 97, 99, 102, 109
\item	84, 86, 103, 105, 110, 112, 343, 2130, 2136, 2161, 2169, 2279, 2714, 2717, 2915, 2950, 3637, 3639, 3994, 3996, 4107
\item*	5, 14, 66, 2948
item-pos*	2823
item-sym*	2823
\itemindent	89
\itemindent	88
itemindent	791
\itemsep	98, 99
\itemsep	3472, 3478
\itemwidth	3526, 3570, 3574, 3885, 3929, 3933

K

keyans	13, 3277
keyans*	13, 3983
keyanspic	14, 3432

Keys for command provide by enumext:

break-col	72, 73, 77, 79
item-join	72, 73, 77, 79
item-pos*	72, 73, 77, 79
item-star	72, 73, 77, 79
item-sym*	72, 73, 77, 79

Keys for environments provide by enumext:

above*	27, 55, 56
above	27, 55, 56, 92, 95, 103, 109
after	44, 45, 93, 96, 103, 110
align	27, 37, 87, 105, 117
before*	44, 45, 92, 103, 109
before	44, 45, 95
below*	27, 55, 56
below	27, 55, 56, 93, 96, 103, 110
check-ans	28, 30, 31, 61-63, 65, 66, 69, 82, 84, 85, 92-94, 107, 119
columns-sep	46, 92, 95
columns	27, 46, 49, 54, 92, 95
first	44-46, 105
font	36, 87, 105
item-pos*	83
item-sym*	28, 83, 85
item*-sep	85
itemindent	27, 42, 43, 86, 105

itemsep	41, 90
labelsep	36, 85, 88, 105
labelwidth	35–38, 40, 41, 88
label	26, 27, 35–37, 40, 41, 99
lisparindent	90
list-indent	27, 42, 43, 98
list-offset	42, 43
listparindent	42, 105
mark-ans	67, 69, 76
mark-pos	67, 117
mark-ref	67, 69, 74
mini-env	27, 33, 46, 54, 69, 84, 92, 95, 101, 103, 108, 109
mini-right*	27, 46, 69, 101
mini-right	27, 46, 53, 69, 101
mini-rigth*	30
mini-rigth	30
mini-sep	27, 46, 69, 92, 95
no-store	28, 61–64, 69, 72
noitemsep	41, 50
nosep	41, 50
parindent	90
parsep	41, 89, 90, 105
partopsep	41
ref	26, 30, 37, 38, 40, 118
resume*	26, 57, 58, 61, 62, 69, 93
resume	26, 32, 57–62, 69, 93, 103
rightmargin	42
save-ans	28, 32, 57–62, 64–66, 68–70, 72, 76, 77, 80, 81, 86, 91, 94, 97, 103, 110, 111, 113, 114, 119
save-key	28, 57, 68, 91
save-pos	69
save-ref	29, 34, 67, 69, 73, 74, 80, 82, 86, 111
save-sep	67, 69, 111
series	26, 57–61, 69, 91, 93
show-ans	67, 69, 71, 73, 76, 86, 111
show-length	30, 44, 118
show-pos	28, 67, 71, 73, 76, 82, 86, 111
start	27, 30, 40, 41, 57
store-key	68
topsep	41
widest	27, 30, 41
wrap-ans	67, 69, 71, 75
wrap-label*	36, 84, 87, 104, 105, 110
wrap-label	36, 87, 104, 105, 110
wrap-opt	67, 69
keys commands:	
\keys_define:nn	467, 489, 521, 537, 584, 655, 729, 749, 793, 812, 869, 878, 957, 974, 1385, 1496, 1743, 1804, 1978, 2017, 2053, 2058, 2202, 2487, 2508, 2825, 4249, 4318
\l_keys_key_str	4459
\keys_precompile:nnN	115, 4248, 4251, 4255, 4259, 4263, 4267, 4271
\keys_set:nn	481, 980, 1390, 1395, 1637, 1642, 1729, 1737, 2270, 2617, 3145, 3149, 3317, 3665, 4024, 4320, 4321, 4322, 4323, 4324, 4325, 4326, 4327, 4328, 4329, 4330, 4331, 4332, 4370
keyval commands:	
\keyval_parse:NNn	1510, 2083
L	
label	535, 582, 655
Labels provide by enumext:	
\Alph*	35, 36
\Roman*	35, 36
\alph*	35, 36

\arabic*	30, 35, 36
\roman*	35, 36
\labelsep	99
\labelsep	3473, 3476
labelsep	465
\labelwidth	36, 99
\labelwidth	3473, 3474
labelwidth	465
\leftmargin	89
\leftmargin	88, 3473
legacy commands:	
\legacy_if:nTF	3776, 3779, 4133, 4136
\legacy_if_gset_false:n	357
\legacy_if_set_false:n	3778, 4135
\legacy_if_set_true:n	3738, 3763, 3770, 3783, 4071, 4099, 4140
\linewidth	92, 95
\linewidth	3201, 3327, 3495, 3522, 3583, 3881, 3942
\list	33
\list	341
list-indent	791
list-offset	791
\listparindent	3475
listparindent	791
\lrbox	3793, 4146
M	
\makebox	99
\makebox	2192, 2194, 2970, 3807, 3815, 3819, 4159, 4163
\makelabel	84, 87, 99
\makelabel	87, 2976, 2992
\makesavenoteenv	387
mark-ans	1976
mark-pos	1976, 2015
mark-ref	1976
mini-env	955
mini-sep	955
\minipage	33
\minipage	347
\miniright	10, 53, 1331, 3263, 3380
\miniright*	10
mode commands:	
\mode_if_vertical:TF	1025, 1053, 1169, 1248
\mode_leave_vertical:	824, 838, 850, 862, 2161, 2169, 2190, 2968, 3805
msg commands:	
\msg_error:nn	2252, 2257, 3308, 3312, 3399, 3459, 3655, 4010, 4017, 4333
\msg_error:nnn	560, 607, 624, 677, 1335, 1339, 1364, 1381, 1649, 1653, 1768, 2450, 2460, 2468, 2472, 2476, 4238, 4243, 4315, 4386
\msg_error:nnnn	2233, 2237, 2241, 2245, 3299, 3395, 3403, 4298
\msg_error:nnnnn	508, 528, 1996
\msg_fatal:nn	3134
\msg_fatal:nnn	419
\msg_info:nnn	13, 16, 21, 24, 369, 383
\msg_line_context:	4424, 4429, 4434, 4463, 4468, 4473, 4488, 4503, 4507, 4511, 4515, 4519, 4523, 4530, 4537, 4543, 4557, 4561, 4566, 4570, 4575, 4579, 4583, 4588, 4593, 4597, 4602, 4607, 4611, 4616, 4620, 4625, 4630, 4635, 4639, 4643
\msg_log:nnn	1788, 1793, 1798
\msg_log:nnnnn	334, 1926, 1931, 1936
\msg_log:nnnnnn	326



\msg_new:nnn	4387, 4391, 4395, 4399, 4404, 4421, 4426, 4431, 4436, 4445, 4453, 4457, 4461, 4466, 4471, 4486, 4501, 4505, 4509, 4513, 4517, 4521, 4525, 4534, 4540, 4546, 4550, 4554, 4559, 4564, 4568, 4573, 4577, 4581, 4586, 4591, 4595, 4600, 4605, 4609, 4614, 4618, 4623, 4628, 4633, 4637, 4641
\msg_new:nnnn	4412
\msg_term:nnnn	1752, 1757, 3060, 3070, 3099, 3104
\msg_term:nnnnn	1907
\msg_warning:nn	3262, 3379
\msg_warning:nnnn	1944, 1950, 3008, 3013, 3533, 3546, 3892, 3905
\msg_warning:nnnnn	1902, 1912
\multicolsep	92, 95
\multicolsep	3231, 3352
\myenv	4410
\mypkg	4410
N	
\NeedsTeXFormat	3
\newcounter	422
\NewDocumentCommand	1331, 2217, 3391, 4222, 4276, 4340
\NewDocumentEnvironment	3109, 3277, 3432, 3625, 3983
\newenvsc	2481
\newlabel	35
\newlabel	405
no-store	1802
\noindent	102, 109
\noindent	3208, 3332, 3592, 3638, 3848, 3951, 3995, 4201
\nointerlineskip	3208, 3332, 3592, 3951
noitemsep	747
\nopagebreak	1036, 1064, 1180, 1259, 1322, 1328
\normalfont	2426, 2795, 2808, 4119
nosep	747
P	
Packages:	
enumext	25, 37, 61, 88, 97, 117
enumitem	35
expl3	99
footnotehyper	34
hyperref	29, 30, 34, 35, 74, 82, 105, 117
lua-visual-debug	49
multicol	25, 117
scontents	25, 76, 79
shortlst	99
\par	1036, 1064, 1180, 1259, 1322, 1328, 1357, 1374, 2405, 3252, 3267, 3369, 3384, 3507, 3610, 3617, 3848, 3862, 3969, 3976, 4201, 4217
\parindent	3825, 4169
\parsep	47, 50, 98, 99
\parsep	2162, 2170, 3090, 3472, 3479, 3484
parsep	747
\parskip	3826, 4170
\partopsep	99
\partopsep	3091, 3477
partopsep	747
peek commands:	
\peek_meaning:N	3715, 3729, 3746, 3757, 4048, 4062, 4079
\peek_meaning_remove:N	3722, 4055
\peek_remove_spaces:n	2954
\phantomsection	34
\phantomsection	394
prg commands:	
\prg_do_nothing:	398

\prg_new_protected_conditional:N	198
\prg_replicate:nn	215
\prg_return_false:	202
\prg_return_true:	201
\printkeyans	16, 114, 4276
prop commands:	
\prop_count:N	328, 2112, 2393, 2429, 2695, 2798, 2811, 4122
\prop_gput_if_not_in:N	2110
\prop_if_exist:N	1786, 4242
\prop_item:N	4245
\prop_new:N	1789
\ProvidesExplPackage	4

R

\raggedcolumns	3240, 3358
\ref	74, 80
ref	535, 582, 655
\refstepcounter	3785, 4142
regex commands:	
\regex_match:nnTF	200, 704, 706, 718, 720
\regex_replace_once:nnN	208
\renewcommand	570, 615, 632, 685
\RenewDocumentCommand	2847, 2915, 2950, 2976, 2992
\RequirePackage	17, 25
resume	1494
resume*	1494
rightmargin	791
\Roman	36, 40, 41
\Roman	441
\roman	36, 40, 41
\roman	442, 553, 4266

S

save-ans	1741
save-key	2051
save-ref	1976
save-sep	1976
scan commands:	
\scan_stop:	99, 3486, 3637, 3994, 4233, 4236
scontents internal commands:	
\l_scontents_fname_out_tl	2518
\__scontents_parse_environment_keys:n	2524
\__scontents_rescan_tokens:n	2531
\l_scontents_storing_bool	2516
\l_scontents_writing_bool	2517
seq commands:	
\seq_clear:N	4347
\seq_const_from_clist:Nn	4335
\seq_count:N	329, 3445, 4351
\seq_gclear:N	2845, 2846
\seq_gput_right:Nn	2119, 2858, 2859
\seq_if_empty:N	2864, 4291, 4365
\seq_if_exist:N	1791, 4289
\seq_if_in:NnTF	4296
\seq_item:Nn	2553, 3504
\seq_map_function:NN	4356
\seq_map_inline:Nn	4303, 4309, 4344, 4366, 4367
\seq_map_pairwise_function:NNN	2866
\seq_new:N	119, 120, 132, 155, 156, 1794
\seq_pop_left:NN	4355
\seq_put_right:Nn	3405, 4363, 4380
\seq_set_from_clist:Nn	4348
\seq_set_map_e:NNn	4357
\seq_show:N	4293

series .....	1494
\setcounter .....	715, 719, 721, 3051, 3093, 3450
\setenumext .....	6, 116, 4340
\setlength .....	2163, 2171
show-ans .....	1976, 2015
show-length .....	867
show-pos .....	2015
skip commands:	
\skip_add:Nn	1005, 1011, 1017, 1027, 1031, 1055, 1059, 1149, 1155, 1161, 1171, 1175, 1197, 1250, 1254, 3472
\skip_eval:n	2162, 2170
\skip_gset:Nn	1270, 1274, 1278
\skip_gzero_new:N	1265, 1266
\skip_horizontal:N	839, 851, 863, 3808, 3822, 4166
\skip_horizontal:n	825, 2191, 2199, 2969, 2971, 3806, 4175
\skip_if_eq:nnTF	1003, 1009, 1015, 1072, 1106, 1147, 1153, 1159, 1190, 1195, 1216, 1267, 1289, 1402, 1416, 1430, 1441, 1452, 1463, 1474, 1485
\skip_new:N	79, 80, 84, 85, 86, 87, 88, 136, 174
\skip_set:Nn	988, 992, 1041, 1045, 1075, 1079, 1083, 1090, 1094, 1098, 1109, 1114, 1118, 1124, 1129, 1134, 1192, 1193, 1194, 1201, 1205, 1209, 1218, 1223, 1227, 1230, 1234, 1238, 1269, 1273, 1291, 1295, 1299, 1305, 1309, 1313, 3466, 3480
\skip_set_eq:NN	3049, 3089, 3090, 3825, 3826, 4169, 4170
\skip_use:N	990, 994, 1029, 1033, 1037, 1057, 1061, 1073, 1092, 1101, 1107, 1112, 1116, 1127, 1131, 1132, 1137, 1173, 1177, 1203, 1403, 1407, 1410, 1417, 1421, 1424, 3252
\skip_zero:N	3091, 3231, 3352, 3477, 3478
\skip_zero_new:N	1185, 1186, 1187, 1264, 1286, 1287, 1288
\c_zero_skip	1003, 1009, 1015, 1073, 1107, 1147, 1153, 1159, 1190, 1195, 1216, 1267, 1289, 1403, 1417, 1430, 1441, 1452, 1463, 1474, 1485
\small .....	4254, 4258, 4262, 4266, 4270, 4274
\star .....	2829
start .....	727
\stepcounter .....	2851, 3412
str commands:	
\c_backslash_str	4424, 4429, 4434, 4439, 4441, 4443, 4448, 4450, 4548, 4552, 4556, 4570, 4571, 4575, 4583, 4584, 4588, 4589, 4620, 4621, 4625, 4630, 4631
\c_colon_str	2392, 2694, 4233
\c_left_brace_str	4529, 4536, 4542
\c_right_brace_str	4529, 4536, 4542
\str_case:nn	220, 275
\str_case:nnTF	1517, 1526, 2090, 2098
\str_clear:N	3142, 3664
\str_count:n	215
\str_if_empty:NTF	1538, 1579, 1606
\str_if_eq:nnTF	3052, 3095
\str_if_in:nnTF	4229
\str_new:N	122, 169
\str_set:Nn	524, 525, 526, 1993, 1994, 2020, 2021
\string .....	387
\strutbox	1077, 1081, 1085, 1096, 1100, 1111, 1120, 1126, 1136, 1149, 1155, 1161, 1192, 1193, 1194, 1197, 1207, 1211, 1220, 1227, 1232, 1240, 1269, 1270, 1273, 1280, 1293, 1301, 1307, 1315, 3482

T	
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@auxout .....	403
\@currentvar .....	220, 275
\protected@write .....	403
tex commands:	
\tex_newlinechar:D .....	2530
text commands:	
\text_expand:n .....	4225
\textasteriskcentered .....	1990, 2007
\thepage .....	409
tl commands:	
\c_space_tl .....	2776, 4473, 4488, 4511, 4515
\tl_clear:N	494, 500, 1954, 2037, 2047, 2068, 2076, 2272, 2549, 2550, 2637, 2711, 4085
\tl_clear_new:N	451
\tl_const:Nn	48, 435
\tl_gclear:N	320, 321, 322, 1559, 1564, 2632, 2987, 3621, 3809, 3980
\tl_gclear_new:N	1546
\tl_gput_right:Nn	436
\tl_greplace_all:Nnn	457
\tl_gset:Nn	252, 253, 266, 267, 1547, 1560, 1565, 1784, 2551, 3752
\tl_gset_eq:NN	453, 2896, 3802
\tl_if_blank:nTF	3800
\tl_if_empty:NTF	558, 577, 605, 622, 642, 649, 675, 692, 1572, 1577, 1599, 1604, 1662, 1726, 1734, 1763, 1823, 1960, 2126, 2157, 2292, 2584, 2606, 2608, 2648, 2721, 2770, 2966, 4088, 4378
\tl_if_empty:nTF	1627, 2250
\tl_if_exist:NTF	1632
\tl_if_novalue:nTF	2268, 2645, 2719, 2755, 2849, 2874, 2892, 2897, 2926, 3140, 3443, 3662, 4022, 4086, 4342
\tl_map_inline:Nn	206, 454
\tl_new:N	41, 42, 45, 50, 51, 54, 55, 61, 63, 64, 66, 67, 100, 101, 102, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 121, 123, 126, 127, 139, 146, 147, 148, 151, 168, 171
\tl_put_left::Ne	2573
\tl_put_left:Nn	2134, 2167, 2277, 2567, 2580, 2586, 2596, 2782, 2817, 4104, 4107
\tl_put_right:Nn	452, 568, 613, 630, 683, 2138, 2173, 2279, 2284, 2291, 2294, 2304, 2309, 2312, 2318, 2344, 2354, 2368, 2384, 2390, 2395, 2640, 2643, 2650, 2652, 2679, 2684, 2689, 2692, 2701, 2714, 2717, 2723, 2728, 2738, 4090, 4091
\tl_remove_all:Nn	4377
\tl_remove_once:Nn	2332, 2664
\tl_replace_all:Nnn	456
\tl_reverse:N	2331, 2333, 2663, 2665
\tl_set:Nn	56, 279, 286, 293, 421, 495, 499, 504, 505, 557, 602, 674, 822, 836, 848, 860, 1661, 1762, 2038, 2048, 2069, 2077, 2423, 2518, 2757, 2792, 2805, 2894, 4093, 4116, 4375
\tl_set_eq:NN	462, 563, 566, 610, 612, 627, 629, 680, 682, 2330, 2662, 2675, 2938, 2942, 3424, 3426
\tl_to_str:n	1632, 1638, 1643, 4225
\tl_trim_spaces:n	452, 4363, 4375, 4381
\tl_use:N	458, 461, 579, 644, 651, 694, 893, 897, 901, 905, 909, 913, 917, 921, 925, 929, 933, 937, 941, 945, 949, 953, 2196, 2337, 2345, 2356, 2370, 2375, 2387, 2881, 2887, 2911, 2929, 2933, 2941, 2978, 2979, 2986, 2994, 2995, 3001, 3116, 3283, 3429, 3615, 3812, 3823,



3827, 3974, 4156, 4167, 4173, 4178, 4279, 4280, 4281,  
4282, 4283, 4301, 4359

token commands:

\token\_to\_str:N . . . . . 405

\topsep . . . . . 2163, 2171

topsep . . . . . 747

\typeout . . . . . 373, 376, 386, 387

U

\u . . . . . 209

use commands:

\use:N . . . . . 216, 2983, 3118

\use:n . . . . . 1508, 2081, 4231

\use\_none:nn . . . . . 397

\usecounter . . . . . 3050, 3092

V

\value . . . . 1575, 1581, 1588, 1594, 1602, 1608, 1615, 1621

\vspace 358, 1407, 1410, 1421, 1424, 1434, 1436, 1445, 1447,  
1456, 1458, 1467, 1469, 1478, 1480, 1489, 1491, 2162,  
2170, 3440, 3451, 3863, 4218

W

widest . . . . . 727

wrap-ans . . . . . 1976

wrap-label . . . . . 465

wrap-label\* . . . . . 465

wrap-opt . . . . . 1976

©2024 by Pablo González L

135 / 135