

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	Introduction	1	5	The storage system	10
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for <code>label</code> and <code>ref</code>	11
1.3	User interface	3	5.1.2	Keys for <code>wrap</code> and <code>display</code>	11
1.3.1	Internal counters	3	5.1.3	Keys for <code>debug</code> and <code>checking</code>	11
1.3.2	Support for <code>multicol</code>	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>minipage</code>	3	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	13
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
2	The environments provided	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	16
3	The command <code>\setenumext</code>	6	6	Full examples	17
4	The <code>keyval</code> system	6	7	The way of non-enumerated lists	19
4.1	Keys for <code>label</code> and <code>ref</code>	6	8	References	21
4.2	Keys for spaces	7	9	Change history	21
4.2.1	Vertical spaces	7	10	Index of Documentation	22
4.2.2	Horizontal spaces	8	11	Implementation	24
4.3	Keys for <code>add code</code>	8	12	Index of Implementation	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 \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.

License and Requirements

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

1 Introduction

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

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

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

3. True False

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

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

* $(x - 1)^2$

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

* $3(x + y + z)$

3. True False

(a) $\alpha > \delta$

* False

(b) L^AT_EXze is cool?

* Very True!

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

* Yes

(b) Usually uses the package manager?

* Yes, dnf

(c) Rate the following package and class

i. xsim-exam

* doesn't exist for now :(

ii. xsim

* very good

iii. exsheets

* obsolete

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

* 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

E) value

4. Question with image and label below:

A

B

A

A) B) C)

A

duck

D) E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

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

2 / 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 \LaTeX , but “*does not redefine*” any internal commands associated with it such as `\list`, `\endlist` or `\item` outside of the “*scope*” in which they are defined.

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.
- This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex>dvips>ps2pdf` and is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

enumext.sty

>>

TDS:tex/latex/enumext/

enumext.pdf

>>

TDS:doc/latex/enumext/

README.md

>>

TDS:doc/latex/enumext/

enumext.dtx

>>

TDS:source/latex/enumext/

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

1.3 User interface

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

2.1 The environment enumext

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

Example with columns=2

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

2.2 The environment enumext*

The `enumext*` 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*

<code>\item*</code>	<code>\item*</code>
	<code>\item* [\langle symbol \rangle]</code>
	<code>\item* [\langle symbol \rangle] [\langle offset \rangle]</code>

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`

`\setenumext`

`\setenumext{ $\langle key = val \rangle$ }`

`\setenumext[$\langle enumext, level \rangle$]{ $\langle key = val \rangle$ }`

`\setenumext[$\langle enumext* \rangle$]{ $\langle key = val \rangle$ }`

`\setenumext[$\langle keyans \rangle$]{ $\langle key = val \rangle$ }`

`\setenumext[$\langle keyans* \rangle$]{ $\langle key = val \rangle$ }`

`\setenumext[$\langle print, level \rangle$]{ $\langle key = val \rangle$ }`

`\setenumext[$\langle print, * \rangle$]{ $\langle key = val \rangle$ }`

`\setenumext[$\langle print* \rangle$]{ $\langle key = val \rangle$ }`

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

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

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

4 The keyval system

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

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

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

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

4.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2 Keys for spaces

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

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

4.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 \LaTeX will be used.

By design the environment cannot be nested but full supports “*verbatim material*” in the body and it is assumed that each `\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.

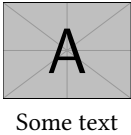
Example

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

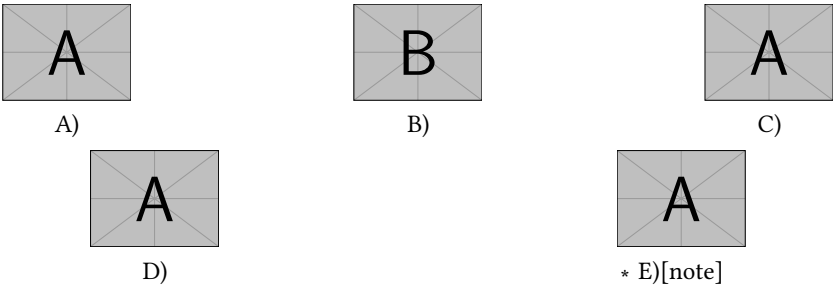
The `\anspic` command take three arguments, the *starred argument* `*` store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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 {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) ~~LaTeXe~~ 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 arara¹ tool to compile them.

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

D $3,60 \times 10^8$ km/h.
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 La velocità di $1,00 \times 10^2$ m/s espressa in km/h è: (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

2. A

3. B

4. A
- *

*

*

*

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

Example 3

A “first type of questions” test 📄.

- 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
3. Third type of questions
- (1) $2\alpha + 2\delta = 90^\circ$

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

 value
- B

 value
- C

 value
4. Question with image and label below:



A



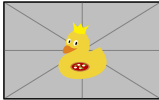
D



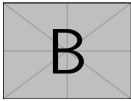
B



C



E



5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value

Test keys

1. B, $x = 5$
2. D
3. C, some note


- *


 4. E, A duck
- *

 5. D, other note
- *


Example 4

A “simple worksheet” using ducks :) 📄.


 Factor $x^2 - 2x + 1$

 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

 True False

- (a) $\alpha > \delta$
- (b) ~~La~~X₂e is cool?

 Related to Linux

- (a) You use linux?
- (b) Usually uses the package manager?
- (c) Rate the following package and class
- i.

`xsim-exam`
- ii.

`xsim`
- iii.

`exsheets`

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

1.

 $(x - 1)^2$
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

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.

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.

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.

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

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

Fake description environment

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

- SomeThing** A short one-line description.
This is an entry *without* a label.
- Something** A short *one-line* description text.
- Something long** A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
- If we add `list-indent=0pt` you get *widest style*:
- SomeThing** A short one-line description.
This is an entry *without* a label.
- Something** A short *one-line* description text.
- Something long** A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Description indented by label

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

Something 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,

LoNg vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

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

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

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

Why list environments?

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

Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

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

8 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The \LaTeX Project. “The 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, 20
<code>\alph*</code>	7	<code>l3seq</code>	1, 20
<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>	20
<code>enumext</code>	1–6, 15, 20
<code>enumitem</code>	3–5, 9, 20, 21
<code>footnotehyper</code>	4, 5
<code>hyperref</code>	4, 5, 11–13, 20, 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 \TeX pert are superfluous, but, after so many years developing this project is the only way to remember what does what.

11.1 General conventions

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

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

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

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

11.2 Initial set up

Start the DocStrip guards.

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

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

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

11.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-06-07}
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 (§11.23).

```
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 (§11.16). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.34.1).

```

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 §11.20).

```

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 (§11.21.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.19, §11.21).

```

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 (§11.16.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.11). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§11.18).

```

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 (§11.24.1) 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` (§11.27) 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*` (§11.28).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§11.32.2) and `\anspic*` (§11.37.1) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```
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` (§11.43).

```
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` (§11.42), `show-pos` key (§11.26), `item-sym*` key (§11.30), `save-key` key (§11.26.2) 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 (§11.37.2).

```
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 (§11.24.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
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 (§11.26). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the `⟨labels⟩` defined by level on which operations will be performed.

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

```
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` (§11.31).

```
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
\l__enumext_miniright_code_X_box
\g__enumext_minipage_center_X_bool
\g__enumext_minipage_right_X_dim
\g__enumext_minipage_right_X_skip
```

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

```
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   \box_new:c { l__enumext_miniright_code_#1_box }
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-right` and `mini-right*` 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:Npnn \__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^AT_EX has the following plain form:

```

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

```

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

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w

```

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

```

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^AT_EX has the following (simplified) plain form:

```

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

```

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

```

\__enumext_minipage:w
\__enumext_endminipage:

```

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

```

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 (§11.35) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.39)

```

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 `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “label and ref” system.

11.12.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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 #1: \__enumext_label_X_tl
\__enumext_start_from:ccn #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.)

```

start
widest
\l__enumext_start_X_int

```

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

```

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 base-fix key

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

```

base-fix
\__enumext_nested_base_line_fix:

```

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

```

791 \cs_set_protected:Npn \__enumext_tmp:n #1
792 {
793   \keys_define:nn { enumext / #1 }
794   {
795     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
796     base-fix .initial:n = true,
797     base-fix .value_forbidden:n = true,
798   }
799 }
800 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

```

801 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
802 {
803   \bool_lazy_and:nnT

```

```

804 { \bool_if_p:N \__enumext_standar_first_bool }
805 { \bool_if_p:N \__enumext_base_line_fix_bool }
806 {
807   \mode_leave_vertical:
808   \vspace { -\baselineskip }
809   \keys_set:nn { enumext / level-1 }
810   {
811     topsep = 0pt, above = 0pt, above* = 0pt,
812   }
813 }
814 \bool_lazy_and:nnT
815 { \bool_if_p:N \__enumext_starred_first_bool }
816 { \bool_if_p:N \__enumext_base_line_fix_bool }
817 {
818   \mode_leave_vertical:
819   \vspace { -\baselineskip }
820   \keys_set:nn { enumext / enumext* }
821   {
822     topsep = 0pt, above = 0pt, above* = 0pt,
823   }
824 }
825 \bool_set_false:N \__enumext_base_line_fix_bool
826 }

```

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

11.16 Setting keys for horizontal spaces

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

```

827 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
828 {
829   \keys_define:nn { enumext / #1 }
830   {
831     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
832     itemindent .value_required:n = true,
833     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
834     rightmargin .value_required:n = true,
835     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
836     listparindent .value_required:n = true,
837     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
838     list-offset .value_required:n = true,
839     list-indent .code:n =
840       \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
841       \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {##1},
842     list-indent .value_required:n = true,
843   }
844 }
845 \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.

```

846 \cs_set_protected:Npn \__enumext_tmp:n #1
847 {
848   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
849 }
850 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

11.16.1 Functions for setting the fake `itemindent`

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

```

851 \cs_set_protected:Nn \__enumext_fake_item:
852 {
853   \dim_compare:nNnT
854     { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
855     >
856     { \c_zero_dim }
857   {

```

```

858         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
859         {
860             \exp_not:N \mode_leave_vertical:
861             \exp_not:n { \skip_horizontal:n }
862             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
863             \ignorespaces
864         }
865     }
866 }
867 \cs_set_protected:Nn \__enumext_keyans_fake_item:
868 {
869     \dim_compare:nNnT
870     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
871     {
872         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
873         {
874             \exp_not:N \mode_leave_vertical:
875             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
876         }
877     }
878 }
879 \cs_set_protected:Nn \__enumext_fake_item_vii:
880 {
881     \dim_compare:nNnT
882     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
883     {
884         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
885         {
886             \exp_not:N \mode_leave_vertical:
887             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
888         }
889     }
890 }
891 \cs_set_protected:Nn \__enumext_fake_item_viii:
892 {
893     \dim_compare:nNnT
894     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
895     {
896         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
897         {
898             \exp_not:N \mode_leave_vertical:
899             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
900         }
901     }
902 }

```

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

11.17 Setting show-length key

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

```

903 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
904 {
905     \keys_define:nn { enumext / #1 }
906     {
907         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
908         show-length .initial:n = false,
909     }
910 }
911 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.18 Setting before, after and first keys

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

```

before
before*
after
first
912 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
913 {

```

```

914     \keys_define:nn { enumext / #1 }
915     {
916         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
917         before .value_required:n = true,
918         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
919         before* .value_required:n = true,
920         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
921         after .value_required:n = true,
922         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
923         first .value_required:n = true,
924     }
925 }
926 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.18.1 Functions for before, after and first keys in enumext

`__enumext_before_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.

```

927 \cs_new_protected:Nn \__enumext_before_args_exec:
928 {
929     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
930 }

```

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 $\langle keys \rangle$.

```

931 \cs_new_protected:Nn \__enumext_before_keys_exec:
932 {
933     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
934 }

```

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

```

935 \cs_new_protected:Nn \__enumext_after_stop_list:
936 {
937     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
938 }

```

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*.

```

939 \cs_new_protected:Nn \__enumext_after_args_exec:
940 {
941     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
942 }

```

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

11.18.2 Functions for before, after and first keys in keyans

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

```

943 \cs_new_protected:Nn \__enumext_before_args_exec_v:
944 {
945     \tl_use:N \l__enumext_before_starred_key_v_tl
946 }

```

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

```

947 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
948 {
949     \tl_use:N \l__enumext_before_no_starred_key_v_tl
950 }

```

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

```

951 \cs_new_protected:Nn \__enumext_after_stop_list_v:
952 {
953     \tl_use:N \l__enumext_after_stop_list_v_tl
954 }

```


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`.

```

955 \cs_new_protected:Nn \__enumext_after_args_exec_v:
956 {
957     \tl_use:N \l__enumext_after_list_args_v_tl
958 }

```

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

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

```

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

```

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

```

959 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
960 {
961     \tl_use:N \l__enumext_before_starred_key_vii_tl
962 }
963 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
964 {
965     \tl_use:N \l__enumext_before_starred_key_viii_tl
966 }

```

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⟩`.

```

967 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
968 {
969     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
970 }
971 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
972 {
973     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
974 }

```

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

```

975 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
976 {
977     \tl_use:N \l__enumext_after_stop_list_vii_tl
978 }
979 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
980 {
981     \tl_use:N \l__enumext_after_stop_list_viii_tl
982 }

```

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`.

```

983 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
984 {
985     \tl_use:N \l__enumext_after_list_args_vii_tl
986 }
987 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
988 {
989     \tl_use:N \l__enumext_after_list_args_viii_tl
990 }

```

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

11.19 Setting keys for multicols and minipage

```

mini-env
mini-sep
columns-sep
columns

```

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

```

991 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
992 {
993     \keys_define:nn { enumext / #1 }
994     {
995         mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
996         mini-env .value_required:n = true,
997         mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
998         mini-sep .initial:n = 0.3333em,

```

```

999      mini-sep      .value_required:n = true,
1000      columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1001      columns-sep .value_required:n = true,
1002      columns      .int_set:c  = { l__enumext_columns_#2_int },
1003      columns      .initial:n  = 1,
1004      columns      .value_required:n = true,
1005    }
1006  }
1007  \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.

```

1008  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1009  {
1010    \keys_define:nn { enumext / #1 }
1011    {
1012      mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1013      mini-right .value_required:n = true,
1014      mini-right* .code:n    = {
1015        \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1016        \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1017      },
1018      mini-right* .value_required:n = true,
1019    }
1020  }
1021  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.20 Adjustment of vertical spaces for multicol

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



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

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

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

11.20.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

1022  \cs_new_protected:Npn \__enumext_multi_set_vskip:
1023  {
1024    \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1025    {
1026      \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1027    }
1028    \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1029    {
1030      \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1031    }
1032    \__enumext_add_pre_parsep:
1033  }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

1034 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1035 {
1036   \int_case:nn { \l__enumext_level_int }
1037   {
1038     { 2 }{
1039       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1040       {
1041         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1042       }
1043     }
1044     { 3 }{
1045       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1046       {
1047         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1048       }
1049     }
1050     { 4 }{
1051       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1052       {
1053         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1054       }
1055     }
1056   }
1057 }

```

(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*.

```

1058 \cs_new_protected:Nn \__enumext_multi_addvspace:
1059 {
1060   \__enumext_multi_set_vskip:
1061   \mode_if_vertical:T
1062   {
1063     \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1064     {
1065       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1066     }
1067     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1068     {
1069       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1070     }
1071   }
1072   \par\nopagebreak
1073   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1074 }

```

(End of definition for `__enumext_multi_addvspace:`)

11.20.2 Adjustment of vertical spaces for multicols in keyans

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

```

1075 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1076 {
1077   \skip_set:Nn \l__enumext_multicols_above_v_skip
1078   {
1079     \l__enumext_topsep_v_skip
1080   }
1081   \skip_set:Nn \l__enumext_multicols_below_v_skip
1082   {
1083     \l__enumext_topsep_v_skip
1084   }
1085 }

```

```

1086 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1087 {
1088   \__enumext_keyans_multi_set_vskip:
1089   \mode_if_vertical:T
1090   {
1091     \skip_add:Nn \l__enumext_multicols_above_v_skip
1092     {
1093       \skip_use:N \l__enumext_partopsep_v_skip
1094     }
1095     \skip_add:Nn \l__enumext_multicols_below_v_skip
1096     {
1097       \skip_use:N \l__enumext_partopsep_v_skip
1098     }
1099   }
1100   \par\nopagebreak
1101   \addvspace{ \l__enumext_multicols_above_v_skip }
1102 }

```

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

11.21 Adjustment of vertical spaces for minipage

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

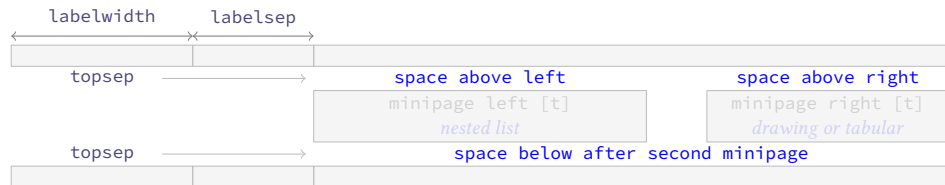


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

Here there are several complications that we must circumvent, the `minipage` environment eliminates the “top” spaces, the `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.

11.21.1 Adjustment of vertical spaces for minipage in enumext

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

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

First determine if the `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`.

```

1103 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1104 {
1105   \int_compare:nNtF
1106   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1107   {

```

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

```

1108   \skip_if_eq:nnTF
1109   { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1110   {
1111     \skip_set:Nn \l__enumext_minipage_left_skip
1112     {
1113       -0.150\box_dp:N \strutbox

```

```

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

```

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.

```

1142     \skip_if_eq:nnTF
1143     { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1144     {
1145         \skip_set:Nn \l__enumext_minipage_left_skip
1146         {
1147             0.5\box_dp:N \strutbox
1148             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1149         }
1150         \skip_set:Nn \l__enumext_minipage_right_skip
1151         {
1152             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1153         }
1154         \skip_set:Nn \l__enumext_minipage_after_skip
1155         {
1156             1.6\box_dp:N \strutbox
1157         }
1158     }
1159     {
1160         \skip_set:Nn \l__enumext_minipage_left_skip
1161         {
1162             0.5875\box_dp:N \strutbox
1163             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1164         }
1165         \skip_set:Nn \l__enumext_minipage_right_skip
1166         {
1167             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1168             + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1169         }
1170         \skip_set:Nn \l__enumext_minipage_after_skip
1171         {
1172             0.325\box_dp:N \strutbox
1173             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1174         }
1175     }
1176 }
1177 }

```

(End of definition for `__enumext_mini_set_vskip:`)

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

```

1178 \cs_new_protected:Nn \__enumext_zero_parsep:
1179 {
1180   \int_case:nn { \l__enumext_level_int }
1181   {
1182     { 2 }{
1183       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1184       {
1185         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1186       }
1187     }
1188     { 3 }{
1189       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1190       {
1191         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1192       }
1193     }
1194     { 4 }{
1195       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1196       {
1197         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1198       }
1199     }
1200   }
1201 }

```

(End of definition for `__enumext_zero_parsep:`.)

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

```

1202 \cs_new_protected:Nn \__enumext_mini_addvspace:
1203 {
1204   \__enumext_mini_set_vskip:
1205   \mode_if_vertical:T
1206   {
1207     \skip_add:Nn \l__enumext_minipage_left_skip
1208     {
1209       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1210     }
1211     \skip_add:Nn \l__enumext_minipage_after_skip
1212     {
1213       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1214     }
1215   }
1216   \par\nopagebreak
1217   \addvspace { \l__enumext_minipage_left_skip }
1218 }

```

(End of definition for `__enumext_mini_addvspace:`.)

11.21.2 Adjustment of vertical spaces for minipage in keyans

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

```

1219 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1220 {
1221   \skip_zero_new:N \l__enumext_minipage_after_skip
1222   \skip_zero_new:N \l__enumext_minipage_left_skip
1223   \skip_zero_new:N \l__enumext_minipage_right_skip
1224   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1225   {
1226     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1227     {
1228       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1229       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1230       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }

```



```

1231     \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1232     {
1233       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1234     }
1235   }
1236   {
1237     \skip_set:Nn \l__enumext_minipage_left_skip
1238     {
1239       \skip_use:N \l__enumext_topsep_v_skip
1240     }
1241     \skip_set:Nn \l__enumext_minipage_right_skip
1242     {
1243       0.705\box_dp:N \strutbox
1244     }
1245     \skip_set:Nn \l__enumext_minipage_after_skip
1246     {
1247       1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1248     }
1249   }
1250 }
1251 {
1252   \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1253   {
1254     \skip_set:Nn \l__enumext_minipage_left_skip
1255     {
1256       0.5\box_dp:N \strutbox
1257       + \l__enumext_partopsep_v_skip
1258     }
1259     \skip_set:Nn \l__enumext_minipage_right_skip
1260     {
1261       \l__enumext_partopsep_v_skip
1262     }
1263     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1264   }
1265   {
1266     \skip_set:Nn \l__enumext_minipage_left_skip
1267     {
1268       0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1269     }
1270     \skip_set:Nn \l__enumext_minipage_right_skip
1271     {
1272       \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1273     }
1274     \skip_set:Nn \l__enumext_minipage_after_skip
1275     {
1276       0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1277     }
1278   }
1279 }
1280 }

```

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

`\l__enumext_keyans_mini_addvspace:`

The function `\l__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\l__enumext_mini_env*` environment in `keyans`, taking into account whether \TeX is in $\langle horizontal mode \rangle$ or $\langle vertical mode \rangle$. 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`.

```

1281 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1282 {
1283   \l__enumext_keyans_mini_set_vskip:
1284   \mode_if_vertical:T
1285   {
1286     \skip_add:Nn \l__enumext_minipage_left_skip
1287     {
1288       \l__enumext_partopsep_v_skip
1289     }
1290     \skip_add:Nn \l__enumext_minipage_after_skip
1291     {
1292       \l__enumext_partopsep_v_skip
1293     }

```

```

1294     }
1295     \par\nopagebreak
1296     \addvspace { \l__enumext_minipage_left_skip }
1297 }

```

(End of definition for __enumext_keyans_mini_addvspace:.)

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1298 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1299 {
1300     \skip_zero_new:N \l__enumext_minipage_left_skip
1301     \skip_gzero_new:N \g__enumext_minipage_right_skip
1302     \skip_gzero_new:N \g__enumext_minipage_after_skip
1303     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1304     {
1305         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1306         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1307     }
1308     {
1309         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1310         \skip_gset:Nn \g__enumext_minipage_right_skip
1311         {
1312             \l__enumext_topsep_vii_skip
1313         }
1314         \skip_gset:Nn \g__enumext_minipage_after_skip
1315         {
1316             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1317         }
1318     }
1319 }
1320 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1321 {
1322     \skip_zero_new:N \l__enumext_minipage_after_skip
1323     \skip_zero_new:N \l__enumext_minipage_left_skip
1324     \skip_zero_new:N \l__enumext_minipage_right_skip
1325     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1326     {
1327         \skip_set:Nn \l__enumext_minipage_left_skip
1328         {
1329             0.5\box_dp:N \strutbox
1330         }
1331         \skip_set:Nn \l__enumext_minipage_right_skip
1332         {
1333             \l__enumext_partopsep_viii_skip
1334         }
1335         \skip_set:Nn \l__enumext_minipage_after_skip
1336         {
1337             1.6\box_dp:N \strutbox
1338         }
1339     }
1340     {
1341         \skip_set:Nn \l__enumext_minipage_left_skip
1342         {
1343             0.5875\box_dp:N \strutbox
1344         }
1345         \skip_set:Nn \l__enumext_minipage_right_skip
1346         {
1347             \l__enumext_topsep_viii_skip
1348         }
1349         \skip_set:Nn \l__enumext_minipage_after_skip
1350         {
1351             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1352         }
1353     }
1354 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

```

1355 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1356 {
1357   \__enumext_mini_set_vskip_vii:
1358   \par\nopagebreak
1359   \addvspace { \l__enumext_minipage_left_skip }
1360 }
1361 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1362 {
1363   \__enumext_mini_set_vskip_viii:
1364   \par\nopagebreak
1365   \addvspace { \l__enumext_minipage_left_skip }
1366 }

```

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

11.21.4 The command `\miniright`

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

`\miniright`

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

```

1367 \NewDocumentCommand \miniright { s }
1368 {
1369   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1370   {
1371     \msg_error:nnn { enumext } { wrong-miniright-place }
1372   }
1373   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1374   {
1375     \msg_error:nnn { enumext } { wrong-miniright-place }
1376   }
1377   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1378   {
1379     \__enumext_keyans_mini_right_cmd:n {#1}
1380   }
1381   { \__enumext_mini_right_cmd:n {#1} }
1382 }

```

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

`__enumext_mini_right_cmd:n`

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

```

1383 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1384 {
1385   \dim_compare:nNnTF
1386   { \dim_use:c { \l__enumext_minipage_right_ } \__enumext_level: _dim } > { \c_zero_dim }
1387   {
1388     \__enumext_multicols_stop:
1389     \end{__enumext_mini_env*}
1390     \hfill
1391     \begin{__enumext_mini_env*}
1392     { \dim_use:c { \l__enumext_minipage_right_ } \__enumext_level: _dim } }
1393     \par\addvspace { \l__enumext_minipage_right_skip }
1394     \bool_if:nF {#1}
1395     {

```

```

1396         \centering
1397     }
1398     \int_gzero:N \g__enumext_minipage_stat_int
1399 }
1400 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1401 }

```

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

`__enumext_keyans_mini_right_cmd:n`

The function `__enumext_keyans_mini_right_cmd:n` takes as argument the *starred* ‘`*`’ of the `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1402 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1403 {
1404     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1405     {
1406         \__enumext_keyans_multicols_stop:
1407         \end{__enumext_mini_env*}
1408         \hfill
1409         \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1410         \par\addvspace { \l__enumext_minipage_right_skip }
1411         \bool_if:nF {#1}
1412         {
1413             \centering
1414         }
1415         \int_gzero:N \g__enumext_minipage_stat_int
1416     }
1417     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1418 }

```

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

11.22 Setting above and below keys

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

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

```

above* 1419 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1420 {
below* 1421 \keys_define:nn { enumext / #1 }
1422 {
1423     above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1424     above .value_required:n = true,
1425     above* .code:n = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1426                 \keys_set:nn { enumext / #1 } { above = {##1} },
1427     above* .value_required:n = true,
1428     below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1429     below .value_required:n = true,
1430     below* .code:n = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1431                 \keys_set:nn { enumext / #1 } { below = {##1} },
1432     below* .value_required:n = true,
1433 }
1434 }
1435 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.22.1 Functions for above and below keys in enumext

`__enumext_vspace_above:`

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

```

1436 \cs_new_protected:Nn \__enumext_vspace_above:
1437 {
1438     \skip_if_eq:nnF
1439     { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1440     {
1441         \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1442         {

```

```

1443         \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1444     }
1445     {
1446         \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1447     }
1448 }
1449 }

```

(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.

```

1450 \cs_new_protected:Nn \__enumext_vspace_below:
1451 {
1452     \skip_if_eq:nnF
1453     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1454     {
1455         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1456         {
1457             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1458         }
1459         {
1460             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1461         }
1462     }
1463 }

```

(End of definition for __enumext_vspace_below:.)

11.22.2 Functions for above and below keys in keyans

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

```

1464 \cs_new_protected:Nn \__enumext_vspace_above_v:
1465 {
1466     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1467     {
1468         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1469         {
1470             \vspace*{ \l__enumext_vspace_above_v_skip }
1471         }
1472         { \vspace { \l__enumext_vspace_above_v_skip } }
1473     }
1474 }

```

(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.

```

1475 \cs_new_protected:Nn \__enumext_vspace_below_v:
1476 {
1477     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1478     {
1479         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1480         {
1481             \vspace*{ \l__enumext_vspace_below_v_skip }
1482         }
1483         { \vspace { \l__enumext_vspace_below_v_skip } }
1484     }
1485 }

```

(End of definition for __enumext_vspace_below_v:.)

11.22.3 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:

```

1486 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1487 {
1488     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1489     {
1490         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1491         {

```

```

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

```

(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:`

```

1508 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1509 {
1510     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1511     {
1512         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1513         {
1514             \vspace*{ \l__enumext_vspace_below_vii_skip }
1515         }
1516         { \vspace { \l__enumext_vspace_below_vii_skip } }
1517     }
1518 }
1519 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1520 {
1521     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1522     {
1523         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1524         {
1525             \vspace*{ \l__enumext_vspace_below_viii_skip }
1526         }
1527         { \vspace { \l__enumext_vspace_below_viii_skip } }
1528     }
1529 }

```

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

11.23 Setting series, resume and resume* keys

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

`series`
`resume`
`resume*`

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

```

1530 \cs_set_protected:Npn \__enumext_tmp:n #1
1531 {
1532     \keys_define:nn { enumext / #1 }
1533     {
1534         series .str_set:N = \l__enumext_series_str,
1535         series .value_required:n = true,
1536         resume .code:n = \__enumext_resume_series:n {##1},
1537         resume* .code:n = \__enumext_resume_starred:,
1538         resume* .value_forbidden:n = true,
1539     }
1540 }
1541 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.23.1 Internal functions for series key

```

__enumext_filter_series:n
  __enumext_filter_series_key:n
  __enumext_filter_series_pair:nn

```

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

```

1542 \cs_new:Npn __enumext_filter_series:n #1
1543 {
1544   \use:e
1545   {
1546     \keyval_parse:NNn
1547     __enumext_filter_series_key:n
1548     __enumext_filter_series_pair:nn {#1}
1549   }
1550 }

```

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.

```

1551 \cs_new:Npn __enumext_filter_series_key:n #1
1552 {
1553   \str_case:nnF {#1}
1554   {
1555     { resume } {}
1556     { resume* } {}
1557   }
1558   { , { \exp_not:n {#1} } }
1559 }

```

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.

```

1560 \cs_new:Npn __enumext_filter_series_pair:nn #1#2
1561 {
1562   \str_case:nnF {#1}
1563   {
1564     { series } {}
1565     { resume } {}
1566     { start } {}
1567     { save-ans } {}
1568     { save-key } {}
1569   }
1570   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1571 }

```

(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 (§11.35) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39).

```

1572 \cs_new_protected:Npn __enumext_parse_series:n #1
1573 {
1574   \str_if_empty:NTF \l__enumext_series_str
1575   {
1576     \bool_if:NF \l__enumext_resume_active_bool
1577     {
1578       __enumext_resume_last:n {#1}
1579     }
1580   }
1581   {
1582     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1583     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1584     { __enumext_filter_series:n {#1} }
1585     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1586     {
1587       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1588     }
1589   }

```

```
1590 }
```

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.

```
1591 \cs_new_protected:Npn \__enumext_resume_last:n #1
1592 {
1593   \bool_if:NT \l__enumext_standar_first_bool
1594   {
1595     \tl_gclear:N \g__enumext_standar_series_tl
1596     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1597   }
1598   \bool_if:NT \l__enumext_starred_first_bool
1599   {
1600     \tl_gclear:N \g__enumext_starred_series_tl
1601     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1602   }
1603 }
```

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

11.23.2 Internal function to save counter value

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

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

```
1604 \cs_new_protected:Nn \__enumext_resume_save_counter:
1605 {
1606   \bool_if:NT \g__enumext_standar_bool
1607   {
1608     \tl_if_empty:NF \l__enumext_series_str
1609     {
1610       \int_gset_eq:cN
1611       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1612     }
1613     \tl_if_empty:NTF \l__enumext_resume_name_tl
1614     {
1615       \str_if_empty:NT \l__enumext_series_str
1616       {
1617         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1618       }
1619     }
1620     {
1621       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1622       {
1623         \int_gset_eq:cN
1624         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1625       }
1626     }
1627     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1628     {
1629       \int_gset_eq:cN
1630       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1631     }
1632   }
1633   \bool_if:NT \g__enumext_starred_bool
1634   {
1635     \tl_if_empty:NF \l__enumext_series_str
1636     {
1637       \int_gset_eq:cN
1638       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1639     }
1640   }
```

```

1640     \tl_if_empty:NTF \l__enumext_resume_name_tl
1641     {
1642         \str_if_empty:NT \l__enumext_series_str
1643         {
1644             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1645         }
1646     }
1647     {
1648         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1649         {
1650             \int_gset_eq:cN
1651             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1652         }
1653     }
1654     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1655     {
1656         \int_gset_eq:cN
1657         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1658     }
1659 }
1660 }

```

(End of definition for `__enumext_resume_save_counter:`.)

11.23.3 Internal functions for resume key

`__enumext_resume_series:n`

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

```

1661 \cs_new_protected:Npn \__enumext_resume_series:n #1
1662 {
1663     \tl_if_empty:NTF {#1}
1664     {
1665         \__enumext_resume_counter:n { }
1666     }
1667     {
1668         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1669         {
1670             \__enumext_resume_counter:n {#1}
1671             \bool_if:NT \g__enumext_standar_bool
1672             {
1673                 \keys_set:nv { enumext / level-1 }
1674                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1675             }
1676             \bool_if:NT \g__enumext_starred_bool
1677             {
1678                 \keys_set:nv { enumext / enumext* }
1679                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1680             }
1681         }
1682         {
1683             \bool_if:NT \g__enumext_standar_bool
1684             {
1685                 \msg_error:nnn { enumext } { unknown-series } {#1}
1686             }
1687             \bool_if:NT \g__enumext_starred_bool
1688             {
1689                 \msg_error:nnn { enumext } { unknown-series } {#1}
1690             }
1691         }
1692     }
1693 }

```

(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`.

```

1694 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1695 {
1696   \bool_set_true:N \l__enumext_resume_active_bool
1697   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1698   \tl_if_empty:NTF \l__enumext_resume_name_tl
1699   {
1700     \__enumext_resume_counter:
1701   }
1702   {
1703     \__enumext_resume_counter_series:
1704   }
1705   \__enumext_resume_counter_save_ans:
1706 }

```

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.

```

1707 \cs_new_protected:Nn \__enumext_resume_counter:
1708 {
1709   \bool_if:NT \g__enumext_standar_bool
1710   {
1711     \int_gincr:N \g__enumext_resume_int
1712     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1713   }
1714   \bool_if:NT \g__enumext_starred_bool
1715   {
1716     \int_gincr:N \g__enumext_resume_vii_int
1717     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1718   }
1719 }

```

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.

```

1720 \cs_new_protected:Nn \__enumext_resume_counter_series:
1721 {
1722   \bool_if:NT \g__enumext_standar_bool
1723   {
1724     \int_set:Nn \l__enumext_start_i_int
1725     {
1726       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1727     }
1728   }
1729   \bool_if:NT \g__enumext_starred_bool
1730   {
1731     \int_set:Nn \l__enumext_start_vii_int
1732     {
1733       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1734     }
1735   }
1736 }

```

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.

```

1737 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1738 {
1739   \bool_lazy_and:nnT
1740   { \bool_if_p:N \l__enumext_standar_first_bool }
1741   { \bool_if_p:N \l__enumext_store_active_bool }
1742   {
1743     \int_set:Nn \l__enumext_start_i_int
1744     {
1745       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1746     }
1747   }
1748   \bool_lazy_and:nnT
1749   { \bool_if_p:N \l__enumext_starred_first_bool }

```

```

1750     { \bool_if_p:N \__enumext_store_active_bool }
1751     {
1752         \int_set:Nn \__enumext_start_vii_int
1753         {
1754             \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1755         }
1756     }
1757 }

```

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

11.23.4 Internal function for `resume*` key

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

```

1758 \cs_new_protected:Nn \__enumext_resume_starred:
1759 {
1760     \bool_if:NT \g__enumext_standar_bool
1761     {
1762         \tl_if_empty:NF \g__enumext_standar_series_tl
1763         {
1764             \__enumext_resume_counter:n { }
1765             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1766         }
1767     }
1768     \bool_if:NT \g__enumext_starred_bool
1769     {
1770         \tl_if_empty:NF \g__enumext_starred_series_tl
1771         {
1772             \__enumext_resume_counter:n { }
1773             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1774         }
1775     }
1776 }

```

(End of definition for `__enumext_resume_starred:`.)

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

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

11.24.1 Setting save-ans key

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

```

1777 \cs_set_protected:Npn \__enumext_tmp:n #1
1778 {
1779     \keys_define:nn { enumext / #1 }
1780     {
1781         save-ans .code:n = \__enumext_storing_set:n {##1},
1782         save-ans .value_required:n = true,
1783     }
1784 }
1785 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.24.2 Internal functions for `save-ans` key

`__enumext_start_save_ans_msg:` 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:`.

```

1786 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1787 {
1788     \msg_term:nnVV { enumext } { save-ans-log }
1789     \g__enumext_envir_name_tl \__enumext_store_name_tl
1790 }
1791 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1792 {
1793     \msg_term:nnVV { enumext } { save-ans-log-hook }

```

```

1794     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1795 }

```

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

```

\__enumext_storing_set:n
\__enumext_storing_exec:

```

The function `__enumext_storing_set:n` first pass the value of the `save-ans` key to the variable `\l__enumext_store_name_tl` which will contain the “store name” of the *⟨sequence⟩* and *⟨prop list⟩* we will use. If `\l__enumext_store_name_tl` is *empty* we return an error message, otherwise will return the appropriate message `__enumext_start_save_ans_msg:` and proceed to execute the function `__enumext_storing_exec:` for `enumext` and `enumext*` environments.

```

1796 \cs_new_protected:Npn \__enumext_storing_set:n #1
1797 {
1798   \tl_set:Nx \l__enumext_store_name_tl {#1}
1799   \tl_if_empty:NTF \l__enumext_store_name_tl
1800   {
1801     \bool_lazy_or:nnT
1802     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1803     {
1804       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1805     }
1806   }
1807   {
1808     \bool_lazy_or:nnT
1809     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1810     {
1811       \__enumext_start_save_ans_msg:
1812       \__enumext_storing_exec:
1813     }
1814   }
1815 }

```

The function `__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy *⟨{store name}⟩* into the global variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§11.28). The *⟨prop list⟩* `\g__enumext_series_⟨store name⟩_prop` and the *⟨sequence⟩* `\g__enumext_series_⟨store name⟩_seq` will be created globally to “store content” in case they do not exist together with the integer variable `\g__enumext_series_⟨store name⟩_int` used by the keys `resume` and `resume*`.

```

1816 \cs_new_protected:Nn \__enumext_storing_exec:
1817 {
1818   \bool_set_true:N \l__enumext_store_active_bool
1819   \bool_set_true:N \l__enumext_check_answers_bool
1820   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1821   \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1822   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1823   {
1824     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1825     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1826   }
1827   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1828   {
1829     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1830     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1831   }
1832   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1833   {
1834     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1835     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1836   }
1837 }

```

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

11.24.3 The check answer mechanism

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

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

executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.

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

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

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

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

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

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

11.24.4 Setting check-ans and no-store keys

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

```
check-ans 1838 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1839 {
1840   \keys_define:nn { enumext / #1 }
1841   {
1842     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1843     check-ans .initial:n = false,
1844     check-ans .value_required:n = true,
1845     no-store .code:n = {
1846       \bool_set_false:N \l__enumext_check_answers_bool
1847       \bool_set_false:N \l__enumext_check_ans_key_bool
1848     },
1849     no-store .value_forbidden:n = true,
1850   }
1851 }
1852 \clist_map_inline:nn
1853 {
1854   level-1, level-2, level-3, level-4, enumext*
1855 }
1856 { \__enumext_tmp:n {#1} }
```

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

11.24.5 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.

```
1857 \cs_new_protected:Nn \__enumext_check_ans_active:
1858 {
1859   \tl_if_empty:NF \l__enumext_store_name_tl
1860   {
1861     \bool_if:NT \l__enumext_check_answers_bool
1862     {
1863       \__enumext_check_ans_level:
1864     }
1865   }
1866 }
```

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.

```

1867 \cs_new_protected:Nn \__enumext_check_ans_level:
1868 {
1869   \int_case:nn { \l__enumext_level_int }
1870   {
1871     { 1 }{
1872       \bool_lazy_all:nT
1873       {
1874         { \bool_if_p:N \g__enumext_starred_bool }
1875         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1876       }
1877       {
1878         \int_gdecr:N \g__enumext_item_number_int
1879       }
1880     }
1881     { 2 }{
1882       \int_gdecr:N \g__enumext_item_number_int
1883     }
1884     { 3 }{
1885       \int_gdecr:N \g__enumext_item_number_int
1886     }
1887     { 4 }{
1888       \int_gdecr:N \g__enumext_item_number_int
1889     }
1890   }

```

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.

```

1891   \int_case:nn { \l__enumext_level_h_int }
1892   {
1893     { 1 }{
1894       \bool_lazy_all:nT
1895       {
1896         { \bool_if_p:N \g__enumext_standar_bool }
1897         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1898       }
1899       {
1900         \int_gdecr:N \g__enumext_item_number_int
1901       }
1902     }
1903   }
1904 }

```

(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.

```

1905 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1906 {
1907   \bool_lazy_and:nnT
1908   { \bool_if_p:N \l__enumext_check_ans_key_bool }
1909   { \bool_if_p:N \g__enumext_standar_bool }
1910   {
1911     \bool_gset_true:N \g__enumext_check_ans_key_bool
1912   }
1913   \bool_lazy_and:nnT
1914   { \bool_if_p:N \l__enumext_check_ans_key_bool }
1915   { \bool_if_p:N \g__enumext_starred_bool }
1916   {
1917     \bool_gset_true:N \g__enumext_check_ans_key_bool
1918   }
1919 }

```

(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:`.

```

1920 \cs_new_protected:Nn \__enumext_item_answer_diff:
1921 {
1922   \int_gset:Nn \g__enumext_item_answer_diff_int
1923   {
1924     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1925   }
1926 }

```

(End of definition for __enumext_item_answer_diff:.)

```

\__enumext_check_ans_show:
  \__enumext_check_ans_msg_less:
  \__enumext_check_ans_msg_same_ok:
  \__enumext_check_ans_msg_greater:

```

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

```

1927 \cs_new_protected:Nn \__enumext_check_ans_show:
1928 {
1929   \int_case:nn { \g__enumext_item_answer_diff_int }
1930   {
1931     { -1 } { \__enumext_check_ans_msg_less: }
1932     { 0 } { \__enumext_check_ans_msg_same_ok: }
1933     { 1 } { \__enumext_check_ans_msg_greater: }
1934   }
1935 }
1936 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1937 {
1938   \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1939   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1940 }
1941 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1942 {
1943   \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1944   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1945 }
1946 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1947 {
1948   \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1949   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1950 }

```

(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:.

```

1951 \cs_new_protected:Nn \__enumext_check_ans_log:
1952 {
1953   \int_case:nn { \g__enumext_item_answer_diff_int }
1954   {
1955     { -1 } { \__enumext_check_ans_log_msg_less: }
1956     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1957     { 1 } { \__enumext_check_ans_log_msg_greater: }
1958   }
1959 }
1960 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1961 {
1962   \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1963   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1964 }
1965 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1966 {
1967   \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1968   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1969 }
1970 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1971 {
1972   \msg_log:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1973   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1974 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.24.6 Check for `\item*` and `\anspic*` commands

`__enumext_check_starred_cmd:n`

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

```

1975 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1976 {
1977   \int_compare:nNnT
1978     { \g__enumext_check_starred_cmd_int } = { 0 }
1979     {
1980       \msg_warning:nnnV
1981         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1982     }
1983   \int_compare:nNnT
1984     { \g__enumext_check_starred_cmd_int } > { 1 }
1985     {
1986       \msg_warning:nnnV
1987         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1988     }
1989   \int_gzero:N \g__enumext_check_starred_cmd_int
1990   \tl_clear:N \l__enumext_check_start_line_env_tl
1991 }

```

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

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

`__enumext_execute_after_env:`

The `__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `__enumext_check_ans_log:` and write the results in the `.log` file, undefine the environment `anskey*` (§11.28) through the function `__enumext_undefine_anskey_env:` and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

1992 \cs_new_protected:Npn \__enumext_execute_after_env:
1993 {
1994   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1995   {
1996     \tl_if_empty:NF \g__enumext_store_name_tl
1997     {
1998       \__enumext_stop_save_ans_msg:
1999       \__enumext_item_answer_diff:
2000       \__enumext_log_global_vars:
2001       \__enumext_log_answer_vars:
2002       \bool_if:NTF \g__enumext_check_ans_key_bool
2003       {
2004         \__enumext_check_ans_show:
2005       }
2006       { \__enumext_check_ans_log: }
2007       \__enumext_undefine_anskey_env:
2008     }
2009     \__enumext_reset_global_vars:
2010   }
2011 }

```

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

(End of definition for `__enumext_execute_after_env:`.)

11.26 Keys and functions associated with storage

wrap-ans
wrap-opt
save-sep
mark-ans
mark-pos
show-ans
mark-ref
save-ref

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*`.

```

2012 \cs_set_protected:Npn \__enumext_tmp:n #1
2013 {
2014   \keys_define:nn { enumext / #1 }
2015   {
2016     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2017     wrap-ans .initial:n = \fbox{##1},
2018     wrap-ans .value_required:n = true,

```

```

2019     wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2020     wrap-opt   .initial:n = [{##1}],
2021     wrap-opt   .value_required:n = true,
2022     save-sep   .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2023     save-sep   .initial:n = {, ~ },
2024     save-sep   .value_required:n = true,
2025     mark-ans   .tl_set:N = \__enumext_mark_answer_sym_tl,
2026     mark-ans   .initial:n = \textasteriskcentered,
2027     mark-ans   .value_required:n = true,
2028     mark-pos   .choice:,
2029     mark-pos / left   .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2030     mark-pos / right  .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2031     mark-pos / unknown .code:n =
2032                 \msg_error:nnee { enumext } { unknown-choice }
2033                 { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2034     mark-pos   .initial:n = right,
2035     mark-pos   .value_required:n = true,
2036     show-ans   .bool_set:N = \__enumext_show_answer_bool,
2037     show-ans   .initial:n = false,
2038     show-ans   .value_required:n = true,
2039     show-pos   .bool_set:N = \__enumext_show_position_bool,
2040     show-pos   .initial:n = false,
2041     show-pos   .value_required:n = true,
2042     mark-ref   .tl_set:N = \__enumext_mark_ref_sym_tl,
2043     mark-ref   .initial:n = \textasteriskcentered,
2044     mark-ref   .value_required:n = true,
2045     save-ref   .bool_set:N = \__enumext_store_ref_key_bool,
2046     save-ref   .initial:n = false,
2047     save-ref   .value_required:n = true,
2048   }
2049 }
2050 \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.

```

2051 \cs_set_protected:Npn \__enumext_tmp:n #1
2052 {
2053   \keys_define:nn { enumext / #1 }
2054   {
2055     mark-pos .choice:,
2056     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2057     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2058     mark-pos .initial:n = right,
2059     mark-pos .value_required:n = true,
2060     show-ans .bool_set:N = \__enumext_show_answer_bool,
2061     show-ans .initial:n = false,
2062     show-ans .value_required:n = true,
2063     show-pos .bool_set:N = \__enumext_show_position_bool,
2064     show-pos .initial:n = false,
2065     show-pos .value_required:n = true,
2066   }
2067 }
2068 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

11.26.1 Store optional arguments of the environments

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

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

```

2069 \cs_new_protected:Npn \__enumext_store_active_keys:n #1

```

```

2070 {
2071   \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2072   {
2073     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2074     \tl_set:ce
2075       { l__enumext_store_save_key_ \__enumext_level: _tl }
2076       { \__enumext_filter_save_key:n {#1} }
2077   }
2078 }
2079 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2080 {
2081   \bool_if:NF \l__enumext_store_save_key_vii_bool
2082   {
2083     \tl_clear:N \l__enumext_store_save_key_vii_tl
2084     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2085   }
2086 }

```

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

11.26.2 Setting save-key key

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

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

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

```

2087 \cs_set_protected:Npn \__enumext_tmp:n #1
2088 {
2089   \keys_define:nn { enumext / enumext* }
2090   {
2091     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2092     save-key .value_required:n = true,
2093   }
2094   \keys_define:nn { enumext / #1 }
2095   {
2096     save-key .code:n = \__enumext_parse_save_key:n {##1},
2097     save-key .value_required:n = true,
2098   }
2099 }
2100 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

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

```

2101 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2102 {
2103   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2104   \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2105   \tl_set:ce
2106     { l__enumext_store_save_key_ \__enumext_level: _tl }
2107     { \__enumext_filter_save_key:n {#1} }
2108 }
2109 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2110 {
2111   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2112   \tl_clear:N \l__enumext_store_save_key_vii_tl
2113   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2114 }

```

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

11.26.3 Internal functions to store optional arguments

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

```
2115 \cs_new:Npn \__enumext_filter_save_key:n #1
2116 {
2117   \use:e
2118   {
2119     \keyval_parse:NNn
2120     \__enumext_filter_save_key_key:n
2121     \__enumext_filter_save_key_pair:nn {#1}
2122   }
2123 }
```

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.

```
2124 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2125 {
2126   \str_case:nnF {#1}
2127   {
2128     { resume } {} { resume* } {} { no-store } {}
2129   }
2130   { , { \exp_not:n {#1} } }
2131 }
```

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.

```
2132 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2133 {
2134   \str_case:nnF {#1}
2135   {
2136     { series } {} { resume } {} { save-ans } {}
2137     { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2138     { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2139     { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2140     { mini-right } {} { mini-right* } {}
2141   }
2142   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2143 }
```

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

11.26.4 Function for storing content in prop list

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

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

```
2144 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2145 {
2146   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2147   {
2148     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2149   }
2150   { #1 }
2151 }
2152 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

11.26.5 Function for storing content in sequence

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

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

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

```
2153 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2154 {
```

```

2155 \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2156 }
2157 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }

```

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

11.26.6 Functions for storing the list structure in the sequence

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

```

2158 \cs_new_protected:Nn \__enumext_store_level_open:
2159 {
2160   \bool_if:NT \l__enumext_check_answers_bool
2161   {
2162     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2163     {
2164       \__enumext_store_addto_seq:n
2165       {
2166         \item \begin{enumext}
2167       }
2168     }
2169     {
2170       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2171       {
2172         \item \begin{enumext} [
2173       }
2174       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2175       {
2176         ]
2177       }
2178       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2179     }
2180   }
2181 }
2182 \cs_new_protected:Nn \__enumext_store_level_close:
2183 {
2184   \bool_if:NT \l__enumext_check_answers_bool
2185   {
2186     \__enumext_store_addto_seq:n { \end{enumext} }
2187   }
2188 }

```

(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.

```

2189 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2190 {
2191   \bool_if:NT \l__enumext_check_answers_bool
2192   {
2193     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2194     {
2195       \__enumext_store_addto_seq:n
2196       {
2197         %\item \mode_leave_vertical:
2198         % \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2199         % \begin{enumext*}[before={\setlength{\topsep}{\opt}}],
2200         \item \begin{enumext*}[base-fix]
2201       }
2202     }
2203     {
2204       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2205       {
2206         \item \mode_leave_vertical:
2207         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2208         \begin{enumext*}[before={\setlength{\topsep}{\opt}}],
2209       }
2210       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2211       {

```



```

2212         ]
2213     }
2214     \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2215 }
2216 }
2217 }
2218 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2219 {
2220     \bool_if:NT \l__enumext_check_answers_bool
2221     {
2222         \__enumext_store_addto_seq:n { \end{enumext*} }
2223     }
2224 }

```

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

11.26.7 Function for show marks and position

```

\__enumext_print_keyans_box:NN
\__enumext_print_keyans_box:cc

```

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

#1: `\l__enumext_labelwidth_X_dim`

#2: `\l__enumext_labelsep_X_dim`

```

2225 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2226 {
2227     \mode_leave_vertical:
2228     \skip_horizontal:n { -\dim_use:N #2 }
2229     \makebox[0pt][ r ]
2230     {
2231         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2232         {
2233             \tl_use:N \l__enumext_mark_answer_sym_tl
2234         }
2235     }
2236     \skip_horizontal:n { \dim_use:N #2 }
2237 }
2238 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

11.27 The command \anskey and internal label and ref

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

```

2239 \keys_define:nn { enumext / anskey }
2240 {
2241     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2242     break-col .default:n = true,
2243     break-col .value_forbidden:n = true,
2244     item-join .int_set:N = \l__enumext_store_item_join_int,
2245     item-join .value_required:n = true,
2246     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2247     item-star .default:n = true,
2248     item-star .value_forbidden:n = true,
2249     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2250     item-sym* .value_required:n = true,
2251     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2252     item-pos* .value_required:n = true,
2253 }

```

• 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`.

```

2254 \NewDocumentCommand \anskey { o +m }
2255 {
2256   \__enumext_anskey_safe_outer:
2257   \group_begin:
2258     \bool_if:NT \l__enumext_check_answers_bool
2259     {
2260       \int_gincr:N \g__enumext_item_anskey_int
2261       \__enumext_anskey_safe_inner:n {#2}
2262       \__enumext_store_anskey_code:nn {#1} {#2}
2263     }
2264   \group_end:
2265 }

```

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

11.27.1 Internal functions for the command

`__enumext_anskey_safe_outer:`
`__enumext_anskey_safe_inner:n`

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.

```

2266 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2267 {
2268   \bool_if:NF \l__enumext_store_active_bool
2269   {
2270     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2271   }
2272   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2273   {
2274     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2275   }
2276   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2277   {
2278     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2279   }
2280   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2281   {
2282     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2283   }
2284 }

```

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.

```

2285 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2286 {
2287   \tl_if_empty:nT {#1}
2288   {
2289     \msg_error:nn { enumext } { anskey-empty-arg }
2290   }
2291   \int_incr:N \l__enumext_anskey_level_int
2292   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2293   {
2294     \msg_error:nn { enumext } { anskey-nested }
2295   }
2296 }

```

(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.

```

2297 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2298 {
2299   \__enumext_store_addto_prop:n {#2}
2300   \bool_if:NT \l__enumext_store_ref_key_bool
2301   {
2302     \__enumext_store_internal_ref:
2303   }
2304   \__enumext_anskey_show_wrap_left:n { #2 }

```

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

```

2305 \tl_if_novalue:nF {#1}
2306 {
2307     \keys_set:nn { enumext / anskey } {#1}
2308 }
2309 \tl_clear:N \l__enumext_store_anskey_arg_tl
2310 \bool_lazy_and:nnT
2311 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2312 { \bool_not_p:n { \l__enumext_starred_bool } }
2313 {
2314     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2315 }
2316 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2317 \bool_lazy_and:nnT
2318 { \bool_not_p:n { \l__enumext_starred_bool } }
2319 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2320 {
2321     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2322     {
2323         ( \exp_not:V \l__enumext_store_item_join_int )
2324     }
2325 }

```

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

```

2326 \bool_if:NTF \l__enumext_store_item_star_bool
2327 {
2328     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2329     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2330     {
2331         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2332         {
2333             [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2334         }
2335     }
2336     \dim_compare:nT
2337     {
2338         \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2339     }
2340     {
2341         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2342         {
2343             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2344         }
2345     }
2346     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2347 }
2348 {
2349     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2350 }

```

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

```

2351 \bool_lazy_and:nnT
2352 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2353 { \bool_if_p:N \l__enumext_hyperref_bool }
2354 {
2355     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2356     {
2357         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2358         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2359     }
2360 }
2361 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2362 }

```

(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 `__enumext_newlabel_arg_two_tl`.

```

2363 \cs_new_protected:Nn \__enumext_store_internal_ref:
2364 {
2365   \cs_set_protected:Npn \__enumext_tmp:n ##1
2366   {
2367     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2368     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2369     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2370     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2371   }
2372   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2373   \cs_set:Npn \__enumext_tmp:n ##1
2374   { . \tl_use:c { \__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.

```

2375 \bool_lazy_all:nT
2376 {
2377   { \bool_if_p:N \g__enumext_starred_bool }
2378   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2379 }
2380 {
2381   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2382   { \tl_use:N \__enumext_label_copy_vii_tl }
2383 }
2384 \bool_lazy_all:nT
2385 {
2386   { \bool_if_p:N \l__enumext_standar_bool }
2387   { \bool_if_p:N \g__enumext_starred_bool }
2388   { \int_compare_p:nNn { \__enumext_level_int } > { 0 } }
2389 }
2390 {
2391   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2392   {
2393     \tl_use:N \__enumext_label_copy_vii_tl
2394     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2395   }
2396 }

```

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.

```

2397 \bool_lazy_all:nT
2398 {
2399   { \bool_if_p:N \l__enumext_standar_bool }
2400   { \int_compare_p:nNn { \__enumext_level_int } > { 0 } }
2401   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2402   { \bool_not_p:n { \g__enumext_starred_bool } }
2403 }
2404 {
2405   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2406   {
2407     \tl_use:N \__enumext_label_copy_i_tl
2408     \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
2409   }
2410 }
2411 \cs_set:Npn \__enumext_tmp:n ##1
2412 { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2413 \bool_lazy_all:nT
2414 {
2415   { \bool_if_p:N \l__enumext_standar_bool }
2416   { \int_compare_p:nNn { \__enumext_level_int } > { 0 } }
2417   { \bool_not_p:n { \g__enumext_starred_bool } }
2418   { \int_compare_p:nNn { \__enumext_level_h_int } > { 0 } }
2419 }
2420 {

```

```

2421         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2422         {
2423             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2424             . \tl_use:N \l__enumext_label_copy_vii_tl
2425         }
2426     }

```

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

```

2427     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2428     {
2429         \l__enumext_store_name_tl \c_colon_str
2430         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2431     }

```

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.

```

2432     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2433     {
2434         \__enumext_newlabel:nn
2435         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2436         { \l__enumext_newlabel_arg_two_tl }
2437     }
2438     \l__enumext_write_aux_file_tl
2439 }

```

(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.

```

2440 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2441 {
2442     \par
2443     \bool_if:NT \l__enumext_starred_bool
2444     {
2445         \cs_set:Nn \__enumext_level: { vii }
2446     }
2447     \__enumext_print_keyans_box:cc
2448     { \l__enumext_labelwidth_ \__enumext_level: _dim }
2449     { \l__enumext_labelsep_ \__enumext_level: _dim }
2450     \__enumext_anskey_wrapper:n { #1 }
2451 }

```

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

`__enumext_anskey_show_wrap_left:n`

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

```

2452 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2453 {
2454     \bool_if:NT \l__enumext_show_answer_bool
2455     {
2456         \__enumext_anskey_show_wrap_arg:n { #1 }
2457     }
2458     \bool_if:NT \l__enumext_show_position_bool
2459     {
2460         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2461         {
2462             \group_begin:
2463             \exp_not:N \normalfont
2464             \exp_not:N \footnotesize [ \int_eval:n
2465             {
2466                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2467             }
2468             ]
2469             \group_end:
2470         }
2471         \__enumext_anskey_show_wrap_arg:n { #1 }
2472     }
2473 }

```

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

11.28 The environment anskey*

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

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

`__enumext_undefine_anskey_env:`

The function `__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `__enumext_execute_after_env:` (§11.25) which is executed after the environment in which the key `save-ans` is active.

```
2474 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2475 {
2476   \cs_undefine:c { anskey* }
2477   \cs_undefine:c { endanskey* }
2478   \cs_undefine:c { __scontents_anskey*_env_begin: }
2479   \cs_undefine:c { __scontents_anskey*_env_end: }
2480 }
```

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

```
2481 \__enumext_before_env:nn { enumext }
2482 {
2483   \bool_lazy_and:nnT
2484     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2485     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2486     {
2487       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2488       {
2489         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2490       }
2491     }
2492 }
2493 \__enumext_before_env:nn { enumext* }
2494 {
2495   \bool_lazy_and:nnT
2496     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2497     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2498     {
2499       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2500       {
2501         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2502       }
2503     }
2504 }
```

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

```
2505 \__enumext_before_env:nn { anskey* }
2506 {
2507   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2508   {
2509     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2510   }
2511   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2512   {
2513     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2514   }
2515   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2516   {
2517     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2518   }
2519 }
```

(End of definition for `__enumext_undefine_anskey_env:`.)

anskey*

The function `__enumext_anskey_env_make:n` creates the environment **anskey*** (*custom version of `scontents` environment*) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`. To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec:` (§11.24.1).

```

2520 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2521 {
2522   \newenvsc{anskey*}[store-env=#1,print-env=false]
2523   \__enumext_anskey_env_exec:
2524 }
2525 \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`.

```

2526 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2527 {
2528   \keys_define:nn { scontents / scontents }
2529   {
2530     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2531     break-col .default:n = true,
2532     break-col .value_forbidden:n = true,
2533     item-join .int_gset:N = \g__enumext_store_item_join_int,
2534     item-join .value_required:n = true,
2535     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2536     item-star .default:n = true,
2537     item-star .value_forbidden:n = true,
2538     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2539     item-sym* .value_required:n = true,
2540     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2541     item-pos* .value_required:n = true,
2542     print-env .undefine:,
2543     store-env .undefine:,
2544     write-out .undefine:,
2545   }
2546 }

```

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`.

```

2547 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2548 {
2549   \keys_define:nn { scontents / scontents }
2550   {
2551     break-col .undefine:,
2552     item-join .undefine:,
2553     item-star .undefine:,
2554     item-sym* .undefine:,
2555     item-pos* .undefine:,
2556     write-out .code:n = {
2557       \bool_set_false:N \l__scontents_storing_bool
2558       \bool_set_true:N \l__scontents_writing_bool
2559       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2560     },
2561     write-out .value_required:n = true,
2562     print-env .meta:nn = { scontents } { print-env = ##1 },
2563     print-env .default:n = true,
2564     store-env .meta:nn = { scontents } { store-env = ##1 },
2565     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} }
2566   }
2567 }

```

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*.

```

2568 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2569 {
2570   \group_begin:
2571   \int_set:Nn \tex_newlinechar:D { `^^J }
2572   \__scontents_rescan_tokens:x
2573   {

```



```

2574         \endgroup % This assumes \catcode\|=0... Things might go off otherwise.
2575         #1
2576     }
2577 }

```

(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)`.

```

2578 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2579 {
2580     \__enumext_before_env:nn { anskey* }
2581     {
2582         \__enumext_anskey_env_define_keys:
2583     }

```

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.

```

2584     \hook_if_empty:nF {env/anskey*/after}
2585     {
2586         \hook_gremove_code:nn {env/anskey*/after} { * }
2587     }
2588     \__enumext_after_env:nn { anskey* }
2589     {
2590         \tl_clear:N \l__enumext_store_anskey_env_tl
2591         \tl_clear:N \l__enumext_store_anskey_opt_tl
2592         \tl_gset:Ne \l__enumext_store_anskey_env_tl
2593         {
2594             \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2595         }
2596         \__enumext_anskey_env_keys:
2597         \__enumext_anskey_env_store:
2598         \__enumext_anskey_env_clean:
2599         \__enumext_anskey_env_undefine_keys:
2600     }
2601 }

```

• 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`.

```

2602 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2603 {
2604     \bool_lazy_and:nnT
2605     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2606     { \bool_not_p:n { \l__enumext_starred_bool } }
2607     {
2608         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2609     }

```

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

```

2610     \bool_lazy_and:nnT
2611     { \bool_not_p:n { \l__enumext_starred_bool } }
2612     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2613     {
2614         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2615         {
2616             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2617         }
2618     }

```

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

```

2619     \bool_if:NT \g__enumext_store_item_star_bool
2620     {

```

```

2621 \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2622 {
2623   ,item-star,
2624 }
2625 \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2626 {
2627   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2628   {
2629     ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2630   }
2631 }
2632 \dim_compare:nT
2633 {
2634   \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2635 }
2636 {
2637   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2638   {
2639     ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2640   }
2641 }
2642 }
2643 }

```

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`.

```

2644 \cs_new_protected:Nn \__enumext_anskey_env_store:
2645 {
2646   \group_begin:
2647   \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2648   {
2649     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2650     {
2651       \exp_args:Ne
2652       \anskey
2653       {
2654         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2655       }
2656     }
2657     {
2658       \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2659       \exp_args:Ne
2660       \anskey
2661       {
2662         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2663       }
2664     }
2665   }
2666   \group_end:
2667 }

```

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

```

2668 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2669 {
2670   \bool_gset_false:N \g__enumext_store_columns_break_bool
2671   \int_gzero:N \g__enumext_store_item_join_int
2672   \bool_gset_false:N \g__enumext_store_item_star_bool
2673   \tl_gclear:N \g__enumext_store_item_symbol_tl
2674   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2675 }

```

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

11.29 Common functions for keyans, keyans* and keyanspic

11.29.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the contents of the current `⟨label⟩` `\l__enumext_label_v_tl` for the `keyans` environment and the current `⟨label⟩` `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the

optional argument of both commands to the `\l__enumext_store_current_label_tl` variable, which will be passed to the *(prop list)* defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

2676 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2677 {
2678   \tl_clear:N \l__enumext_store_current_label_tl
2679   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2680   {
2681     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2682   }
2683   {
2684     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2685   }
2686   \tl_if_novalue:nF { #1 }
2687   {
2688     % Set save-sep
2689     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2690     {
2691       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2692     }
2693     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2694   }
2695   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2696 }

```

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

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

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

```

\__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.

```

2697 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2698 {
2699   \bool_if:NT \l__enumext_store_ref_key_bool
2700   {
2701     \cs_set_protected:Npn \__enumext_tmp:n ##1
2702     {
2703       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2704       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2705       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2706       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2707     }
2708     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2709     \__enumext_keyans_store_ref_aux_i:
2710   }
2711 }

```

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

```

2712 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2713 {
2714   \bool_if:NT \g__enumext_starred_bool
2715   {
2716     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2717   }
2718   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2719   {
2720     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2721     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2722   }
2723   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2724   {
2725     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2726     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }

```

```

2727     }
2728     \int_compare:nNtT { \l__enumext_keyans_level_h_int } = { 1 }
2729     {
2730         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2731         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2732     }
2733     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2734     {
2735         \l__enumext_store_name_tl \c_colon_str
2736         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2737     }
2738     \__enumext_keyans_store_ref_aux_ii:
2739 }

```

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.

```

2740 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2741 {
2742     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2743     {
2744         \__enumext_newlabel:nn
2745         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2746         { \l__enumext_newlabel_arg_two_tl }
2747     }
2748     \l__enumext_write_aux_file_tl
2749 }

```

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

11.29.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

2750 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2751 {
2752     \tl_clear:N \l__enumext_store_current_label_tl
2753     \int_compare:nNtTF { \l__enumext_keyans_pic_level_int } = { 1 }
2754     {
2755         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2756     }
2757     {
2758         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2759     }
2760     \tl_if_novalue:nF { #1 }
2761     {
2762         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2763         {
2764             \tl_put_right:Ne \l__enumext_store_current_label_tl
2765             {
2766                 \l__enumext_store_keyans_item_opt_sep_tl
2767             }
2768         }
2769         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2770     }
2771     \__enumext_keyans_addto_seq_link:
2772 }

```

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.

```

2773 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2774 {
2775     \bool_lazy_and:nnT
2776     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2777     { \bool_if_p:N \l__enumext_hyperref_bool }

```

```

2778     {
2779         \tl_put_right:Ne \l__enumext_store_current_label_tl
2780         {
2781             \hfill \exp_not:N \hyperlink
2782             {
2783                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2784             }
2785             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2786         }
2787     }
2788     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2789     \bool_if:NT \l__enumext_check_answers_bool
2790     {
2791         \int_gincr:N \g__enumext_item_anskey_int
2792     }
2793 }

```

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

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

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off `<label>` are incorrect.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:
2794 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2795 {
2796     \tl_if_novalue:nF { #1 }
2797     {
2798         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2799     }
2800     \bool_if:NT \l__enumext_show_answer_bool
2801     {
2802         \__enumext_keyans_show_ans:
2803     }
2804     \bool_if:NT \l__enumext_show_position_bool
2805     {
2806         \__enumext_keyans_show_pos:
2807     }
2808 }
2809 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2810 {
2811     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2812     {
2813         \bool_lazy_or:nnT
2814         { \bool_if_p:N \l__enumext_show_answer_bool }
2815         { \bool_if_p:N \l__enumext_show_position_bool }
2816         {
2817             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2818         }
2819     }
2820 }
2821 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2822 {
2823     \tl_put_left:Nn \l__enumext_label_v_tl
2824     {
2825         \__enumext_print_keyans_box:NN
2826         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2827     }
2828 }
2829 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2830 {
2831     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2832     {
2833         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2834         {
2835             \group_begin:
2836             \exp_not:N \normalfont
2837             \exp_not:N \footnotesize [ \int_eval:n
2838                 {
2839                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }

```

```

2840         }
2841     ]
2842     \group_end:
2843 }
2844 }
2845 {
2846     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2847     {
2848         \group_begin:
2849         \exp_not:N \normalfont
2850         \exp_not:N \footnotesize [ \int_eval:n
2851             {
2852                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2853             }
2854         ]
2855         \group_end:
2856     }
2857 }
2858 \tl_put_left:Nn \l__enumext_label_v_tl
2859 {
2860     \__enumext_print_keyans_box:NN
2861     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2862 }
2863 }

```

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

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

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

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

```

item-pos* 2864 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2865 {
2866     \keys_define:nn { enumext / #1 }
2867     {
2868         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2869         item-sym* .value_required:n = true,
2870         item-sym* .initial:n = { $\star$ },
2871         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2872         item-pos* .value_required:n = true,
2873     }
2874 }
2875 \clist_map_inline:nn
2876 {
2877     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2878 }
2879 { \__enumext_tmp:nn #1 }

```

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

11.31 Redefining `\footnote` command

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

```

2880 \cs_new_protected:Nn \__enumext_footnotetext:nn
2881 {
2882     \footnotetext[#1]{#2}
2883 }
2884 \cs_new_protected:Nn \__enumext_renew_footnote:
2885 {
2886     \seq_gclear:N \g__enumext_footnote_arg_seq
2887     \seq_gclear:N \g__enumext_footnote_int_seq
2888     \RenewDocumentCommand \footnote { o +m }
2889     {
2890         \tl_if_novalue:nTF {##1}
2891         {
2892             \stepcounter{footnote}
2893             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2894         }

```

```

2895         {
2896             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2897         }
2898         \footnotemark [ \g__enumext_footnote_int ]
2899         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2900         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2901     }
2902 }
2903 \cs_new_protected:Nn \__enumext_print_footnote:
2904 {
2905     \seq_if_empty:NF \g__enumext_footnote_int_seq
2906     {
2907         \seq_map_pairwise_function:NNN
2908         \g__enumext_footnote_int_seq
2909         \g__enumext_footnote_arg_seq
2910         \__enumext_footnotetext:nn
2911     }
2912 }

```

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

11.32 Redefining \item command

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

11.32.1 The \item command in enumext

`__enumext_default_item:n` The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext`.

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

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

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

```

2913 \cs_new_protected:Npn \__enumext_default_item:n #1
2914 {
2915     \tl_if_novalue:nTF {#1}
2916     {
2917         \bool_if:NT \l__enumext_check_answers_bool
2918         {
2919             \int_gincr:N \g__enumext_item_number_int
2920         }
2921         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2922         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2923     }
2924     {
2925         \bool_set_eq:cc
2926         { \l__enumext_wrap_label_ \__enumext_level: _bool }
2927         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2928         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2929     }
2930 }

```

(End of definition for __enumext_default_item:n)

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as optional argument in the global variable `\g__enumext_item_symbol_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item*-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:` (§11.33).

```

2931 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2932 {
2933   \tl_if_novalue:nF {#1}
2934   {
2935     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2936   }
2937   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2938   \tl_if_novalue:nTF {#2}
2939   {
2940     \dim_set_eq:cc
2941     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2942     { l__enumext_labelsep_ \__enumext_level: _dim }
2943   }
2944   {
2945     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2946   }
2947   \bool_if:NT \l__enumext_check_answers_bool
2948   {
2949     \int_gincr:N \g__enumext_item_number_int
2950   }
2951   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2952   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2953 }

```

(End of definition for `__enumext_starred_item:nn`)

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

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

```

2954 \cs_new_protected:Nn \__enumext_redefine_item:
2955 {
2956   \RenewDocumentCommand \item { s o o }
2957   {
2958     \bool_if:nTF {##1}
2959     {
2960       \__enumext_starred_item:nn {##2} {##3}
2961     }
2962     { \__enumext_default_item:n {##2} }
2963   }
2964 }

```

(End of definition for `__enumext_redefine_item:.`)

11.32.2 The `\item` command in keyans

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

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

```

2965 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2966 {
2967   \tl_if_novalue:nTF { #1 }
2968   {
2969     \bool_set_true:N \l__enumext_wrap_label_v_bool
2970     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2971   }
2972   {
2973     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2974     \__enumext_item_std:w [ #1 ] \tl_use:N \l__enumext_fake_item_indent_v_tl
2975   }
2976 }

```

(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 $\langle label \rangle$.

```
2977 \cs_new_protected:Npn \_enumext_keyans_starred_item:n #1
2978 {
2979   \tl_set_eq:NN \_enumext_store_current_label_tmp_tl \_enumext_label_v_tl
2980   \_enumext_keyans_show_left:n { #1 }
2981   \bool_set_true:N \_enumext_wrap_label_v_bool
2982   \_enumext_item_std:w \tl_use:N \_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 $\langle sequence \rangle$.

```
2983   \tl_set_eq:NN \_enumext_label_v_tl \_enumext_store_current_label_tmp_tl
2984   \_enumext_keyans_addto_prop:n { #1 }
2985   \_enumext_keyans_store_ref:
2986   \_enumext_keyans_addto_seq:n { #1 }
2987   \int_gincr:N \g__enumext_check_starred_cmd_int
2988 }
```

(End of definition for _enumext_keyans_starred_item:n.)

_enumext_keyans_redefine_item:
 \item*

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

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

```
2989 \cs_new_protected:Npn \_enumext_keyans_redefine_item:
2990 {
2991   \RenewDocumentCommand \item { s o }
2992   {
2993     \bool_if:nTF {##1}
2994     {
2995       \peek_remove_spaces:n
2996       {
2997         \_enumext_keyans_starred_item:n {##2}
2998       }
2999     }
3000     {
3001       \_enumext_keyans_default_item:n {##2}
3002     }
3003   }
3004 }
```

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

11.33 Redefining \makeLabel command

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

11.33.1 Redefining \makeLabel for enumext

_enumext_item_starred:

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

```
3005 \cs_new_protected:Npn \_enumext_item_starred:
3006 {
3007   \tl_if_empty:cF { \_enumext_item_symbol_ \_enumext_level: _tl }
3008   {
3009     \mode_leave_vertical:
3010     \skip_horizontal:n { -\dim_use:c { \_enumext_item_symbol_sep_ \_enumext_level: _dim } }
3011     \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_tl }
3012     \skip_horizontal:n { \dim_use:c { \_enumext_item_symbol_sep_ \_enumext_level: _dim } }
3013   }
3014 }
```

(End of definition for _enumext_item_starred:.)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makelabel` for the `enumext` environment. This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.34.2).

```

3015 \cs_new_protected:Nn \__enumext_make_label:
3016 {
3017   \RenewDocumentCommand \makelabel { m }
3018   {
3019     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
3020     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
3021     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
3022     {
3023       \__enumext_item_starred:
3024       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3025     }
3026     { ##1 }
3027     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
3028     \tl_gclear:N \g__enumext_item_symbol_tl
3029   }
3030 }

```

(End of definition for `__enumext_make_label:`)

11.33.2 Redefining `\makelabel` for `keyans`

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

```

3031 \cs_new_protected:Nn \__enumext_keyans_make_label:
3032 {
3033   \RenewDocumentCommand \makelabel { m }
3034   {
3035     \tl_use:N \l__enumext_label_fill_left_v_tl
3036     \tl_use:N \l__enumext_label_font_style_v_tl
3037     \bool_if:NTF \l__enumext_wrap_label_v_bool
3038     {
3039       \__enumext_wrapper_label_v:n { ##1 }
3040     }
3041     { ##1 }
3042     \tl_use:N \l__enumext_label_fill_right_v_tl
3043   }
3044 }

```

(End of definition for `__enumext_keyans_make_label:`)

11.34 Second argument of the lists

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

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

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

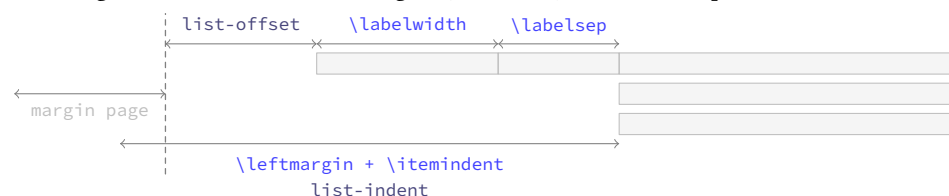
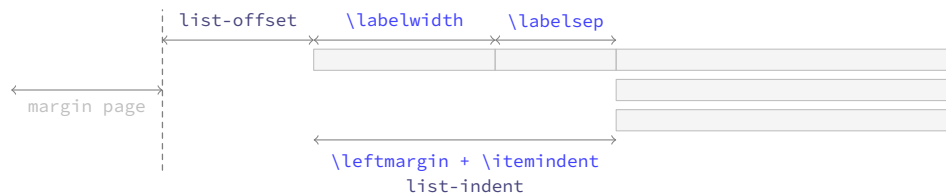
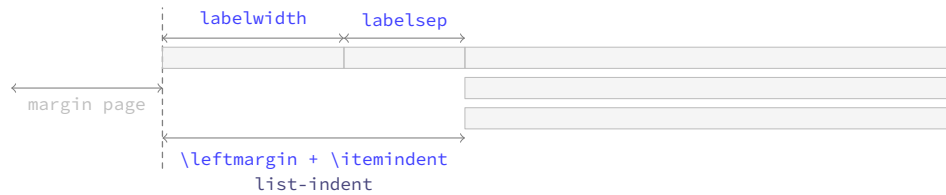


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

`__enumext_calc_hspace:NNNNNNN`
`__enumext_calc_hspace:ccccccc`

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

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

```

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

```

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

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

```

3045 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3046 {
3047   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3048   {
3049     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3050     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3051   }
3052   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3053   {
3054     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3055     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3056   }

```

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

```

3057   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3058   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3059   {
3060     \dim_set:Nn #6 { #1 + #2 - #4 }
3061     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3062   }
3063   {
3064     \dim_compare:nNnT { #4 } = { #1 + #2 }
3065     { \dim_set:Nn #6 { \c_zero_dim } }
3066     \dim_compare:nNnT { #4 } < { #1 + #2 }
3067     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3068     \dim_compare:nNnT { #4 } > { #1 + #2 }
3069     {
3070       \dim_set:Nn #6 { -#1 - #2 + #4 }
3071       \dim_set:Nn #6 { #6*-1 }
3072     }
3073     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3074   }
3075 }
3076 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNN { ccccccc }

```

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

11.34.2 Setting second argument of the lists

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

```

3077 \cs_set_protected:Npn \__enumext_tmp:n #1
3078 {
3079   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3080   {
3081     \__enumext_calc_hspace:cccccc
3082     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3083     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3084     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3085     { \__enumext_leftmargin_tmp_#1_bool }
3086     \clist_map_inline:nn
3087       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3088       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3089     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3090       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3091     \usecounter { enumX#1 }
3092     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3093     \str_if_eq:nnTF {#1} { v }
3094     {
3095       \__enumext_keyans_redefine_item:
3096       \__enumext_keyans_make_label:
3097       \__enumext_keyans_ref:
3098       \__enumext_keyans_fake_item:
3099       \bool_if:cT { \__enumext_show_length_#1_bool }
3100       {
3101         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3102       }
3103     }
3104     {
3105       \__enumext_redefine_item:
3106       \__enumext_make_label:
3107       \__enumext_standar_ref:
3108       \__enumext_fake_item:
3109       \bool_if:cT { \__enumext_show_length_#1_bool }
3110       {
3111         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
3112       }
3113     }
3114   }
3115 }
3116 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

3117 \cs_set_protected:Npn \__enumext_tmp:n #1
3118 {
3119   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3120   {
3121     \__enumext_calc_hspace:cccccc
3122     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3123     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3124     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3125     { \__enumext_leftmargin_tmp_#1_bool }
3126     \clist_map_inline:nn
3127       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3128       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3129     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3130       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3131     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3132     \skip_zero:N \partopsep
3133     \usecounter { enumX#1 }
3134     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3135     \__enumext_starred_ref:
3136     \str_if_eq:nnTF {#1} { vii }

```

```

3137     {
3138         \__enumext_fake_item_vii:
3139         \bool_if:cT { l__enumext_show_length_vii_bool }
3140         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3141     }
3142     {
3143         \__enumext_fake_item_viii:
3144         \bool_if:cT { l__enumext_show_length_#1_bool }
3145         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3146     }
3147 }
3148 }
3149 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

11.35 The environment enumext

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

```

3150 \NewDocumentEnvironment{enumext}{0}{}
3151 {
3152     \__enumext_safe_exec:
3153     \__enumext_parse_keys:n {#1}
3154     \__enumext_before_list:
3155     \__enumext_start_store_level:
3156     \__enumext_start_list:nn
3157     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3158     {
3159         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3160         \__enumext_before_keys_exec:
3161     }
3162     \__enumext_after_args_exec:
3163 }
3164 {
3165     \__enumext_stop_list:
3166     \__enumext_stop_store_level:
3167     \__enumext_after_list:
3168 }

```

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

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

```

3169 \cs_new_protected:Nn \__enumext_safe_exec:
3170 {
3171     \__enumext_internal_mini_page:
3172     \__enumext_is_not_nested:
3173     \int_incr:N \l__enumext_level_int
3174     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3175     { \msg_fatal:nn { enumext } { list-too-deep } }
3176     \bool_set_true:N \l__enumext_standar_bool
3177     \__enumext_is_on_first_level:
3178 }

```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n The **__enumext_parse_store_keys:n** function will parse the **(keys)** passed to the optional environment argument **enumext** by levels only if present. First we clear the variable **\l__enumext_series_str** and then we check if we are at the first level, if so we process the **(keys)** and then execute the function **__enumext_parse_series:n** used by the key **series**, otherwise we will pass the **(keys)** to the inner levels of the environment and finally if the variable **\l__enumext_store_active_bool** established by the key **save-ans** is true we execute **__enumext_parse_store_keys:n** used by the key **save-key**.

```

3179 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3180 {
3181     \tl_if_novalue:nF {#1}
3182     {
3183         \str_clear:N \l__enumext_series_str

```

```

3184         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3185         {
3186             \keys_set:nn { enumext / level-1 } {#1}
3187             \__enumext_parse_series:n {#1}
3188             \__enumext_nested_base_line_fix:
3189         }
3190         {
3191             \exp_args:Ne \keys_set:nn
3192             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3193         }
3194         \__enumext_store_active_keys:n {#1}
3195     }
3196 }

```

(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.

```

3197 \cs_new_protected:Nn \__enumext_start_store_level:
3198 {
3199     \bool_lazy_all:nT
3200     {
3201         { \bool_if_p:N \l__enumext_store_active_bool }
3202         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3203         { \bool_not_p:n { \g__enumext_starred_bool } }
3204     }
3205     {
3206         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3207         {
3208             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3209             \__enumext_store_level_open:
3210         }
3211     }
3212     \bool_lazy_all:nT
3213     {
3214         { \bool_if_p:N \l__enumext_store_active_bool }
3215         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3216         { \bool_if_p:N \g__enumext_starred_bool }
3217     }
3218     {
3219         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3220         {
3221             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3222             \__enumext_store_level_open:
3223         }
3224     }
3225 }
3226 \cs_new_protected:Nn \__enumext_stop_store_level:
3227 {
3228     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3229     {
3230         \__enumext_store_level_close:
3231     }
3232 }

```

(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.

```

3233 \cs_new_protected:Nn \__enumext_before_list:
3234 {
3235     \__enumext_vspace_above:
3236     \__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.

```

3237     \__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.

```

3238 \dim_compare:nNt
3239 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3240 {
3241   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3242   {
3243     \linewidth
3244     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3245     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3246   }

```

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

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

```

3247 \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3248 \int_gincr:N \g__enumext_minipage_stat_int
3249 \__enumext_mini_addvspace:
3250 \nointerlineskip\noindent
3251 \begin{__enumext_mini_env*}
3252   { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3253 }
3254 \__enumext_multicols_start:
3255 }

```

(End of definition for `__enumext_before_list:`)

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

```

3256 \cs_new_protected:Nn \__enumext_multicols_start:
3257 {
3258   \int_compare:nNt
3259   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3260   {
3261     \dim_compare:nNt
3262     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3263     {
3264       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3265       {
3266         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3267         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3268         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3269         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3270       }
3271     }
3272     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3273     \skip_zero:N \multicolsep
3274     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3275     {
3276       \dim_zero:N \columnseprule
3277     }

```

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.

```

3278 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3279 {
3280   \__enumext_multi_addvspace:
3281 }
3282 \raggedcolumns

```

```

3283     \begin{multicols}{\int_use:c { \l__enumext_columns_ \l__enumext_level: _int } }
3284   }
3285 }

```

(End of definition for `__enumext_multicols_start:`)

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

```

3286 \cs_new_protected:Nn \__enumext_multicols_stop:
3287 {
3288   \int_compare:nNtT
3289     { \int_use:c { \l__enumext_columns_ \l__enumext_level: _int } } > { 1 }
3290     {
3291       \end{multicols}
3292       \bool_if:cF { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3293       {
3294         \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \l__enumext_level: _skip } }
3295       }
3296     }
3297 }

```

(End of definition for `__enumext_multicols_stop:`)

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

```

3298 \cs_new_protected:Nn \__enumext_after_list:
3299 {
3300   \bool_if:cTF { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3301   {
3302     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3303     {
3304       \msg_warning:nn { enumext } { missing-miniright }
3305       \miniright
3306     }
3307     \int_gzero:N \g__enumext_minipage_stat_int
3308     \end{__enumext_mini_env*}
3309     \par\addvspace { \l__enumext_minipage_after_skip }
3310   }
3311   { \__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`.

```

3312   \__enumext_check_ans_key_hook:

```

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

```

3313   \__enumext_after_stop_list:
3314   \__enumext_vspace_below:
3315   \bool_set_false:N \l__enumext_standar_bool
3316   \__enumext_resume_save_counter:
3317 }

```

(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`.

```

3318 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

11.36 The environment keyans

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

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

```

3319 \NewDocumentEnvironment{keyans}{0}{ }
3320 {
3321   \__enumext_keyans_safe_exec:
3322   \__enumext_keyans_parse_keys:n {#1}
3323   \__enumext_before_list_v:
3324   \__enumext_start_list:nn
3325   { \tl_use:N \__enumext_label_v_tl }
3326   {
3327     \__enumext_list_arg_two_v:
3328     \__enumext_before_keys_exec_v:
3329   }
3330   \__enumext_after_args_exec_v:
3331 }
3332 {
3333   \__enumext_check_starred_cmd:n { item }
3334   \__enumext_stop_list:
3335   \__enumext_after_list_v:
3336 }

```

(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.

```

3337 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3338 {
3339   \bool_if:NF \l__enumext_store_active_bool
3340   {
3341     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3342   }
3343   \int_incr:N \l__enumext_keyans_level_int
3344   \bool_set_true:N \l__enumext_keyans_env_bool
3345   \__enumext_keyans_start_line:
3346   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3347   \bool_set_false:N \l__enumext_store_active_bool
3348   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3349   {
3350     \msg_error:nn { enumext } { keyans-nested }
3351   }
3352   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3353   {
3354     \msg_error:nn { enumext } { keyans-wrong-level }
3355   }
3356 }

```

(End of definition for **__enumext_keyans_safe_exec:**)

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

```

3357 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3358 {
3359   \keys_set:nn { enumext / keyans } {#1}
3360 }

```

(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.

```

3361 \cs_new_protected:Nn \__enumext_before_list_v:
3362 {
3363   \__enumext_vspace_above_v:
3364   \__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.

```

3365   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3366   {
3367     \dim_set:Nn \l__enumext_minipage_left_v_dim

```

```

3368         {
3369             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3370         }

```

The boolean variable `\l__enumext_minipage_active_v_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_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*.

```

3371         \bool_set_true:N \l__enumext_minipage_active_v_bool
3372         \int_gincr:N \g__enumext_minipage_stat_int
3373         \__enumext_keyans_mini_addvspace:
3374         \nointerlineskip\noindent
3375         \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3376     }

```

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

```

3377     \__enumext_keyans_multicols_start:
3378 }

```

(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.

```

3379 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3380 {
3381     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3382     {

```

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

```

3383         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3384         {
3385             \dim_set:Nn \l__enumext_columns_sep_v_dim
3386             {
3387                 (
3388                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3389                 ) / \l__enumext_columns_v_int
3390                 - \l__enumext_listoffset_v_dim
3391             }
3392         }
3393         \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).

```

3394         \skip_zero:N \multicolsep
3395         \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.

```

3396         \bool_if:NF \l__enumext_minipage_active_v_bool
3397         {
3398             \__enumext_keyans_multi_addvspace:
3399         }
3400         \raggedcolumns
3401         \begin{multicols}{ \l__enumext_columns_v_int }
3402     }
3403 }

```

(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.

```

3404 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3405 {
3406     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3407     {
3408         \end{multicols}
3409         \bool_if:NF \l__enumext_minipage_active_v_bool

```

```

3410         {
3411         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3412         }
3413     }
3414 }

```

(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.

```

3415 \cs_new_protected:Nn \__enumext_after_list_v:
3416 {
3417     \bool_if:NTF \l__enumext_minipage_active_v_bool
3418     {
3419         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3420         {
3421             \msg_warning:nn { enumext } { missing-miniright }
3422             \miniright
3423         }
3424         \int_gzero:N \g__enumext_minipage_stat_int
3425         \end{__enumext_mini_env*}
3426         \par\addvspace{ \l__enumext_minipage_after_skip }
3427     }
3428     { \__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.

```

3429     \bool_set_false:N \l__enumext_keyans_env_bool
3430     \__enumext_after_stop_list_v:
3431     \__enumext_vspace_below_v:
3432 }

```

(End of definition for `__enumext_after_list_v:`.)

11.37 The environment `keyanspic` and `\anspic`

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

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

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

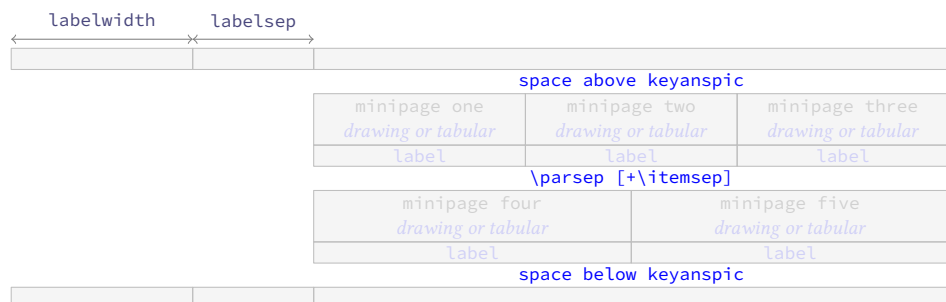


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

11.37.1 The command `\anspic`

`\anspic` The `\anspic` command takes three arguments, the starred (*) versions `\anspic*` and `\anspic*[\content]` store the current `\label` next to the `[\content]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```

3433 \NewDocumentCommand \anspic { s o +m }
3434 {

```

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

```

3435 \bool_if:NF \__enumext_store_active_bool
3436 {
3437   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3438 }
3439 \int_compare:nNt { \__enumext_level_int } > { 1 }
3440 {
3441   \msg_error:nn { enumext } { keyanspic-wrong-level }
3442 }
3443 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
3444 {
3445   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3446 }

```

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.

```

3447 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3448 {
3449   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3450 }
3451 }

```

(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.

```

3452 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3453 {
3454   \stepcounter { enumXvi }
3455   #3 \\\
3456   \bool_if:nT { #1 }
3457   {
3458     \__enumext_keyans_addto_prop:n { #2 }
3459     \__enumext_keyans_store_ref:
3460     \__enumext_keyans_addto_seq:n { #2 }
3461     \int_gincr:N \__enumext_check_starred_cmd_int
3462     \bool_lazy_or:nnT
3463     { \bool_if_p:N \__enumext_show_answer_bool }
3464     { \bool_if_p:N \__enumext_show_position_bool }
3465     {
3466       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3467       \__enumext_keyans_show_left:n { #2 }
3468       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3469     }
3470   }
3471   \tl_use:N \__enumext_label_font_style_v_tl
3472   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3473 }

```

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

11.37.2 The environment `keyanspic`

`keyanspic`

Now we define the environment `keyanspic` based on list. The optional argument [`<number above, number below>`] will determine the number of `minipage` environments that will be above and below separated by `\parsep+ \itemsep` within it.

```

3474 \NewDocumentEnvironment{keyanspic}{ o }
3475 {
3476   \__enumext_keyans_pic_safe_exec:
3477   \__enumext_start_list:nn
3478   { }
3479   {
3480     \__enumext_keyans_pic_arg_two:
3481   }

```

We apply the “adjusted” vertical spacing above the environment

```

3482 \vspace { \__enumext_keyans_pic_above_skip }
3483 }

```

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.

```

3484 {
3485   \tl_if_novalue:nTF { #1 }
3486   {
3487     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3488   }
3489   { \__enumext_keyans_pic_do:n { #1 } }
3490   \__enumext_stop_list:
3491   \__enumext_check_starred_cmd:n { anspic }
3492   \setcounter { enumXvi } { 0 }
3493   \vspace { \l__enumext_topsep_v_skip }
3494   %\bool_set_false:N \l__enumext_store_active_bool
3495 }

```

(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.

```

3496 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3497 {
3498   \int_incr:N \l__enumext_keyans_pic_level_int
3499   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3500   {
3501     \msg_error:nn { enumext } { keyanspic-nested }
3502   }
3503   \__enumext_keyans_start_line:
3504 }

```

(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`.

```

3505 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3506 {
3507   \dim_compare:nNnT { #1 } < { 0pt }
3508   { \skip_set:Nn #1 { -#1 } }
3509 }

```

(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.

```

3510 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3511 {

```

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.

```

3512   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3513   \__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.

```

3514   \skip_add:Nn \parsep { \itemsep }
3515   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3516   \dim_zero:N \labelwidth
3517   \dim_zero:N \listparindent
3518   \dim_zero:N \labelsep
3519   \skip_zero:N \partopsep
3520   \skip_zero:N \itemsep

```

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

```

3521   \__enumext_keyans_pic_skip_abs:N \parsep
3522   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3523   {

```



```

3524         \box_dp:N \strutbox
3525         + \l__enumext_topsep_v_skip
3526         - \parsep
3527     }
3528     \__enumext_item_std:w \scan_stop:
3529 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

The optional argument is split by comma and is handled directly by the function __enumext_keyans_pic_do:n and passed to the function __enumext_keyans_pic_row:n.

```

3530 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3531 {
3532     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3533 }
3534 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

__enumext_keyans_pic_row:n

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

```

3535 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3536 {
3537     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3538     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3539     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3540     \int_step_inline:nnn
3541     { \l__enumext_keyans_pic_above_int + 1 }
3542     { \l__enumext_keyans_pic_below_int }
3543     {
3544         \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3545         \centering
3546         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3547         \__enumext_endminipage:
3548     }
3549     \par
3550 }

```

(End of definition for __enumext_keyans_pic_row:n.)

11.38 The horizontal environments

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

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

11.38.1 Functions for item box width

__enumext_starred_columns_set_vii:
__enumext_starred_columns_set_viii:

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3551 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3552 {
3553     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3554     {
3555         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3556         {
3557             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3558             / \l__enumext_columns_vii_int
3559         }
3560     }
3561     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3562     \dim_set:Nn \l__enumext_item_width_vii_dim
3563     {
3564         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )

```

```

3565         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3566         - \l__enumext_labelsep_vii_dim
3567     }
3568     \dim_zero_new:N \itemwidth
3569 }
3570 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3571 {
3572     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3573     {
3574         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3575         {
3576             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3577             / \l__enumext_columns_viii_int
3578         }
3579     }
3580     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3581     \dim_set:Nn \l__enumext_item_width_viii_dim
3582     {
3583         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3584         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3585         - \l__enumext_labelsep_viii_dim
3586     }
3587     \dim_zero_new:N \itemwidth
3588 }

```

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

11.38.2 Functions for join item columns

__enumext_starred_joined_item_vii:n
 __enumext_starred_joined_item_viii:n

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

```

3589 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3590 {
3591     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3592     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3593     {
3594         \msg_warning:nnee { enumext } { item-joined }
3595         { \int_use:N \l__enumext_joined_item_vii_int }
3596         { \int_use:N \l__enumext_columns_vii_int }
3597         \int_set:Nn \l__enumext_joined_item_vii_int
3598         {
3599             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3600         }
3601     }
3602     \int_compare:nNnT
3603     { \l__enumext_joined_item_vii_int }
3604     >
3605     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3606     {
3607         \msg_warning:nnee { enumext } { item-joined-columns }
3608         { \int_use:N \l__enumext_joined_item_vii_int }
3609         {
3610             \int_eval:n
3611             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3612         }
3613         \int_set:Nn \l__enumext_joined_item_vii_int
3614         {
3615             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3616         }
3617     }
3618     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3619     {
3620         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3621         \int_decr:N \l__enumext_joined_item_aux_vii_int
3622         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3623         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3624         \dim_set:Nn \l__enumext_joined_width_vii_dim
3625         {
3626             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3627             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3628                 + \l__enumext_columns_sep_vii_dim

```

```

3629         )*\__enumext_joined_item_aux_vii_int
3630     }
3631     \dim_set_eq:NN \itemwidth \__enumext_joined_width_vii_dim
3632 }
3633 {
3634     \dim_set_eq:NN \__enumext_joined_width_vii_dim \__enumext_item_width_vii_dim
3635     \dim_set_eq:NN \itemwidth \__enumext_item_width_vii_dim
3636 }
3637 }
3638 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3639 {
3640     \int_set:Nn \__enumext_joined_item_viii_int {#1}
3641     \int_compare:nNnT { \__enumext_joined_item_viii_int } > { \__enumext_columns_viii_int }
3642     {
3643         \msg_warning:nnee { enumext } { item-joined }
3644         { \int_use:N \__enumext_joined_item_viii_int }
3645         { \int_use:N \__enumext_columns_viii_int }
3646         \int_set:Nn \__enumext_joined_item_viii_int
3647         {
3648             \__enumext_columns_viii_int - \__enumext_item_column_pos_viii_int + 1
3649         }
3650     }
3651     \int_compare:nNnT
3652     { \__enumext_joined_item_viii_int }
3653     >
3654     { \__enumext_columns_viii_int - \__enumext_item_column_pos_viii_int + 1 }
3655     {
3656         \msg_warning:nnee { enumext } { item-joined-columns }
3657         { \int_use:N \__enumext_joined_item_viii_int }
3658         {
3659             \int_eval:n
3660             { \__enumext_columns_viii_int - \__enumext_item_column_pos_viii_int + 1 }
3661         }
3662         \int_set:Nn \__enumext_joined_item_viii_int
3663         {
3664             \__enumext_columns_viii_int - \__enumext_item_column_pos_viii_int + 1
3665         }
3666     }
3667     \int_compare:nNnTF { \__enumext_joined_item_viii_int } > { 1 }
3668     {
3669         \int_set_eq:NN \__enumext_joined_item_aux_viii_int \__enumext_joined_item_viii_int
3670         \int_decr:N \__enumext_joined_item_aux_viii_int
3671         \int_add:Nn \__enumext_item_column_pos_viii_int { \__enumext_joined_item_aux_viii_int }
3672         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \__enumext_joined_item_aux_viii_int }
3673         \dim_set:Nn \__enumext_joined_width_viii_dim
3674         {
3675             \__enumext_item_width_viii_dim * \__enumext_joined_item_viii_int
3676             + ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim
3677               + \__enumext_columns_sep_viii_dim
3678             )*\__enumext_joined_item_aux_viii_int
3679         }
3680         \dim_set_eq:NN \itemwidth \__enumext_joined_width_viii_dim
3681     }
3682 }
3683 \dim_set_eq:NN \__enumext_joined_width_viii_dim \__enumext_item_width_viii_dim
3684 \dim_set_eq:NN \itemwidth \__enumext_item_width_viii_dim
3685 }
3686 }

```

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

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

`__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 `__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

3687 \cs_new_protected:Nn \__enumext_start_mini_vii:
3688 {
3689     \dim_compare:nNnT { \__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3690     {
3691         \dim_set:Nn \__enumext_minipage_left_vii_dim

```

```

3692     {
3693         \linewidth
3694         - \l__enumext_minipage_right_vii_dim
3695         - \l__enumext_minipage_hsep_vii_dim
3696     }
3697     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3698     \dim_gset_eq:NN
3699         \g__enumext_minipage_right_vii_dim
3700         \l__enumext_minipage_right_vii_dim
3701     \__enumext_mini_addvspace_vii:
3702     \nointerlineskip\noindent
3703     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3704 }
3705 }

```

The function `__enumext_stop_mini_vii`: closes the `__enumext_mini_env*` environment on the left side, applies `\hfill` and sets the value of the variable `\g__enumext_minipage_active_vii_bool` to true which will be used in the function `__enumext_after_env:nn` to execute the `__enumext_mini-env*` on the “right side”.

```

3706 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3707 {
3708     \bool_if:NT \l__enumext_minipage_active_vii_bool
3709     {
3710         \end{__enumext_mini_env*}
3711         \hfill
3712         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3713     }
3714 }

```

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “right side”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

3715 \__enumext_after_env:nn {enumext*}
3716 {
3717     \bool_if:NT \g__enumext_minipage_active_vii_bool
3718     {
3719         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3720         \par\addvspace { \g__enumext_minipage_right_skip }
3721         \bool_if:NF \g__enumext_minipage_center_vii_bool
3722         {
3723             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3724             {
3725                 \centering
3726             }
3727         }
3728         \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3729         {
3730             \tl_use:N \g__enumext_miniright_code_vii_tl
3731         }
3732         \box_use_drop:N \l__enumext_miniright_code_vii_box
3733         \end{__enumext_mini_env*}
3734         \par\addvspace{ \g__enumext_minipage_after_skip }
3735     }
3736     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3737     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3738     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3739     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3740     \bool_gset_false:N \g__enumext_starred_bool
3741 }

```

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

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

```

3742 \cs_new_protected:Nn \__enumext_start_mini_viii:
3743 {
3744     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3745     {
3746         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3747         {

```

```

3748         \linewidth
3749         - \l__enumext_minipage_right_viii_dim
3750         - \l__enumext_minipage_hsep_viii_dim
3751     }
3752     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3753     \dim_gset_eq:NN
3754         \g__enumext_minipage_right_viii_dim
3755         \l__enumext_minipage_right_viii_dim
3756     \__enumext_mini_addvspace_viii:
3757     \nointerlineskip\noindent
3758     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3759 }
3760 }
3761 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3762 {
3763     \bool_if:NT \l__enumext_minipage_active_viii_bool
3764     {
3765         \end{__enumext_mini_env*}
3766         \hfill
3767         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3768     }
3769 }
3770 \__enumext_after_env:nn {keyans*}
3771 {
3772     \bool_if:NT \g__enumext_minipage_active_viii_bool
3773     {
3774         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3775         \par\addvspace { \g__enumext_minipage_right_skip }
3776         \bool_if:NF \g__enumext_minipage_center_viii_bool
3777         {
3778             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3779             {
3780                 \centering
3781             }
3782         }
3783         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3784         {
3785             \tl_use:N \g__enumext_miniright_code_viii_tl
3786         }
3787         \box_use_drop:N \l__enumext_miniright_code_viii_box
3788         \end{__enumext_mini_env*}
3789         \par\addvspace{ \g__enumext_minipage_after_skip }
3790     }
3791     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3792     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3793     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3794     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3795 }

```

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

11.39 The environment enumext*

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

```

3796 \NewDocumentEnvironment{enumext*}{ o }
3797 {
3798     \__enumext_safe_exec_vii:
3799     \__enumext_parse_keys_vii:n {#1}
3800     \__enumext_before_list_vii:
3801     \__enumext_start_store_level_vii:
3802     \__enumext_start_list:nn { }
3803     {
3804         \__enumext_list_arg_two_vii:
3805         \__enumext_before_keys_exec_vii:
3806     }
3807     \__enumext_starred_columns_set_vii:
3808     \item[] \scan_stop:
3809     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3810     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3811 }

```

```

3812 {
3813     \__enumext_stop_item_tmp_vii:
3814     \__enumext_remove_extra_parsep_vii:
3815     \__enumext_stop_list:
3816     \__enumext_stop_store_level_vii:
3817     \__enumext_after_list_vii:
3818 }

```

(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`.

```

3819 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3820 {
3821     \__enumext_internal_mini_page:
3822     \__enumext_is_not_nested:
3823     \int_incr:N \l__enumext_level_h_int
3824     \int_compare:nNtT { \l__enumext_level_h_int } > { 1 }
3825     {
3826         \msg_error:nn { enumext } { nested }
3827     }
3828     \bool_set_true:N \l__enumext_starred_bool
3829     \__enumext_is_on_first_level:
3830 }

```

(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>`.

```

3831 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3832 {
3833     \tl_if_novalue:nF {#1}
3834     {
3835         \str_clear:N \l__enumext_series_str
3836         \keys_set:nn { enumext / enumext* } {#1}
3837         \__enumext_parse_series:n {#1}
3838         \__enumext_store_active_keys_vii:n {#1}
3839         \__enumext_nested_base_line_fix:
3840     }
3841 }

```

(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`.

```

3842 \cs_new_protected:Nn \__enumext_before_list_vii:
3843 {
3844     \__enumext_vspace_above_vii:
3845     \__enumext_check_ans_active:
3846     \__enumext_before_args_exec_vii:
3847     \__enumext_start_mini_vii:
3848 }

```

(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.

```

3849 \cs_new_protected:Nn \__enumext_after_list_vii:
3850 {
3851     \__enumext_stop_mini_vii:
3852     \__enumext_after_stop_list_vii:
3853     \__enumext_check_ans_key_hook:
3854     \__enumext_vspace_below_vii:
3855     \bool_set_false:N \l__enumext_starred_bool
3856     \__enumext_resume_save_counter:
3857 }

```

(End of definition for `__enumext_after_list_vii:`)

`__enumext_start_store_level_vii:`
`__enumext_stop_store_level_vii:`

The `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:` functions activate the level saving mechanism for storage in *(sequence)* of the `\anskey` command if `enumext*` are nested in `enumext`.

```

3858 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3859 {
3860   \bool_if:NT \l__enumext_store_active_bool
3861   {
3862     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3863     {
3864       \__enumext_store_level_open_vii:
3865     }
3866   }
3867 }
3868 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3869 {
3870   \bool_if:NT \l__enumext_store_active_bool
3871   {
3872     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3873     {
3874       \__enumext_store_level_close_vii:
3875     }
3876   }
3877 }

```

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

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

`__enumext_start_item_tmp_vii:`

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

```

3878 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3879 {
3880   \__enumext_stop_item_tmp_vii:
3881   \int_incr:N \l__enumext_item_column_pos_vii_int
3882   \int_gincr:N \g__enumext_item_count_all_vii_int
3883   \__enumext_item_peek_args_vii:
3884 }

```

(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).

```

3885 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3886 {
3887   \peek_meaning:NTF (
3888     { \__enumext_joined_item_vii:w }
3889     { \__enumext_joined_item_vii:w (1) }
3890   }

```

(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`.

```

3891 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3892 {
3893   \__enumext_starred_joined_item_vii:n {#1}
3894   \peek_meaning_remove:NTF *
3895   { \__enumext_starred_item_vii:w }
3896   { \__enumext_standar_item_vii:w }
3897 }

```

(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].

```

3898 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3899 {
3900   \bool_set_false:N \l__enumext_item_starred_vii_bool
3901   \peek_meaning:NTF [
3902     {
3903       \bool_set_eq:NN
3904         \l__enumext_wrap_label_vii_bool
3905         \l__enumext_wrap_label_opt_vii_bool
3906       \__enumext_start_item_vii:w
3907     }
3908     {
3909       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3910       \legacy_if_set_true:n { @noitemarg }
3911       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3912     }
3913   }

```

(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

```

3914 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3915 {
3916   \bool_set_true:N \l__enumext_item_starred_vii_bool
3917   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3918   \peek_meaning:NTF [
3919     { \__enumext_starred_item_vii_aux_i:w }
3920     { \__enumext_starred_item_vii_aux_ii:w }
3921   }
3922   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3923   {
3924     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3925     \__enumext_starred_item_vii_aux_ii:w
3926   }
3927   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3928   {
3929     \peek_meaning:NTF [
3930       { \__enumext_starred_item_vii_aux_iii:w }
3931       {
3932         \dim_set_eq:NN
3933           \l__enumext_item_symbol_sep_vii_dim
3934           \l__enumext_labelsep_vii_dim
3935         \legacy_if_set_true:n { @noitemarg }
3936         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3937       }
3938     }
3939   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3940   {
3941     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3942     \legacy_if_set_true:n { @noitemarg }
3943     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3944   }

```

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

11.39.2 Real definition of \item in enumext*

__enumext_start_item_vii:w

The functions __enumext_start_item_vii:w and __enumext_stop_item_vii: executing the true definition of `\item` inside the `enumext*` environment.

The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to __enumext_stop_item_vii: which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch

to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

3945 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3946 {
3947   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3948   \legacy_if:nT { @noitemarg }
3949   {
3950     \legacy_if_set_false:n { @noitemarg }
3951     \legacy_if:nT { @nmbrlist }
3952     {
3953       \bool_if:NT \__enumext_hyperref_bool
3954       {
3955         \legacy_if_set_true:n { @hyper@item }
3956       }
3957       \refstepcounter{enumXvii}
3958       \bool_if:NT \__enumext_check_answers_bool
3959       {
3960         \int_gincr:N \g__enumext_item_number_int
3961       }
3962     }
3963   }

```

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

```

3964   \group_begin:
3965   \lrbox{ \l__enumext_item_text_vii_box }
3966   \bool_if:NF \__enumext_footnotes_key_bool
3967   {
3968     \__enumext_renew_footnote:
3969   }
3970   \bool_if:NT \__enumext_item_starred_vii_bool
3971   {
3972     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3973     {
3974       \tl_gset_eq:NN
3975       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3976     }
3977     \mode_leave_vertical:
3978     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3979     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3980     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3981     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3982   }
3983   \group_begin:
3984   \tl_use:N \l__enumext_label_font_style_vii_tl
3985   \bool_if:NTF \__enumext_wrap_label_vii_bool
3986   {
3987     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3988     { \__enumext_wrapper_label_vii:n {#1} }
3989   }
3990   {
3991     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3992   }
3993   \group_end:
3994   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3995   \tl_use:N \l__enumext_after_list_args_vii_tl
3996   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3997   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3998   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3999   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4000 }

```

(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 $\langle contents \rangle$. 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.

```

4001 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4002 {
4003     \__enumext_endminipage:
4004     \endlrbox
4005     \group_end:
4006     \box_set_wd:Nn \l__enumext_item_text_vii_box
4007     {
4008         \l__enumext_joined_width_vii_dim
4009         + \l__enumext_labelwidth_vii_dim
4010         + \l__enumext_labelsep_vii_dim
4011     }
4012     \int_set:Nn \hbadness { 10000 }
4013     \box_use_drop:N \l__enumext_item_text_vii_box
4014     \bool_if:NF \l__enumext_footnotes_key_bool
4015     {
4016         \__enumext_print_footnote:
4017     }
4018     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4019     {
4020         \par\noindent
4021         \int_zero:N \l__enumext_item_column_pos_vii_int
4022     }
4023     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4024 }

```

(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.

```

4025 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4026 {
4027     \int_compare:nNnT
4028     {
4029         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4030     }
4031     =
4032     { 0 }
4033     {
4034         \par
4035         \vspace{ -\l__enumext_itemsep_vii_skip }
4036         \int_gzero:N \g__enumext_item_count_all_vii_int
4037     }
4038 }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`.)

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

```

4039 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

11.40 The environment `keyans*`

keyans*

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

```

4040 \NewDocumentEnvironment{keyans*}{ o }
4041 {
4042     \__enumext_safe_exec_viii:
4043     \__enumext_parse_keys_viii:n {#1}
4044     \__enumext_before_list_viii:
4045     \__enumext_start_list:nn { }
4046     {
4047         \__enumext_list_arg_two_viii:
4048         \__enumext_before_keys_exec_viii:
4049     }
4050     \__enumext_starred_columns_set_viii:
4051     \item[] \scan_stop:
4052     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4053     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4054 }

```

```

4055 {
4056   \__enumext_stop_item_tmp_viii:
4057   \__enumext_remove_extra_parsep_viii:
4058   \__enumext_check_starred_cmd:n { item }
4059   \__enumext_stop_list:
4060   \__enumext_after_list_viii:
4061 }

```

(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.

```

4062 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4063 {
4064   \int_incr:N \__enumext_keyans_level_h_int
4065   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4066   {
4067     \msg_error:nn { enumext } { nested }
4068   }
4069   \__enumext_keyans_start_line:
4070   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4071   \bool_set_false:N \__enumext_store_active_bool
4072   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4073   {
4074     \msg_error:nn { enumext } { keyans-wrong-level }
4075   }
4076 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

4077 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4078 {
4079   \tl_if_novalue:nF {#1}
4080   {
4081     \keys_set:nn { enumext / keyans* } {#1}
4082   }
4083 }

```

(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`.

```

4084 \cs_new_protected:Nn \__enumext_before_list_viii:
4085 {
4086   \__enumext_vspace_above_viii:
4087   \__enumext_before_args_exec_viii:
4088   \__enumext_start_mini_viii:
4089 }

```

(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.

```

4090 \cs_new_protected:Nn \__enumext_after_list_viii:
4091 {
4092   \__enumext_stop_mini_viii:
4093   \__enumext_after_stop_list_viii:
4094   \__enumext_vspace_below_viii:
4095 }

```

(End of definition for `__enumext_after_list_viii:`.)

11.40.1 The command `\item` in `keyans*`

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

`__enumext_start_item_tmp_viii:`

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

```
4096 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4097 {
4098   \__enumext_stop_item_tmp_viii:
4099   \int_incr:N \l__enumext_item_column_pos_viii_int
4100   \int_gincr:N \g__enumext_item_count_all_viii_int
4101   \__enumext_item_peek_args_viii:
4102 }
```

(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).

```
4103 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4104 {
4105   \peek_meaning:NTF (
4106     { \__enumext_joined_item_viii:w }
4107     { \__enumext_joined_item_viii:w (1) }
4108 }
```

(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`.

```
4109 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4110 {
4111   \__enumext_starred_joined_item_viii:n {#1}
4112   \peek_meaning_remove:NTF *
4113     { \__enumext_starred_item_viii:w }
4114     { \__enumext_standar_item_viii:w }
4115 }
```

(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]`.

```
4116 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4117 {
4118   \bool_set_false:N \l__enumext_item_starred_viii_bool
4119   \peek_meaning:NTF [
4120     {
4121       \bool_set_eq:NN
4122         \l__enumext_wrap_label_viii_bool
4123         \l__enumext_wrap_label_opt_viii_bool
4124       \__enumext_start_item_viii:w
4125     }
4126     {
4127       \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

4128         \legacy_if_set_true:n { @noitemarg }
4129         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4130     }
4131 }

```

(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].

```

4132 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4133 {
4134     \bool_set_true:N \__enumext_item_starred_viii_bool
4135     \bool_set_true:N \__enumext_wrap_label_viii_bool
4136     \peek_meaning:NTF [
4137         { \__enumext_starred_item_viii_aux_i:w }
4138         { \__enumext_starred_item_viii_aux_ii:w }
4139     }

```

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.

```

4140 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4141 {
4142     \tl_clear:N \l__enumext_store_current_label_tl
4143     \tl_if_no_value:nF { #1 }
4144     {
4145         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4146         {
4147             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4148             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4149         }
4150         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4151     }
4152     \__enumext_starred_item_viii_aux_ii:w
4153 }
4154 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4155 {
4156     \legacy_if_set_true:n { @noitemarg }
4157     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4158 }

```

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

```
\__enumext_starred_item_exec:
```

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

```

4159 \cs_new_protected:Nn \__enumext_starred_item_exec:
4160 {
4161     \tl_put_left:Ne \l__enumext_store_current_label_tl { \__enumext_label_viii_tl }
4162     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4163     \__enumext_keyans_store_ref:
4164     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4165     \__enumext_keyans_addto_seq_link:
4166     \int_gincr:N \g__enumext_check_starred_cmd_int
4167     \bool_if:NT \l__enumext_show_answer_bool
4168     {
4169         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4170     }
4171     \bool_if:NT \l__enumext_show_position_bool
4172     {
4173         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4174         {
4175             \group_begin:
4176             \exp_not:N \normalfont
4177             \exp_not:N \footnotesize [ \int_eval:n
4178                 {
4179                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4180                 }

```

```

4181         ]
4182     \group_end:
4183 }
4184 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
4185 }
4186 }

```

(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.

```

4187 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4188 {
4189     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4190     \legacy_if:nT { @noitemarg }
4191     {
4192         \legacy_if_set_false:n { @noitemarg }
4193         \legacy_if:nT { @nmbrlist }
4194         {
4195             \bool_if:NT \l__enumext_hyperref_bool
4196             {
4197                 \legacy_if_set_true:n { @hyper@item }
4198             }
4199             \refstepcounter{enumXviii}
4200         }
4201     }

```

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

```

4202     \group_begin:
4203     \lrbox{ \l__enumext_item_text_viii_box }
4204     \bool_if:NF \l__enumext_footnotes_key_bool
4205     {
4206         \__enumext_renew_footnote:
4207     }
4208     \bool_if:NT \l__enumext_item_starred_viii_bool
4209     {
4210         \__enumext_starred_item_exec:
4211     }
4212     \group_begin:
4213     \tl_use:N \l__enumext_label_font_style_viii_tl
4214     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4215     {
4216         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4217         { \__enumext_wrapper_label_viii:n {#1} }
4218     }
4219     {
4220         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4221     }
4222     \group_end:
4223     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4224     \tl_use:N \l__enumext_after_list_args_viii_tl
4225     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4226     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4227     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4228     \bool_if:NT \l__enumext_item_starred_viii_bool
4229     {
4230         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4231         \__enumext_keyans_show_item_opt:
4232         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim
4233     }
4234     {
4235         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4236     }
4237 }

```

(End of definition for `__enumext_start_item_viii:w`)

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.

```

4238 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:

```



```

4239 {
4240   \__enumext_endminipage:
4241   \endlrbox
4242   \group_end:
4243   \box_set_wd:Nn \l__enumext_item_text_viii_box
4244   {
4245     \l__enumext_joined_width_viii_dim
4246     + \l__enumext_labelwidth_viii_dim
4247     + \l__enumext_labelsep_viii_dim
4248   }
4249   \int_set:Nn \hbadness { 10000 }
4250   \box_use_drop:N \l__enumext_item_text_viii_box
4251   \bool_if:NF \l__enumext_footnotes_key_bool
4252   {
4253     \__enumext_print_footnote:
4254   }
4255   \int_compare:nNnTF
4256   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4257   {
4258     \par\noindent
4259     \int_zero:N \l__enumext_item_column_pos_viii_int
4260   }
4261   { \hspace{ \l__enumext_columns_sep_viii_dim } }
4262 }

```

(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.

```

4263 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4264 {
4265   \int_compare:nNnT
4266   {
4267     \int_mod:nn
4268     { \g__enumext_item_count_all_viii_int }
4269     { \l__enumext_columns_viii_int }
4270   }
4271   =
4272   { 0 }
4273   {
4274     \par
4275     \vspace{ -\l__enumext_itemsep_viii_skip }
4276     \int_gzero:N \g__enumext_item_count_all_viii_int
4277   }
4278 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

11.41 The command \getkeyans

\getkeyans

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

```

4279 \NewDocumentCommand \getkeyans { m }
4280 {
4281   \exp_args:Ne \__enumext_getkeyans_aux:n
4282   { \tl_to_str:e { \text_expand:n {#1} } }
4283 }

```

(End of definition for \getkeyans. This function is documented on page 15.)

__enumext_getkeyans_aux:n

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the $\langle \textit{argument} \rangle$ using “.”. If “.” is omitted it will return an error.

```

4284 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4285 {
4286   \str_if_in:nnTF {#1} { : }
4287   {
4288     \use:e
4289     {
4290       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4291       { {##1} {##2} }
4292     }

```

```

4293     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4294   }
4295   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4296 }

```

(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⟩*.

```

4297 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4298 {
4299   \prop_if_exist:cF { g__enumext_#1_prop }
4300   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4301   \group_begin:
4302     \prop_item:cn { g__enumext_#1_prop }{#2}
4303   \group_end:
4304 }

```

(End of definition for __enumext_getkeyans:nn.)

11.42 The command \printkeyans

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

```

4305 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4306 \keys_define:nn { enumext / print }
4307 {
4308   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4309                 { \__enumext_filter_save_key:n {#1} }
4310                 \__enumext_print_keyans_starred_tl, % starred cmd
4311   print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4312   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4313                 { \__enumext_filter_save_key:n {#1} }
4314                 \__enumext_print_keyans_i_tl,
4315   print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4316   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4317                 { \__enumext_filter_save_key:n {#1} }
4318                 \__enumext_print_keyans_ii_tl,
4319   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4320   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4321                 { \__enumext_filter_save_key:n {#1} }
4322                 \__enumext_print_keyans_iii_tl,
4323   print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4324   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4325                 { \__enumext_filter_save_key:n {#1} }
4326                 \__enumext_print_keyans_iv_tl,
4327   print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4328   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4329                 { \__enumext_filter_save_key:n {#1} }
4330                 \__enumext_print_keyans_vii_tl, % starred nested
4331   print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4332 }

```

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

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

```

4333 \NewDocumentCommand \printkeyans { s O{} m }
4334 {
4335   \group_begin:

```

```

4336 \tl_use:N \l__enumext_print_keyans_i_tl
4337 \tl_use:N \l__enumext_print_keyans_ii_tl
4338 \tl_use:N \l__enumext_print_keyans_iii_tl
4339 \tl_use:N \l__enumext_print_keyans_iv_tl
4340 \tl_use:N \l__enumext_print_keyans_vii_tl
4341 \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4342 \group_end:
4343 }

```

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

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

```

4344 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4345 {
4346   \seq_if_exist:cTF { g__enumext_#3_seq }
4347   {
4348     \seq_if_empty:cF { g__enumext_#3_seq }
4349     {
4350       %%\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 `<sequence>`, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default `<keys>` for the environment `enumext*`, it will open the environment `enumext*` passing the optional argument to the first level and then will map the `<sequence>`

```

4351   \bool_if:nTF {#1}
4352   {
4353     \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4354     {
4355       \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4356     }
4357     {
4358       \tl_use:N \l__enumext_print_keyans_starred_tl
4359       \begin{enumext*}[#2]
4360         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4361         \end{enumext*}
4362     }
4363   }

```

Otherwise it will open the environment `enumext` passing the optional argument to the first level and then map the `<sequence>`.

```

4364   {
4365     \begin{enumext}[#2]
4366     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4367     \end{enumext}
4368   }
4369 }
4370 }
4371 {
4372   \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4373 }
4374 }

```

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

11.43 The command `\setenumext`

First we define a “meta families” of `<keys>` to access from `\setenumext`.

```

4375 \keys_define:nn { enumext / meta-families }
4376 {
4377   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4378   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4379   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4380   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4381   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4382   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4383   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4384   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4385   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4386   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4387   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4388   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,

```

```

4389     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4390     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4391 }

```

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

```

4392 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4393 {
4394     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4395     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4396 }

```

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

```

4397 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4398 {
4399     \tl_if_novalue:nTF {#1}
4400     {
4401         \seq_map_inline:Nn \c__enumext_all_families_seq
4402     }
4403     {
4404         \seq_clear:N \l__enumext_setkey_tmpa_seq
4405         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4406         \int_set:Nn \l__enumext_setkey_tmpa_int
4407         {
4408             \seq_count:N \l__enumext_setkey_tmpb_seq
4409         }
4410         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4411         {
4412             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4413             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4414             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4415             {
4416                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4417             }
4418         }
4419         {
4420             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4421         }
4422         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4423         { \seq_map_inline:Nn \c__enumext_all_families_seq }
4424         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4425     }
4426     {
4427         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4428     }
4429 }

```

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

`__enumext_set_parse:n`
`__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```

4430 \cs_new_protected:Npn \__enumext_set_parse:n #1
4431 {
4432     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4433     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } %<- max level
4434     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4435     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4436     {
4437         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4438         { \tl_trim_spaces:n {#1} }
4439     }
4440     { \__enumext_set_error:nn {#1} { } }
4441 }
4442 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4443 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.44 Messages

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

```

4444 \msg_new:nnn { enumext } { package-load }
4445 {
4446   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4447 }
4448 \msg_new:nnn { enumext } { package-not-load }
4449 {
4450   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4451 }
4452 \msg_new:nnn { enumext } { package-load-foot }
4453 {
4454   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4455 }
```

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

```

4456 \msg_new:nnn { enumext } { counters }
4457 {
4458   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4459   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4460 }
```

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

```

4461 \msg_new:nnn { enumext } { unknown-choice }
4462 {
4463   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4464 }
4465 % \begin{macrocode}
4466 %
4467 % Message used by reserved \myenv*{anskey*} environment by \mypkg*{enumext} package.
4468 % \begin{macrocode}
4469 \msg_new:nnnn { enumext } { anskey-env-error }
4470 {
4471   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4472   'enumext' ~ package, ~ It~ is~ already~ defined.
4473 }
4474 {
4475   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4476   for ~ the ~ 'save-ans' ~ key.\\
4477 }
```

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

```

4478 \msg_new:nnn { enumext } { store-prop }
4479 {
4480   * ~ Package ~ enumext: ~ Creating ~
4481   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4482 }
4483 \msg_new:nnn { enumext } { store-seq }
4484 {
4485   * ~ Package ~ enumext: ~ Creating ~
4486   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4487 }
4488 \msg_new:nnn { enumext } { store-int }
4489 {
4490   * ~ Package ~ enumext: ~ Creating ~
4491   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4492 }
4493 \msg_new:nnn { enumext } { prop-seq-int-hook }
4494 {
4495   * ~ Package ~ enumext: ~ Elements ~ in ~
4496   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4497   * ~ Package ~ enumext: ~ Elements ~ in ~
4498   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4499   * ~ Package ~ enumext: ~ Value ~ off ~
4500   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4501 }
4502 \msg_new:nnn { enumext } { item-answer-hook }
4503 {
4504   * ~ Package ~ enumext: ~ Value ~ off ~
4505   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4506   * ~ Package ~ enumext: ~ Value ~ off ~
4507   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
```

```

4508     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4509 }

```

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

```

4510 \msg_new:nnn { enumext } { invalid-key }
4511 {
4512     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4513 }
4514 \msg_new:nnn { enumext } { unknown-key-family }
4515 {
4516     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4517 }

```

Messages used in length calculation.

```

4518 \msg_new:nnn { enumext } { width-negative }
4519 {
4520     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4521     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4522 }
4523 \msg_new:nnn { enumext } { width-zero }
4524 {
4525     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4526     The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4527 }

```

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

```

4528 \msg_new:nnn { enumext } { list-lengths }
4529 {
4530     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4531     \__enumext_show_length:nnn { dim } { labelsep } { #1}
4532     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
4533     \__enumext_show_length:nnn { dim } { itemindent } { #1}
4534     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
4535     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
4536     \__enumext_show_length:nnn { dim } { listparindent } { #1}
4537     \__enumext_show_length:nnn { skip } { topsep } { #1}
4538     \__enumext_show_length:nnn { skip } { parsep } { #1}
4539     \__enumext_show_length:nnn { skip } { partopsep } { #1}
4540     \__enumext_show_length:nnn { skip } { itemsep } { #1}
4541     ****
4542 }

```

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

```

4543 \msg_new:nnn { enumext } { list-lengths-not-nested }
4544 {
4545     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4546     \__enumext_show_length:nnn { dim } { labelsep } { #1}
4547     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
4548     \__enumext_show_length:nnn { dim } { itemindent } { #1}
4549     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
4550     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
4551     \__enumext_show_length:nnn { dim } { listparindent } { #1}
4552     \__enumext_show_length:nnn { skip } { topsep } { #1}
4553     \__enumext_show_length:nnn { skip } { parsep } { #1}
4554     \__enumext_show_length:nnn { skip } { partopsep } { #1}
4555     \__enumext_show_length:nnn { skip } { itemsep } { #1}
4556     ****
4557 }

```

Messages used by `ref` key.

```

4558 \msg_new:nnn { enumext } { key-ref-empty }
4559 {
4560     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4561 }

```

Messages used by `save-ans` key.

```

4562 \msg_new:nnn { enumext } { save-ans-empty }
4563 {
4564     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4565 }
4566 \msg_new:nnn { enumext } { save-ans-log }
4567 {
4568     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4569 }

```

```

4570 \msg_new:nnn { enumext } { save-ans-log-hook }
4571 {
4572   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4573 }
4574 \msg_new:nnn { enumext } { save-ans-hook }
4575 {
4576   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4577 }

```

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

```

4578 \msg_new:nnn { enumext } { need-save-ans }
4579 {
4580   Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4581 }
4582 \msg_new:nnn { enumext } { items-same-answer }
4583 {
4584   *****\\
4585   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4586   for ~ \c_left_brace_str #2 \c_right_brace_str\\
4587   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4588   'OK', ~ all ~ items ~ with ~ answer.\\
4589   *****
4590 }
4591 \msg_new:nnn { enumext } { item-greater-answer }
4592 {
4593   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4594   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4595   Items ~ > ~ Answers.
4596 }
4597 \msg_new:nnn { enumext } { item-less-answer }
4598 {
4599   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4600   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4601   Items ~ < ~ Answers.
4602 }

```

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

```

4603 \msg_new:nnn { enumext } { missing-starred }
4604 {
4605   Missing ~ '\c_backslash_str #1*' ~ #2.
4606 }
4607 \msg_new:nnn { enumext } { many-starred }
4608 {
4609   Many ~ '\c_backslash_str #1*' ~ #2.
4610 }

```

Messages used by `\printkeyans*` command.

```

4611 \msg_new:nnn { enumext } { print-starred }
4612 {
4613   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4614   #2 ~ environment ~ \msg_line_context:.
4615 }

```

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

```

4616 \msg_new:nnn { enumext } { list-too-deep }
4617 {
4618   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4619   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4620 }

```

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

```

4621 \msg_new:nnn { enumext } { anskey-empty-arg }
4622 {
4623   Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4624 }
4625 \msg_new:nnn { enumext } { anskey-wrong-place }
4626 {
4627   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4628   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4629 }
4630 \msg_new:nnn { enumext } { anskey-nested }
4631 {
4632   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.

```



```

4633     }
4634     \msg_new:nnn { enumext } { anskey-env-wrong }
4635     {
4636         The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4637     }
4638     \msg_new:nnn { enumext } { anspic-wrong-place }
4639     {
4640         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4641         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4642     }
4643     \msg_new:nnn { enumext } { command-wrong-place }
4644     {
4645         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4646         '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4647     }

```

Messages used by `keyans` and `keyanspic` environment.

```

4648     \msg_new:nnn { enumext } { keyans-nested }
4649     {
4650         The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4651     }
4652     \msg_new:nnn { enumext } { keyans-wrong-level }
4653     {
4654         Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4655         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4656     }
4657     \msg_new:nnn { enumext } { wrong-place }
4658     {
4659         Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4660         '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4661     }
4662     \msg_new:nnn { enumext } { keyanspic-nested }
4663     {
4664         The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4665     }
4666     \msg_new:nnn { enumext } { keyanspic-wrong-level }
4667     {
4668         Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4669         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4670     }

```

Messages used by `\getkeyans` command.

```

4671     \msg_new:nnn { enumext } { undefined-storage-anskey }
4672     {
4673         Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4674     }

```

Messages used by `\miniright` command.

```

4675     \msg_new:nnn { enumext } { missing-miniright }
4676     {
4677         Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4678         The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4679     }
4680     \msg_new:nnn { enumext } { wrong-miniright-place }
4681     {
4682         Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4683         Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4684     }
4685     \msg_new:nnn { enumext } { wrong-miniright-use }
4686     {
4687         Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4688         '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4689     }

```

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

```

4690     \msg_new:nnn { enumext } { nested }
4691     {
4692         The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4693     }
4694     \msg_new:nnn { enumext } { item-joined }
4695     {
4696         Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4697     }

```

```
4698 \msg_new:nnn { enumext } { item-joined-columns }  
4699 {  
4700   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4701 }
```

11.45 Finish package

Finish package implementation.

```
4702 \file_input_stop:  
4703 </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, 2574, 3455, 4458, 4471, 4476, 4496, 4498, 4505, 4507, 4520, 4525, 4530, 4545, 4584, 4586, 4588, 4593, 4594, 4599, 4600, 4618, 4627, 4640, 4645, 4654, 4659, 4668, 4677, 4682, 4687
A	
<code>above</code>	1419
<code>above*</code>	1419
<code>\addvspace</code>	1073, 1101, 1217, 1296, 1359, 1365, 1393, 1410, 3294, 3309, 3411, 3426, 3720, 3734, 3775, 3789
<code>after</code>	912
<code>align</code>	487
<code>\Alph</code>	35, 39, 40
<code>\Alph</code>	439, 554, 599, 667, 4327
<code>\alph</code>	35, 39, 40
<code>\alph</code>	440, 552, 4319
<code>\anskey</code>	12, 71, 2254, 2652, 2660
<code>anskey*</code>	13, 2520
<code>\anspic</code>	15, 96, 3433
<code>\anspic*</code>	66
<code>\arabic</code>	29, 35
<code>\arabic</code>	438, 551, 598, 4311, 4315, 4331
B	
<code>base-fix</code>	791
<code>\baselineskip</code>	48
<code>\baselineskip</code>	808, 819, 2198, 2207
<code>before</code>	912
<code>before*</code>	912
<code>below</code>	1419
<code>below*</code>	1419
bool commands:	
<code>\bool_gset_false:N</code>	314, 315, 316, 2670, 2672, 3736, 3740, 3791
<code>\bool_gset_true:N</code>	228, 237, 1015, 1911, 1917, 3712, 3737, 3767, 3792
<code>\bool_if:NTF</code>	379, 391, 408, 1441, 1455, 1468, 1479, 1490, 1501, 1512, 1523, 1576, 1593, 1598, 1606, 1633, 1671, 1676, 1683, 1687, 1709, 1714, 1722, 1729, 1760, 1768, 1861, 2002, 2071, 2081, 2160, 2184, 2191, 2220, 2258, 2268, 2300, 2326, 2443, 2454, 2458, 2619, 2699, 2714, 2789, 2800, 2804, 2917, 2947, 3021, 3037, 3099, 3109, 3139, 3144, 3228, 3278, 3292, 3300, 3339, 3396, 3409, 3417, 3435, 3708, 3717, 3721, 3763, 3772, 3776, 3860, 3870, 3953, 3958, 3966, 3970, 3985, 4014, 4167, 4171, 4195, 4204, 4208, 4214, 4228, 4251
<code>\bool_if:nTF</code>	1394, 1411, 2958, 2993, 3057, 3456, 4351
<code>\bool_if_p:N</code>	246, 260, 804, 805, 815, 816, 1740, 1741, 1749, 1750, 1874, 1896, 1908, 1909, 1914, 1915, 2311, 2352, 2353, 2377, 2386, 2387, 2399, 2415, 2605, 2776, 2777, 2814, 2815, 3201, 3214, 3216, 3463, 3464
<code>\bool_lazy_all:nTF</code>	244, 258, 1872, 1894, 2375, 2384, 2397, 2413, 3199, 3212
<code>\bool_lazy_and:nnTF</code>	224, 233, 803, 814, 1739, 1748, 1907, 1913, 2310, 2317, 2351, 2483, 2495, 2604, 2610, 2775
<code>\bool_lazy_or:nnTF</code>	1801, 1808, 2813, 3462
<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, 2312, 2318, 2402, 2417, 2606, 2611, 3202, 3203, 3215
<code>\bool_set_eq:NN</code>	2925, 2973, 3903, 4121
<code>\bool_set_false:N</code>	388, 825, 1846, 1847, 2557, 3315, 3347, 3429, 3494, 3512, 3855, 3900, 4071, 4118
<code>\bool_set_true:N</code>	251, 265, 370, 374, 480, 840, 1425, 1430, 1696, 1818, 1819, 2103, 2111, 2558, 2921, 2951, 2969, 2981, 3176, 3208, 3221, 3247, 3344, 3371, 3697, 3752, 3828, 3909, 3916, 3917, 4127, 4134, 4135
box commands:	
<code>\box_dp:N</code>	1113, 1117, 1121, 1132, 1136, 1147, 1156, 1162, 1172, 1185, 1191, 1197, 1228, 1229, 1230, 1233, 1243, 1247, 1256, 1263, 1268, 1276, 1305, 1306, 1309, 1316, 1329, 1337, 1343, 1351, 3524
<code>\box_new:N</code>	68, 165, 171
<code>\box_set_wd:Nn</code>	4006, 4243
<code>\box_use_drop:N</code>	3732, 3787, 4013, 4250
<code>\box_wd:N</code>	446
C	
<code>\c</code>	208, 209, 704, 706, 718, 720
<code>\catcode</code>	2574
<code>\cB</code>	209
<code>\cE</code>	209
<code>\centering</code>	1396, 1413, 3545, 3725, 3780
<code>check-ans</code>	1838
Document class:	
<code>article</code>	41
clist commands:	
<code>\clist_const:Nn</code>	177
<code>\clist_map_function:nN</code>	3532
<code>\clist_map_inline:Nn</code>	486, 746, 845, 911, 926, 1007, 1435
<code>\clist_map_inline:nn</code>	47, 58, 76, 82, 94, 106, 131, 153, 176, 514, 534, 800, 850, 1021, 1541, 1785, 1852, 2050, 2068, 2100, 2372, 2708, 2875, 3086, 3089, 3116, 3126, 3129, 3149, 4433
<code>\columnbreak</code>	73
<code>\columnbreak</code>	2314
<code>columns</code>	991
<code>columns-sep</code>	991
<code>\columnsep</code>	92, 95
<code>\columnsep</code>	3272, 3393
<code>\columnseprule</code>	92, 95
<code>\columnseprule</code>	3276, 3395
Commands provide by enumext:	
<code>\anskey</code>	27, 62, 63, 68, 69, 71, 74, 75, 80, 82, 91, 105, 113, 114, 119
<code>\anspic*</code>	27, 66, 69, 79–81, 96, 98, 99, 113, 114
<code>\anspic</code>	69, 96, 98, 119
<code>\getkeyans</code>	69, 113, 120
<code>\item*</code>	27, 66, 69, 79–81, 84, 85, 106, 111, 113, 114
<code>\itemwidth</code>	99, 100
<code>\item</code>	84, 85, 100, 105, 106, 110
<code>\miniright</code>	26, 46, 53, 54, 92, 93, 95, 96, 120
<code>\printkeyans*</code>	114

`\printkeyans` 27, 69, 114
`\setenumext` 27, 114–116, 118
 Counters defined by `enumext`:
`enumXiii` 25, 34
`enumXii` 25, 34
`enumXiv` 25, 34
`enumXi` 25, 34
`enumXviii` 25, 34
`enumXvii` 25, 34, 106
`enumXvi` 25, 34
`enumXv` 25, 34
 cs commands:
`\cs_generate_variant:Nn` 448, 464, 710, 726, 2152, 2157, 2238, 2525, 3076, 3534, 4305
`\cs_if_exist:NTF` 418
`\cs_if_free:NTF` 2487, 2499
`\cs_new:Nn` 194
`\cs_new:Npn` . 212, 1542, 1551, 1560, 2115, 2124, 2132
`\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, 801, 927, 931, 935, 939, 943, 947, 951, 955, 959, 963, 967, 971, 975, 979, 983, 987, 1022, 1034, 1058, 1075, 1086, 1103, 1178, 1202, 1219, 1281, 1298, 1320, 1355, 1361, 1436, 1450, 1464, 1475, 1486, 1497, 1508, 1519, 1604, 1707, 1720, 1737, 1758, 1786, 1791, 1816, 1857, 1867, 1905, 1920, 1927, 1936, 1941, 1946, 1951, 1960, 1965, 1970, 1992, 2158, 2182, 2189, 2218, 2225, 2266, 2363, 2474, 2526, 2547, 2578, 2602, 2644, 2668, 2697, 2712, 2740, 2773, 2809, 2821, 2829, 2880, 2884, 2903, 2954, 2989, 3005, 3015, 3031, 3169, 3197, 3226, 3233, 3256, 3286, 3298, 3337, 3361, 3379, 3404, 3415, 3452, 3496, 3510, 3530, 3535, 3551, 3570, 3687, 3706, 3742, 3761, 3819, 3842, 3849, 3858, 3868, 3885, 4025, 4062, 4084, 4090, 4103, 4159, 4263
`\cs_new_protected:Npn` 182, 186, 190, 401, 416, 433, 443, 449, 555, 600, 672, 697, 711, 1383, 1402, 1572, 1591, 1661, 1694, 1796, 1975, 2069, 2079, 2101, 2109, 2144, 2153, 2285, 2297, 2440, 2452, 2520, 2568, 2676, 2750, 2794, 2913, 2931, 2965, 2977, 3045, 3079, 3119, 3179, 3357, 3505, 3589, 3638, 3831, 3891, 3898, 3914, 3922, 3927, 3939, 4077, 4109, 4116, 4132, 4140, 4154, 4284, 4297, 4344, 4430, 4442
`\cs_new_protected_nopar:Nn` ... 3878, 4001, 4096, 4238
`\cs_new_protected_nopar:Npn` 3945, 4187
`\cs_set:Nn` 2445
`\cs_set:Npn` 2373, 2411, 4290
`\cs_set_eq:NN` . 3809, 3810, 3947, 4052, 4053, 4189
`\cs_set_protected:Nn` 851, 867, 879, 891
`\cs_set_protected:Npn` . 43, 52, 69, 77, 89, 95, 124, 149, 157, 465, 487, 519, 535, 582, 727, 747, 791, 827, 846, 903, 912, 991, 1008, 1419, 1530, 1777, 1838, 2012, 2051, 2087, 2365, 2701, 2864, 3077, 3117
`\cs_to_str:N` 435, 458
`\cs_undefine:N` 2476, 2477, 2478, 2479

D

`\d` 200
`\DeclareDocumentEnvironment` 354
 dim commands:
`\dim_abs:n` 3050, 3055
`\dim_add:Nn` 3515
`\dim_compare:nNTF` . 853, 869, 881, 893, 1385, 1404, 3047, 3052, 3058, 3064, 3066, 3068, 3238, 3261, 3365,

3383, 3507, 3553, 3572, 3689, 3744
`\dim_compare:nTF` 2336, 2632
`\dim_gset_eq:NN` 3698, 3753
`\dim_gzero:N` 2674, 3739, 3794
`\dim_new:N` 65, 72, 73, 74, 91, 133, 166, 167, 173
`\dim_set:Nn` .. 446, 841, 2945, 3050, 3055, 3057, 3060, 3061, 3065, 3067, 3070, 3071, 3073, 3241, 3264, 3367, 3385, 3537, 3555, 3562, 3574, 3581, 3624, 3673, 3691, 3746, 3941
`\dim_set_eq:NN` 542, 589, 660, 664, 2940, 3088, 3128, 3272, 3393, 3631, 3634, 3635, 3680, 3683, 3684, 3932
`\dim_use:N` 854, 862, 1386, 1392, 2228, 2231, 2236, 3010, 3012, 3239, 3244, 3245, 3252, 3262, 3266, 3267, 3269
`\dim_zero:N` 3276, 3395, 3516, 3517, 3518
`\dim_zero_new:N` 3568, 3587
`\c_zero_dim` 856, 870, 882, 894, 1386, 1404, 2338, 2634, 3047, 3052, 3058, 3065, 3239, 3262, 3365, 3383, 3553, 3572, 3689, 3744

E

`\end` .. 1389, 1407, 2186, 2222, 3291, 3308, 3408, 3425, 3710, 3733, 3765, 3788, 4353, 4361, 4367
`\endgroup` 2574
`\endlist` 32
`\endlist` 342
`\endlrbox` 4004, 4241
`\endminipage` 32
`\endminipage` 348
`enumext` 5, 3150
 enumext internal commands:
`\l__enumext__check_start_line_env_tl` ... 30
`\l__enumext__ref_the_count_tl` 37
`\l__enumext__resume_name_tl` 58
`__enumext_add_pre_parsep:` . 47, 1032, 1034, 1034
`__enumext_after_args_exec:` . 44, 927, 939, 3162
`__enumext_after_args_exec_v:` 45, 943, 955, 3330
`__enumext_after_args_exec_vii:` ... 959, 983
`__enumext_after_args_exec_viii:` 987
`__enumext_after_env:nn` . 66, 77, 78, 93, 102, 108, 186, 186, 2588, 3318, 3715, 3770, 4039
`__enumext_after_hyperref:` ... 33, 363, 365, 365
`__enumext_after_list:` . 93, 104, 109, 3167, 3298, 3298
`\l__enumext_after_list_args_v_tl` 957
`\l__enumext_after_list_args_vii_tl` 985, 3995
`\l__enumext_after_list_args_viii_tl` 989, 4224
`__enumext_after_list_v:` .. 96, 3335, 3415, 3415
`__enumext_after_list_vii:` ... 3817, 3849, 3849
`__enumext_after_list_viii:` .. 4060, 4090, 4090
`__enumext_after_stop_list:` ... 44, 45, 927, 935, 3313
`__enumext_after_stop_list_v:` 44, 943, 951, 3430
`\l__enumext_after_stop_list_v_tl` 953
`__enumext_after_stop_list_vii:` 959, 975, 3852
`\l__enumext_after_stop_list_vii_tl` ... 977
`__enumext_after_stop_list_viii:` . 979, 4093
`\l__enumext_after_stop_list_viii_tl` ... 981
`\l__enumext_align_label_vii_str` .. 3987, 3991
`\l__enumext_align_label_viii_str` . 4216, 4220
`\l__enumext_align_label_X_str` 157
`\c__enumext_all_envs_clist` .. 177, 486, 746, 845, 911, 926, 1007, 1435
`\c__enumext_all_families_seq` .. 116, 4392, 4401, 4423
`__enumext_anskey__env_keys:` 78

__enumext_anskey_env_clean: .. 79, 2598, 2602, 2668
 __enumext_anskey_env_define_keys: 77, 2520, 2526, 2582
 __enumext_anskey_env_exec: 78, 2523, 2578, 2578
 __enumext_anskey_env_keys: .. 2596, 2602, 2602
 __enumext_anskey_env_make:n 62, 77, 1821, 2520, 2520, 2525
 __enumext_anskey_env_store: .. 79, 2597, 2602, 2644
 __enumext_anskey_env_undefine_keys: . 77, 78, 2547, 2599
 __enumext_anskey_env_undefine_keys:__-enumext_rescan_anskey_env:n 2520
 \l__enumext_anskey_level_int .. 28, 2291, 2292
 __enumext_anskey_safe_inner:n .. 71, 72, 2261, 2266, 2285
 __enumext_anskey_safe_outer: . 71, 2256, 2266, 2266
 __enumext_anskey_show_wrap_arg:n . 75, 2440, 2440, 2456, 2471
 __enumext_anskey_show_wrap_left:n 75, 2304, 2452, 2452
 __enumext_anskey_wrapper:n 2016, 2450
 __enumext_at_begin_document:n .. 32, 182, 182, 339, 345
 \l__enumext_base_line_fix_bool . 795, 805, 816, 825
 __enumext_before_args_exec: 44, 927, 927, 3236
 __enumext_before_args_exec_v: 44, 45, 943, 943, 3364
 __enumext_before_args_exec_vii: .. 959, 959, 3846
 __enumext_before_args_exec_viii: 963, 4087
 __enumext_before_env:nn 77, 186, 190, 2481, 2493, 2505, 2580
 __enumext_before_keys_exec: 44, 927, 931, 3160
 __enumext_before_keys_exec_v: .. 44, 943, 947, 3328
 __enumext_before_keys_exec_vii 959
 __enumext_before_keys_exec_vii: 45, 967, 3805
 __enumext_before_keys_exec_viii: .. 45, 971, 4048
 __enumext_before_list: ... 91, 3154, 3233, 3233
 __enumext_before_list_v: . 94, 3323, 3361, 3361
 __enumext_before_list_vii: .. 104, 3800, 3842, 3842
 __enumext_before_list_viii: .. 109, 4044, 4084, 4084
 \l__enumext_before_no_starred_key_v_tl 949
 \l__enumext_before_no_starred_key_vii_tl 969
 \l__enumext_before_no_starred_key_viii_tl 973
 \l__enumext_before_starred_key_v_tl ... 945
 \l__enumext_before_starred_key_vii_tl . 961
 \l__enumext_before_starred_key_viii_tl 965
 __enumext_calc_hspace:NNNNNNN 87, 3045, 3045, 3076, 3081, 3121
 __enumext_check_ans_active: 63, 91, 1857, 1857, 3237, 3845
 \g__enumext_check_ans_item_tl 81
 \g__enumext_check_ans_key_bool 64, 65, 137, 314, 1911, 1917, 2002
 \l__enumext_check_ans_key_bool .. 64, 84, 1842, 1847, 1908, 1914
 __enumext_check_ans_key_hook: 64, 1905, 1905, 3312, 3853
 __enumext_check_ans_level: 63, 1857, 1863, 1867
 __enumext_check_ans_log: .. 64-66, 1951, 1951, 2006
 __enumext_check_ans_log_msg_greater: 1951, 1957, 1970
 __enumext_check_ans_log_msg_less: 1951, 1955, 1960
 __enumext_check_ans_log_msg_same_ok: 1951, 1956, 1965
 __enumext_check_ans_msg_greater: 1927, 1933, 1946
 __enumext_check_ans_msg_less: 1927, 1931, 1936
 __enumext_check_ans_msg_same_ok: 1927, 1932, 1941
 __enumext_check_ans_show: .. 64-66, 1927, 1927, 2004
 \g__enumext_check_ans_show_bool 93
 \l__enumext_check_answers_bool 62, 63, 71, 137, 1819, 1846, 1861, 2160, 2184, 2191, 2220, 2258, 2789, 2917, 2947, 3958
 __enumext_check_starred_cmd:n 30, 66, 81, 1975, 1975, 3333, 3491, 4058
 \g__enumext_check_starred_cmd_int 137, 1978, 1984, 1989, 2987, 3461, 4166
 \l__enumext_check_start_line_env_tl 137, 279, 286, 293, 1981, 1987, 1990
 \l__enumext_columns_sep_v_dim 3383, 3385, 3393
 \l__enumext_columns_sep_vii_dim .. 3553, 3555, 3564, 3628, 4023
 \l__enumext_columns_sep_viii_dim . 3572, 3574, 3583, 3677, 4261
 \l__enumext_columns_v_int 1224, 3381, 3389, 3401, 3406
 \l__enumext_columns_vii_int .. 3558, 3561, 3565, 3592, 3596, 3599, 3605, 3611, 3615, 4018, 4029
 \l__enumext_columns_viii_int . 3577, 3580, 3584, 3641, 3645, 3648, 3654, 3660, 3664, 4256, 4269
 \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 29, 48, 206
 \g__enumext_counter_styles_tl . 26, 35, 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 26, 65, 460, 543, 590, 661, 665
 __enumext_default_item:n ... 2913, 2913, 2962
 __enumext_define_counters:Nn 25, 416, 416, 425, 426, 427, 428, 429, 430, 431, 432
 __enumext_endminipage: . 32, 345, 348, 360, 3547, 4003, 4240
 \g__enumext_envir_name_tl 30, 34, 252, 266, 322, 1789, 1794, 1804, 1939, 1944, 1949, 1963, 1968, 1973
 __enumext_execute_after_env: 31, 32, 61, 64-66, 76, 1992, 1992, 3318, 4039
 __enumext_fake_item: 851, 851, 3108

`\l__enumext_fake_item_indent_v_dim` 870, 875
`\l__enumext_fake_item_indent_v_tl` 872, 2970, 2974, 2982
`\l__enumext_fake_item_indent_vii_dim` 882, 887
`\l__enumext_fake_item_indent_vii_tl` 884, 3999
`\l__enumext_fake_item_indent_viii_dim` . 894, 899, 4232
`\l__enumext_fake_item_indent_viii_tl` .. 896, 4230, 4235
`\l__enumext_fake_item_indent_X_tl` 95
`__enumext_fake_item_vii:` 851, 879, 3138
`__enumext_fake_item_viii:` 851, 891, 3143
`__enumext_filter_save_key:n` .. 69, 2076, 2084, 2107, 2113, 2115, 2115, 4309, 4313, 4317, 4321, 4325, 4329
`__enumext_filter_save_key_key:n` .. 69, 2115, 2120, 2124
`__enumext_filter_save_key_pair:nn` 69, 2115, 2121, 2132
`__enumext_filter_series:n` 57, 1542, 1542, 1584, 1596, 1601
`__enumext_filter_series_key:n` 57, 1542, 1547, 1551
`__enumext_filter_series_pair:nn` .. 57, 1542, 1548, 1560
`\g__enumext_footnote_arg_seq` . 154, 2886, 2899, 2909
`\g__enumext_footnote_int` . 154, 2893, 2896, 2898, 2900
`\g__enumext_footnote_int_seq` . 154, 2887, 2900, 2905, 2908
`__enumext_footnotes_key_bool` 33
`\l__enumext_footnotes_key_bool` 28, 33, 107, 144, 374, 379, 388, 3966, 4014, 4204, 4251
`__enumext_footnotetext:nn` ... 2880, 2880, 2910
`__enumext_getkeyans:nn` .. 114, 4293, 4297, 4297
`__enumext_getkeyans_aux:n` 113, 4281, 4284, 4284
`\l__enumext_hyperref_bool` 28, 33, 144, 370, 391, 408, 2353, 2777, 3953, 4195
`__enumext_hypertarget:nn` 33, 365, 393, 397, 413
`__enumext_if_is_int:n` 198
`__enumext_if_is_int:nTF` 198, 699, 713
`__enumext_internal_mini_page:` .. 32, 350, 350, 3171, 3821
`__enumext_is_not_nested:` . 25, 30, 90, 218, 218, 3172, 3822
`__enumext_is_on_first_level:` . 25, 30, 90, 218, 242, 3177, 3829
`\g__enumext_item_anskey_int` 71, 81, 137, 309, 336, 337, 1924, 2260, 2791
`__enumext_item_answer_diff:` 64–66, 1920, 1920, 1999
`\g__enumext_item_answer_diff_int` . 64, 65, 143, 310, 1922, 1929, 1953
`\l__enumext_item_column_pos_vii_int` 105, 3599, 3605, 3611, 3615, 3622, 3881, 4018, 4021
`\l__enumext_item_column_pos_viii_int` .. 110, 3648, 3654, 3660, 3664, 3671, 4099, 4256, 4259
`\l__enumext_item_column_pos_X_int` 157
`\g__enumext_item_count_all_vii_int` 105, 3623, 3882, 4029, 4036
`\g__enumext_item_count_all_viii_int` 110, 3672, 4100, 4268, 4276
`\g__enumext_item_count_all_X_int` 157
`\g__enumext_item_number_int` 63, 64, 137, 308, 335, 337, 1878, 1882, 1885, 1888, 1900, 1924, 2919, 2949, 3960
`__enumext_item_peek_args_vii:` 105, 3883, 3885, 3885
`__enumext_item_peek_args_viii:` .. 110, 4101, 4103, 4103
`__enumext_item_starred:` .. 86, 3005, 3005, 3023
`\l__enumext_item_starred_vii_bool` 3900, 3916, 3970
`\l__enumext_item_starred_viii_bool` 4118, 4134, 4208, 4228
`\l__enumext_item_starred_X_bool` 157
`__enumext_item_std:w` 32, 84–86, 98, 339, 343, 2922, 2928, 2952, 2970, 2974, 2982, 3528
`\g__enumext_item_symbol_aux_vii_tl` 3924, 3972, 3975, 3979, 3981
`\g__enumext_item_symbol_aux_X_tl` 157
`\l__enumext_item_symbol_sep_vii_dim` .. 3933, 3941, 3978, 3980
`\g__enumext_item_symbol_tl` ... 84, 59, 121, 2937, 3011, 3028
`\l__enumext_item_symbol_vii_tl` 3975
`\l__enumext_item_text_vii_box` 3965, 4006, 4013
`\l__enumext_item_text_viii_box` 4203, 4243, 4250
`\l__enumext_item_text_X_box` 157
`\l__enumext_item_width_vii_dim` ... 3562, 3626, 3634, 3635
`\l__enumext_item_width_viii_dim` .. 3581, 3675, 3683, 3684
`\l__enumext_item_width_X_dim` 157
`\l__enumext_itemindent_X_dim` 69
`\l__enumext_itemsep_vii_skip` 4035
`\l__enumext_itemsep_viii_skip` 4275
`\l__enumext_joined_item_aux_vii_int` .. 3620, 3621, 3622, 3623, 3629
`\l__enumext_joined_item_aux_viii_int` . 3669, 3670, 3671, 3672, 3678
`\l__enumext_joined_item_aux_X_int` 157
`__enumext_joined_item_vii:w` .. 105, 3888, 3889, 3891, 3891
`\l__enumext_joined_item_vii_int` .. 3591, 3592, 3595, 3597, 3603, 3608, 3613, 3618, 3620, 3626
`__enumext_joined_item_viii:w` . 110, 4106, 4107, 4109, 4109
`\l__enumext_joined_item_viii_int` . 3640, 3641, 3644, 3646, 3652, 3657, 3662, 3667, 3669, 3675
`\l__enumext_joined_item_X_int` 157
`\l__enumext_joined_width_vii_dim` . 3624, 3631, 3634, 3996, 4008
`\l__enumext_joined_width_viii_dim` 3673, 3680, 3683, 4225, 4245
`\l__enumext_joined_width_X_dim` 157
`__enumext_keyans_addto_prop:n` 79, 2676, 2676, 2984, 3458
`__enumext_keyans_addto_seq:n` . 81, 2750, 2750, 2986, 3460
`__enumext_keyans_addto_seq_link:` 2750, 2771, 2773, 4165
`__enumext_keyans_anspic_code:nnn` . 97, 3449, 3452, 3452
`__enumext_keyans_default_item:n` .. 85, 2965, 2965, 3001
`\l__enumext_keyans_env_bool` 34, 3202, 3215, 3344, 3429


```

\__enumext_keyans_fake_item: .. 851, 867, 3098
\l__enumext_keyans_level_h_int .. 28, 620, 647,
    2276, 2511, 2728, 4064, 4065
\l__enumext_keyans_level_int .. 28, 1377, 2272,
    2507, 2723, 3343, 3348, 3443
\__enumext_keyans_make_label: 35, 87, 3031, 3031,
    3096
\__enumext_keyans_mini_addvspace: 51, 95, 1281,
    1281, 3373
\__enumext_keyans_mini_right_cmd:n 54, 1379,
    1402, 1402
\__enumext_keyans_mini_set_vskip: . 50, 1219,
    1219, 1283
\__enumext_keyans_multi_addvspace: 95, 1075,
    1086, 3398
\__enumext_keyans_multi_set_vskip: 47, 1075,
    1075, 1088
\__enumext_keyans_multicols_start: 95, 3377,
    3379, 3379
\__enumext_keyans_multicols_stop: . 95, 1406,
    3404, 3404, 3428
\__enumext_keyans_parse_keys:n 3322, 3357, 3357
\l__enumext_keyans_pic_above_int . 132, 3538,
    3539, 3541
\l__enumext_keyans_pic_above_skip .. 98, 132,
    3482, 3522
\__enumext_keyans_pic_arg_two: 98, 3480, 3510,
    3510
\l__enumext_keyans_pic_below_int . 132, 3538,
    3539, 3542
\l__enumext_keyans_pic_body_seq .. 97-99, 132,
    3447, 3487, 3546
\__enumext_keyans_pic_do:n 99, 3487, 3489, 3530,
    3530, 3534
\l__enumext_keyans_pic_level_int .. 28, 1369,
    2280, 2515, 2679, 2718, 2753, 2831, 3498, 3499
\__enumext_keyans_pic_row:n 99, 3532, 3535, 3535
\__enumext_keyans_pic_safe_exec: .. 98, 3476,
    3496, 3496
\__enumext_keyans_pic_skip_abs:N .. 98, 3505,
    3505, 3521
\l__enumext_keyans_pic_width_dim . 132, 3537,
    3544
\__enumext_keyans_redefine_item: .. 86, 2989,
    2989, 3095
\__enumext_keyans_ref: ..... 39, 672, 690, 3097
\__enumext_keyans_ref:n ..... 39, 669, 672, 672
\__enumext_keyans_safe_exec: . 3321, 3337, 3337
\__enumext_keyans_show_ans: .. 2794, 2802, 2821
\__enumext_keyans_show_item_opt: . 2794, 2809,
    2982, 3472, 4231
\__enumext_keyans_show_left:n . 86, 2794, 2794,
    2980, 3467
\__enumext_keyans_show_pos: .. 2794, 2806, 2829
\__enumext_keyans_starred_item:n .. 86, 2977,
    2977, 2997
\__enumext_keyans_start_line: . 25, 30, 273, 273,
    3345, 3503, 4069
\__enumext_keyans_store_ref: .. 80, 2697, 2697,
    2985, 3459, 4163
\__enumext_keyans_store_ref_aux_i: 80, 2697,
    2709, 2712
\__enumext_keyans_store_ref_aux_ii: 81, 2697,
    2738, 2740
\__enumext_keyans_wrapper_opt:n .. 2019, 2817
\l__enumext_label_copy_i_tl .. 2407, 2716, 2721,
    2726, 2731
\l__enumext_label_copy_v_tl ..... 2726
\l__enumext_label_copy_vi_tl ..... 2721
\l__enumext_label_copy_vii_tl 2382, 2393, 2424,
    2716
\l__enumext_label_copy_viii_tl ..... 2731
\l__enumext_label_copy_X_tl ..... 146
\l__enumext_label_fill_left_v_tl ..... 3035
\l__enumext_label_fill_left_X_tl ..... 95
\l__enumext_label_fill_right_v_tl .... 3042
\l__enumext_label_fill_right_X_tl ..... 95
\l__enumext_label_font_style_v_tl 3036, 3471
\l__enumext_label_font_style_vii_tl ... 3984
\l__enumext_label_font_style_viii_tl .. 4213
\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 25, 35, 449, 449, 464,
    540, 587, 658, 662
\l__enumext_label_v_tl .. 79, 81, 655, 2684, 2758,
    2823, 2858, 2979, 2983, 3325, 3466, 3468
\l__enumext_label_vi_tl . 79, 81, 655, 2681, 2755,
    3466, 3468, 3472
\l__enumext_label_vii_tl . 582, 3911, 3936, 3943
\l__enumext_label_viii_tl 582, 4129, 4157, 4161
\l__enumext_label_width_by_box .. 65, 445, 446
\__enumext_label_width_by_box:Nn 35, 443, 443,
    448, 460, 723
\l__enumext_labelsep_i_dim ... 2826, 2861, 4169,
    4184
\l__enumext_labelsep_v_dim ..... 3388
\l__enumext_labelsep_vii_dim . 3557, 3566, 3627,
    3934, 3994, 4010
\l__enumext_labelsep_viii_dim 3576, 3585, 3676,
    4223, 4232, 4247
\l__enumext_labelwidth_i_dim . 2826, 2861, 4169,
    4184
\l__enumext_labelwidth_v_dim ..... 3388
\l__enumext_labelwidth_vii_dim ... 3557, 3565,
    3627, 3987, 3991, 4009
\l__enumext_labelwidth_viii_dim .. 3576, 3584,
    3676, 4216, 4220, 4246
\l__enumext_leftmargin_tmp_v_bool . 98, 3512
\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,
    854, 858, 862, 929, 933, 937, 941, 1024, 1026, 1028,
    1030, 1063, 1065, 1067, 1069, 1073, 1106, 1109, 1128,
    1137, 1143, 1148, 1152, 1163, 1167, 1168, 1173, 1209,
    1213, 1386, 1392, 1439, 1441, 1443, 1446, 1453, 1455,
    1457, 1460, 2071, 2073, 2075, 2103, 2104, 2106, 2162,
    2170, 2174, 2178, 2445, 2448, 2449, 2921, 2922, 2926,
    2927, 2928, 2935, 2937, 2941, 2942, 2945, 2951, 2952,
    3007, 3010, 3012, 3019, 3020, 3021, 3024, 3027, 3157,
    3159, 3208, 3221, 3228, 3239, 3241, 3244, 3245, 3247,
    3252, 3259, 3262, 3264, 3266, 3267, 3268, 3269, 3272,
    3278, 3283, 3289, 3292, 3294, 3300
\l__enumext_level_h_int .. 28, 226, 248, 261, 603,
    640, 1875, 1891, 2401, 2418, 2485, 2497, 3823, 3824
\l__enumext_level_int . 90, 28, 196, 235, 247, 262,

```


352, 1036, 1180, 1373, 1869, 1897, 1994, 2378, 2388,
 2394, 2400, 2408, 2416, 2423, 2484, 2496, 3111, 3173,
 3174, 3184, 3192, 3206, 3219, 3274, 3352, 3439, 3862,
 3872, 4072
 __enumext_list_arg_two_i: 3077
 __enumext_list_arg_two_ii: 3077
 __enumext_list_arg_two_iii: 3077
 __enumext_list_arg_two_iv: 3077
 __enumext_list_arg_two_v: . 86, 3077, 3327, 3513
 __enumext_list_arg_two_vii: 3117, 3804
 __enumext_list_arg_two_viii: . . . 3117, 4047
 \l__enumext_listoffset_v_dim 3390
 \l__enumext_listparindent_vii_dim . . . 3997
 \l__enumext_listparindent_viii_dim . . 4226
 __enumext_log_answer_vars: . 32, 324, 332, 2001
 __enumext_log_global_vars: . 31, 324, 324, 2000
 __enumext_make_label: 35, 84, 85, 87, 3015, 3015,
 3106
 \l__enumext_mark_answer_sym_tl 71, 2025, 2233,
 2460, 2833, 2846, 4173
 \l__enumext_mark_position_str 121, 2029, 2030,
 2056, 2057, 2231
 \l__enumext_mark_ref_sym_tl . . 2042, 2358, 2785
 __enumext_mini_addvspace: . . 50, 92, 1202, 1202,
 3249
 __enumext_mini_addvspace_vii: 53, 1355, 1355,
 3701
 __enumext_mini_addvspace_viii: 53, 1355, 1361,
 3756
 __enumext_mini_env* 350
 __enumext_mini_right_cmd:n . 53, 54, 1381, 1383,
 1383
 __enumext_mini_set_vskip: . 48, 1103, 1103, 1204
 __enumext_mini_set_vskip_vii: 52, 1298, 1298,
 1357
 __enumext_mini_set_vskip_viii: 52, 1298, 1320,
 1363
 __enumext_minipage:w 32, 345, 347, 356, 3544, 3996,
 4225
 \l__enumext_minipage_active_v_bool . . 95, 96,
 3371, 3396, 3409, 3417
 \g__enumext_minipage_active_vii_bool . . 102,
 3712, 3717, 3736
 \l__enumext_minipage_active_vii_bool . 3697,
 3708
 \g__enumext_minipage_active_viii_bool 3767,
 3772, 3791
 \l__enumext_minipage_active_viii_bool 3752,
 3763
 \g__enumext_minipage_active_X_bool . . 157
 \l__enumext_minipage_active_X_bool . . . 83
 \g__enumext_minipage_after_skip 83, 1302, 1314,
 3734, 3789
 \l__enumext_minipage_after_skip 48, 50, 93, 96,
 83, 1119, 1134, 1154, 1170, 1185, 1191, 1197, 1211,
 1221, 1230, 1233, 1245, 1263, 1274, 1290, 1322, 1335,
 1349, 3309, 3426
 \g__enumext_minipage_center_vii_bool . 3721,
 3737
 \g__enumext_minipage_center_viii_bool 3776,
 3792
 \g__enumext_minipage_center_X_bool . . 157
 \l__enumext_minipage_hsep_v_dim . . . 94, 3369
 \l__enumext_minipage_hsep_vii_dim . . . 3695
 \l__enumext_minipage_hsep_viii_dim . . . 3750
 \l__enumext_minipage_left_skip 48, 95, 83, 1111,
 1126, 1145, 1160, 1207, 1217, 1222, 1228, 1237, 1254,
 1266, 1286, 1296, 1300, 1305, 1309, 1323, 1327, 1341,
 1359, 1365
 \l__enumext_minipage_left_v_dim 94, 3367, 3375
 \l__enumext_minipage_left_vii_dim 3691, 3703
 \l__enumext_minipage_left_viii_dim 3746, 3758
 \l__enumext_minipage_left_X_dim 83
 \g__enumext_minipage_right_skip 83, 1301, 1306,
 1310, 3720, 3775
 \l__enumext_minipage_right_skip . 48, 83, 1115,
 1130, 1150, 1165, 1223, 1229, 1241, 1259, 1270, 1324,
 1331, 1345, 1393, 1410
 \l__enumext_minipage_right_v_dim . . 94, 1404,
 1409, 3365, 3369
 \g__enumext_minipage_right_vii_dim 101, 3699,
 3719, 3739
 \l__enumext_minipage_right_vii_dim 101, 3689,
 3694, 3700
 \g__enumext_minipage_right_viii_dim . . 3754,
 3774, 3794
 \l__enumext_minipage_right_viii_dim . . 3744,
 3749, 3755
 \g__enumext_minipage_right_X_dim 157
 \g__enumext_minipage_right_X_skip . . . 157
 \g__enumext_minipage_stat_int . 92, 95, 83, 1398,
 1415, 3248, 3302, 3307, 3372, 3419, 3424
 \l__enumext_miniright_code_vii_box 3728, 3732
 \g__enumext_miniright_code_vii_tl 102, 3723,
 3730, 3738
 \l__enumext_miniright_code_viii_box . . 3783,
 3787
 \g__enumext_miniright_code_viii_tl 3778, 3785,
 3793
 \l__enumext_miniright_code_X_box 157
 __enumext_multi_addvspace: . 47, 92, 1058, 1058,
 3280
 __enumext_multi_set_vskip: 46, 1022, 1022, 1060
 \l__enumext_multicols_above_ii_skip . . 1041
 \l__enumext_multicols_above_iii_skip . . 1047
 \l__enumext_multicols_above_iv_skip . . 1053
 \l__enumext_multicols_above_v_skip 1077, 1091,
 1101
 \l__enumext_multicols_above_X_skip . . . 77
 \l__enumext_multicols_below_v_skip 1081, 1095,
 3411
 \l__enumext_multicols_below_X_skip . . . 77
 __enumext_multicols_start: 92, 3254, 3256, 3256
 __enumext_multicols_stop: 93, 1388, 3286, 3286,
 3311
 __enumext_nested_base_line_fix: 41, 791, 801,
 3188, 3839
 __enumext_newlabel:nn 28, 33, 75, 401, 401, 2434,
 2744
 \l__enumext_newlabel_arg_one_tl 28, 33, 75, 80,
 146, 2357, 2427, 2435, 2733, 2745, 2783
 \l__enumext_newlabel_arg_two_tl 28, 33, 74, 146,
 2381, 2391, 2405, 2421, 2436, 2720, 2725, 2730, 2746
 __enumext_parse_keys:n 41, 57, 3153, 3179, 3179
 __enumext_parse_keys_vii:n . 41, 57, 3799, 3831,
 3831
 __enumext_parse_keys_viii:n . 4043, 4077, 4077
 __enumext_parse_save_key:n 68, 2096, 2101, 2101

```

\__enumext_parse_save_key_vii:n 68, 2091, 2101,
    2109
\__enumext_parse_serie:n ..... 104
\__enumext_parse_series:n .. 57, 90, 1572, 1572,
    3187, 3837
\__enumext_parse_store_keys:n ..... 90
\l__enumext_parsep_i_skip 1039, 1041, 1183, 1231
\l__enumext_parsep_ii_skip ... 1045, 1047, 1189
\l__enumext_parsep_iii_skip .. 1051, 1053, 1195
\l__enumext_parsep_vii_skip ..... 3998
\l__enumext_parsep_viii_skip ..... 4227
\l__enumext_partopsep_v_skip . 1093, 1097, 1257,
    1261, 1268, 1272, 1288, 1292
\l__enumext_partopsep_viii_skip ..... 1333
\__enumext_phantomsection: 33, 365, 394, 398, 414
\__enumext_print_footnote: ... 2880, 2903, 4016,
    4253
\__enumext_print_keyans_box:NN 71, 2225, 2225,
    2238, 2447, 2825, 2860, 4169, 4184
\l__enumext_print_keyans_i_tl .... 4314, 4336
\l__enumext_print_keyans_ii_tl ... 4318, 4337
\l__enumext_print_keyans_iii_tl .. 4322, 4338
\l__enumext_print_keyans_iv_tl ... 4326, 4339
\l__enumext_print_keyans_starred_tl 114, 115,
    121, 4310, 4358
\l__enumext_print_keyans_vii_tl 114, 4330, 4340
\l__enumext_print_keyans_X_tl ..... 121
\__enumext_printkeyans:nnn 114, 115, 4341, 4344,
    4344
\__enumext_redefine_item: . 85, 2954, 2954, 3105
\l__enumext_ref_key_arg_tl 37, 48, 209, 557, 558,
    571, 602, 605, 616, 622, 633, 674, 675, 686
\l__enumext_ref_the_count_tl . 37, 48, 564, 567,
    570, 610, 612, 615, 627, 629, 632, 680, 682, 685
\__enumext_regex_counter_style: .. 29, 37, 204,
    204, 565, 611, 628, 681
\__enumext_register_counter_style:Nn .. 433,
    433, 438, 439, 440, 441, 442
\__enumext_remove_extra_parsep_vii: .. 3814,
    4025, 4025
\__enumext_remove_extra_parsep_viii: . 4057,
    4263, 4263
\__enumext_renew_footnote: ... 2880, 2884, 3968,
    4206
\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 .. 77, 79, 2568,
    2654, 2662
\__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: . 31, 66, 300, 300,
    2009
\l__enumext_resume_active_bool 57, 59, 59, 1576,
    1696
\__enumext_resume_counter: .. 59, 60, 1694, 1700,
    1707
\__enumext_resume_counter:n . 57, 59, 1665, 1670,
    1694, 1694, 1764, 1772
\__enumext_resume_counter_save_ans: 60, 1694,
    1705, 1737
\__enumext_resume_counter_series: . 60, 1694,
    1703, 1720
\g__enumext_resume_int ... 59, 1617, 1711, 1712
\__enumext_resume_last:n 57, 58, 1572, 1578, 1591
\l__enumext_resume_name_tl 59, 1613, 1621, 1624,
    1640, 1648, 1651, 1697, 1698, 1726, 1733
\__enumext_resume_save_counter: 58, 1604, 1604,
    3316, 3856
\__enumext_resume_series:n . 59, 1536, 1661, 1661
\__enumext_resume_starred: . 61, 1537, 1758, 1758
\g__enumext_resume_vii_int . 104, 59, 1644, 1716,
    1717
\__enumext_safe_exec: .. 32, 90, 3152, 3169, 3169
\__enumext_safe_exec_vii: . 32, 3798, 3819, 3819
\__enumext_safe_exec_viii: ... 4042, 4062, 4062
\l__enumext_series_name_tl ..... 59
\l__enumext_series_str . 58, 90, 1534, 1574, 1582,
    1583, 1585, 1587, 1608, 1611, 1615, 1635, 1638, 1642,
    3183, 3835
\__enumext_set_error:nn ..... 4430, 4440, 4442
\__enumext_set_parse:n ..... 4413, 4430, 4430
\l__enumext_setkey_tmpa_int ... 116, 4406, 4410
\l__enumext_setkey_tmpa_seq .. 116, 4404, 4414,
    4420, 4422, 4424, 4437
\l__enumext_setkey_tmpa_tl .... 116, 4412, 4416
\l__enumext_setkey_tmpb_seq .. 116, 4405, 4408,
    4412, 4413
\l__enumext_setkey_tmpb_tl 116, 4432, 4434, 4435
\l__enumext_show_answer_bool . 2036, 2060, 2454,
    2800, 2814, 3463, 4167
\__enumext_show_length:nnn .. 43, 212, 212, 4531,
    4532, 4533, 4534, 4535, 4536, 4537, 4538, 4539, 4540,
    4546, 4547, 4548, 4549, 4550, 4551, 4552, 4553, 4554,
    4555
\l__enumext_show_position_bool ... 2039, 2063,
    2458, 2804, 2815, 3464, 4171
\g__enumext_standar_bool 30, 90, 34, 225, 228, 246,
    315, 1606, 1671, 1683, 1709, 1722, 1760, 1896, 1909
\l__enumext_standar_bool . 90, 93, 34, 2386, 2399,
    2415, 3176, 3315
\l__enumext_standar_first_bool 30, 90, 34, 251,
    804, 1593, 1740, 1802, 1809
\__enumext_standar_item_vii:w . 105, 106, 3896,
    3898, 3898
\__enumext_standar_item_viii:w 110, 4114, 4116,
    4116
\__enumext_standar_ref: .... 37, 555, 575, 3107
\__enumext_standar_ref:n .... 37, 547, 555, 555
\g__enumext_standar_series_tl . 59, 1595, 1596,
    1762, 1765
\g__enumext_starred_bool 30, 104, 34, 234, 237, 260,
    316, 1633, 1676, 1687, 1714, 1729, 1768, 1874, 1915,
    2377, 2387, 2417, 2714, 3203, 3216, 3740
\l__enumext_starred_bool .. 104, 34, 2312, 2318,
    2402, 2443, 2606, 2611, 3828, 3855
\__enumext_starred_columns_set_vii: .. 3551,
    3551, 3807
\__enumext_starred_columns_set_viii: . 3551,
    3570, 4050
\l__enumext_starred_first_bool 30, 34, 265, 815,
    1598, 1749, 1802, 1809
\__enumext_starred_item:nn ... 2931, 2931, 2960
\__enumext_starred_item_exec: . 111, 4159, 4159,
    4210

```

__enumext_starred_item_vii:w . 105, 106, 3895, 3914, 3914
 __enumext_starred_item_vii_aux_i:w . 3914, 3919, 3922
 __enumext_starred_item_vii_aux_ii:w . 3914, 3920, 3925, 3927
 __enumext_starred_item_vii_aux_iii:w 3914, 3930, 3939
 __enumext_starred_item_viii:w 110, 111, 4113, 4132, 4132
 __enumext_starred_item_viii_aux_i:w . 111, 4132, 4137, 4140
 __enumext_starred_item_viii_aux_ii:w . 111, 4132, 4138, 4152, 4154
 __enumext_starred_joined_item_vii:n 100, 105, 3589, 3589, 3893
 __enumext_starred_joined_item_viii:n . 100, 110, 3589, 3638, 4111
 __enumext_starred_ref: 38, 600, 638, 3135
 __enumext_starred_ref:n 38, 594, 600, 600
 \g__enumext_starred_series_tl . 59, 1600, 1601, 1770, 1773
 __enumext_start_from:NNn 39, 697, 697, 710, 732
 \l__enumext_start_i_int 1712, 1724, 1743
 __enumext_start_item_tmp_vii: 103, 3810, 3878, 3878
 __enumext_start_item_tmp_viii: . . 108, 4053, 4096, 4096
 __enumext_start_item_vii:w . . . 106, 3906, 3911, 3936, 3943, 3945, 3945
 __enumext_start_item_viii:w . . 110, 4124, 4129, 4157, 4187, 4187
 \g__enumext_start_line_tl 30, 34, 253, 267, 321, 1939, 1944, 1949, 1963, 1968, 1973
 __enumext_start_list:nn 32, 87, 98, 339, 341, 3156, 3324, 3477, 3802, 4045
 __enumext_start_mini_vii: 104, 3687, 3687, 3847
 __enumext_start_mini_viii: . . 109, 3742, 3742, 4088
 __enumext_start_save_ans_msg: . . 61, 62, 1786, 1786, 1811
 __enumext_start_store_level: . 91, 3155, 3197, 3197
 __enumext_start_store_level_vii: 105, 3801, 3858, 3858
 \l__enumext_start_vii_int . . . 1717, 1731, 1752
 \l__enumext_start_X_int 95, 727
 __enumext_stop_item_tmp_vii: . . 103, 105, 106, 3809, 3813, 3880, 3947
 __enumext_stop_item_tmp_viii: 108, 110, 4052, 4056, 4098, 4189
 __enumext_stop_item_vii: 106, 107, 3947, 4001, 4001
 __enumext_stop_item_viii: 112, 4189, 4238, 4238
 __enumext_stop_list: . . 32, 339, 342, 3165, 3334, 3490, 3815, 4059
 __enumext_stop_mini_vii: 102, 104, 3687, 3706, 3851
 __enumext_stop_mini_viii: 109, 3742, 3761, 4092
 __enumext_stop_save_ans_msg: . 61, 1786, 1791, 1998
 __enumext_stop_store_level: . . 91, 3166, 3197, 3226
 __enumext_stop_store_level_vii: . 105, 3816, 3858, 3868
 \l__enumext_store_active_bool . 27, 62, 90, 104, 107, 1741, 1750, 1818, 2268, 3201, 3214, 3339, 3347, 3435, 3494, 3860, 3870, 4071
 __enumext_store_active_keys:n . . 67, 68, 2069, 2069, 3194
 __enumext_store_active_keys_vii:n 67, 68, 104, 2069, 2079, 3838
 __enumext_store_addto_prop:n 69, 80, 2144, 2144, 2152, 2299, 2695, 4162
 __enumext_store_addto_seq:n 69, 81, 2153, 2153, 2157, 2164, 2178, 2186, 2195, 2214, 2222, 2361, 2788
 \l__enumext_store_anskey_arg_tl . . 27, 73, 107, 2309, 2314, 2316, 2321, 2328, 2331, 2341, 2346, 2349, 2355, 2361
 __enumext_store_anskey_code:nn 72, 2262, 2297, 2297
 \l__enumext_store_anskey_env_tl 27, 78, 79, 107, 2590, 2592, 2647, 2654, 2662
 \l__enumext_store_anskey_opt_tl . . 27, 78, 107, 2591, 2608, 2614, 2621, 2627, 2637, 2649, 2658
 __enumext_store_anskey_safe_outer: 72
 \g__enumext_store_columns_break_bool . 2530, 2605, 2670
 \l__enumext_store_columns_break_bool . 2241, 2311
 \l__enumext_store_current_label_tl 27, 80, 81, 111, 107, 2678, 2681, 2684, 2691, 2693, 2695, 2752, 2755, 2758, 2764, 2769, 2779, 2788, 4142, 4147, 4148, 4161, 4162, 4164
 \l__enumext_store_current_label_tmp_tl . 27, 107, 2979, 2983
 \l__enumext_store_current_opt_arg_tl 27, 111, 107, 2798, 2811, 2817, 4150
 __enumext_store_internal_ref: . . 72, 74, 2302, 2363, 2363
 \g__enumext_store_item_join_int . . 2533, 2612, 2616, 2671
 \l__enumext_store_item_join_int . . 2244, 2319, 2323
 \g__enumext_store_item_star_bool . 2535, 2619, 2672
 \l__enumext_store_item_star_bool . 2246, 2326
 \g__enumext_store_item_symbol_sep_dim 2540, 2634, 2639, 2674
 \l__enumext_store_item_symbol_sep_dim 2251, 2338, 2343
 \g__enumext_store_item_symbol_tl . 2538, 2625, 2629, 2673
 \l__enumext_store_item_symbol_tl . 2249, 2329, 2333
 \l__enumext_store_keyans_item_opt_sep_tl 2022, 2689, 2691, 2762, 2766, 4145, 4147
 __enumext_store_level_close: . 70, 2158, 2182, 3230
 __enumext_store_level_close_vii: 2189, 2218, 3874
 __enumext_store_level_open: . . 70, 2158, 2158, 3209, 3222
 __enumext_store_level_open_vii: . 2189, 2189, 3864
 \g__enumext_store_name_tl . 27, 62, 93, 107, 320, 327, 328, 329, 330, 1794, 1820, 1938, 1943, 1948, 1962, 1967, 1972, 1996
 \l__enumext_store_name_tl 27, 62, 63, 107, 1627,

1630, 1654, 1657, 1745, 1754, 1789, 1798, 1799, 1820, 1821, 1822, 1824, 1825, 1827, 1829, 1830, 1832, 1834, 1835, 1859, 2146, 2148, 2155, 2429, 2430, 2466, 2594, 2735, 2736, 2839, 2852, 4179

`\l__enumext_store_ref_key_bool` 72, 2045, 2300, 2352, 2699, 2776

`\l__enumext_store_save_key_vii_bool` .. 2081, 2111

`\l__enumext_store_save_key_vii_tl` 2083, 2084, 2112, 2113, 2193, 2204, 2210, 2214

`\l__enumext_store_save_key_X_bool` .. 67, 121

`\l__enumext_store_save_key_X_tl` .. 67, 68, 121

`\l__enumext_store_upper_level_X_bool` .. 121

`__enumext_storing_exec:` 62, 77, 1796, 1812, 1816

`__enumext_storing_set:n` 61, 62, 1781, 1796, 1796

`\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, 791, 800, 846, 850, 1530, 1541, 1777, 1785, 1838, 1856, 2012, 2050, 2051, 2068, 2087, 2100, 2365, 2372, 2373, 2394, 2408, 2411, 2423, 2701, 2708, 3077, 3116, 3117, 3149

`__enumext_tmp:nn` 465, 486, 487, 518, 519, 534, 727, 746, 827, 845, 903, 911, 912, 926, 991, 1007, 1008, 1021, 1419, 1435, 2864, 2879

`__enumext_tmp:nnn` 535, 551, 552, 553, 554, 582, 598, 599

`__enumext_tmp:nnnnnn` 747, 772, 775, 778, 780, 782, 785, 788

`__enumext_tmp:w` 4290, 4293

`\l__enumext_tmpa_vii_int` 3561, 3564

`\l__enumext_tmpa_viii_int` 3580, 3583

`\l__enumext_tmpa_X_int` 157

`\l__enumext_topsep_v_skip` 1079, 1083, 1226, 1239, 1247, 1252, 1272, 1276, 3493, 3525

`\l__enumext_topsep_vii_skip` .. 1303, 1312, 1316

`\l__enumext_topsep_viii_skip` . 1325, 1347, 1351

`__enumext_undefine_anskey_env:` . 66, 76, 2007, 2474, 2474

`\l__enumext_vspace_a_star_v_bool` 1468

`\l__enumext_vspace_a_star_vii_bool` ... 1490

`\l__enumext_vspace_a_star_viii_bool` ... 1501

`\l__enumext_vspace_a_star_X_bool` 95

`__enumext_vspace_above:` .. 54, 1436, 1436, 3235

`__enumext_vspace_above_v:` . 55, 1464, 1464, 3363

`\l__enumext_vspace_above_v_skip` .. 1466, 1470, 1472

`__enumext_vspace_above_vii:` .. 55, 1486, 1486, 3844

`\l__enumext_vspace_above_vii_skip` 1488, 1492, 1494

`__enumext_vspace_above_viii:` . 55, 1486, 1497, 4086

`\l__enumext_vspace_above_viii_skip` 1499, 1503, 1505

`\l__enumext_vspace_b_star_v_bool` 1479

`\l__enumext_vspace_b_star_vii_bool` ... 1512

`\l__enumext_vspace_b_star_viii_bool` ... 1523

`\l__enumext_vspace_b_star_X_bool` 95

`__enumext_vspace_below:` .. 55, 1450, 1450, 3314

`__enumext_vspace_below_v:` . 55, 1475, 1475, 3431

`\l__enumext_vspace_below_v_skip` .. 1477, 1481, 1483

`__enumext_vspace_below_vii:` .. 56, 1508, 1508, 3854

`\l__enumext_vspace_below_vii_skip` 1510, 1514, 1516

`__enumext_vspace_below_viii:` . 56, 1508, 1519, 4094

`\l__enumext_vspace_below_viii_skip` 1521, 1525, 1527

`__enumext_widest_from:nnn` .. 40, 711, 711, 726, 738

`\g__enumext_widest_label_tl` 26, 35, 65, 453, 457, 461

`\l__enumext_wrap_label_opt_v_bool` 2973

`\l__enumext_wrap_label_opt_vii_bool` 106, 3905

`\l__enumext_wrap_label_opt_viii_bool` .. 110, 4123

`\l__enumext_wrap_label_opt_X_bool` 95

`\l__enumext_wrap_label_v_bool` 2969, 2973, 2981, 3037

`\l__enumext_wrap_label_vii_bool` .. 106, 3904, 3909, 3917, 3985

`\l__enumext_wrap_label_viii_bool` . 110, 4122, 4127, 4135, 4214

`\l__enumext_wrap_label_X_bool` 95

`__enumext_wrapper_label_v:n` 3039, 3472

`__enumext_wrapper_label_vii:n` 3988

`__enumext_wrapper_label_viii:n` 4217

`\l__enumext_write_aux_file_tl` . 28, 75, 81, 146, 2432, 2438, 2742, 2748

`__enumext_zero_parsep:` ... 50, 1123, 1178, 1178

`enumext*` 5, 3796

`enumXi` 425

`enumXii` 425

`enumXiii` 425

`enumXiv` 425

`enumXv` 425

`enumXvi` 425

`enumXvii` 425

`enumXviii` 425

Environments provide by **enumext**:

`anskey*` 27, 62, 66, 76–78

`enumext*` 24, 25, 28–30, 32, 34, 37, 38, 40–43, 45, 46, 52, 53, 55–59, 61–64, 66–71, 73, 74, 76, 78, 80, 83, 89–91, 102, 104–106, 108, 110, 112, 114, 115, 118, 120

`enumext` 24, 25, 29, 30, 32, 34–51, 53–59, 61–64, 66–71, 73, 74, 76, 78, 80, 83–88, 90, 91, 93, 94, 98, 99, 101, 105, 114, 115, 118, 119

`keyans*` 24, 25, 27–30, 34, 37–40, 42, 43, 45, 46, 52, 53, 55, 56, 62, 63, 66, 67, 69, 76, 80, 83, 89, 109, 118, 120

`keyanspic` 24, 25, 27, 30, 34, 35, 39, 53, 62, 63, 66, 69, 76, 79–82, 96–98, 120

`keyans` 24, 25, 27, 29, 30, 34, 35, 39, 40, 42–45, 47, 50, 51, 53–55, 62, 63, 66, 67, 69, 76, 79–82, 86–88, 93, 94, 96–98, 101, 110, 118, 120

Environments:

`list` 29, 32, 87, 90

`lrbox` 99, 107, 112

`minipage` 29, 32, 46, 48, 96–99, 107, 112

`multicols` 46–48, 53, 92, 93, 95, 96

`scontents` 77, 78

exp commands:

`\exp_after:wN` 4293

`\exp_args:Ne` 2651, 2659, 3191, 4281

©2024 by Pablo González L

columns	26, 45, 48, 54, 92, 95
first	43-45, 107
font	35, 86, 107
item-pos*	83
item-sym*	27, 83, 84
item*-sep	84
itemindent	26, 42, 86, 107
itemsep	40, 89
labelsep	35, 84, 88, 107
labelwidth	34-37, 39, 40, 88
label	25, 26, 34-36, 39, 40, 99
lisparindent	89
list-indent	26, 42, 98
list-offset	42
listparindent	42, 107
mark-ans	66, 69, 75
mark-pos	66, 67, 117
mark-ref	66, 69, 73, 74
mini-env	26, 32, 45, 53, 54, 69, 83, 92, 94, 101, 102, 104, 109
mini-right*	26, 29, 46, 69, 102
mini-right	26, 29, 46, 53, 69, 102
mini-sep	26, 45, 69, 92, 94
no-store	27, 61-63, 69, 71
noitemsep	40, 50
nosep	40, 50
parindent	89
parsep	40, 89, 107
partopsep	40
ref	25, 29, 36, 37, 39, 118
resume*	25, 56, 57, 61, 62, 69, 93
resume	25, 31, 56-62, 69, 93, 104
rightmargin	42
save-ans	27, 31, 57-64, 66-69, 71, 72, 76, 77, 80, 81, 85, 90, 94, 96, 97, 104, 110, 111, 113, 114, 118
save-key	27, 57, 68, 90
save-pos	69
save-ref	28, 33, 66, 69, 72-74, 80, 81, 86, 111
save-sep	66, 69, 111
series	25, 56-61, 69, 90, 93
show-ans	66, 67, 69, 71, 72, 75, 86, 111
show-length	29, 43, 118
show-pos	27, 66, 67, 71, 72, 75, 82, 86, 111
start	26, 29, 39, 40, 57
store-key	67
topsep	40
widest	26, 29, 40
wrap-ans	66, 69, 71, 75
wrap-label*	35, 84, 86, 106, 107, 110
wrap-label	35, 86, 106, 107, 110
wrap-opt	66, 69
keys commands:	
\keys_define:nn	467, 489, 521, 537, 584, 655, 729, 749, 793, 829, 848, 905, 914, 993, 1010, 1421, 1532, 1779, 1840, 2014, 2053, 2089, 2094, 2239, 2528, 2549, 2866, 4306, 4375
\l_keys_key_str	4516
\keys_precompile:nnN	114, 4305, 4308, 4312, 4316, 4320, 4324, 4328
\keys_set:nn	481, 809, 820, 1016, 1426, 1431, 1673, 1678, 1765, 1773, 2307, 2658, 3186, 3191, 3359, 3836, 4081, 4377, 4378, 4379, 4380, 4381, 4382, 4383, 4384, 4385, 4386, 4387, 4388, 4389, 4427
keyval commands:	
\keyval_parse:NNn	1546, 2119

L	
label	535, 582, 655
Labels provide by enumext:	
\Alph*	34, 35
\Roman*	34, 35
\alph*	34, 35
\arabic*	29, 34, 35
\roman*	34, 35
\labelsep	98
\labelsep	3515, 3518
labelsep	465
\labelwidth	35, 98
\labelwidth	3515, 3516
labelwidth	465
\leftmargin	88
\leftmargin	87, 3515
legacy commands:	
\legacy_if:nTF	3948, 3951, 4190, 4193
\legacy_if_gset_false:n	357
\legacy_if_set_false:n	3950, 4192
\legacy_if_set_true:n	3910, 3935, 3942, 3955, 4128, 4156, 4197
\linewidth	92, 94
\linewidth	3243, 3369, 3537, 3564, 3583, 3693, 3748
\list	32
\list	341
list-indent	827
list-offset	827
\listparindent	3517
listparindent	827
\lrbbox	3965, 4203
M	
\makebox	99
\makebox	2229, 2231, 3011, 3979, 3987, 3991, 4216, 4220
\makelabel	84, 86, 87, 99
\makelabel	86, 87, 3017, 3033
\makesavenoteenv	387
mark-ans	2012
mark-pos	2012, 2051
mark-ref	2012
mini-env	991
mini-sep	991
\minipage	32
\minipage	347
\miniright	10, 53, 1367, 3305, 3422
\miniright*	10
mode commands:	
\mode_if_vertical:TF	1061, 1089, 1205, 1284
\mode_leave_vertical:	807, 818, 860, 874, 886, 898, 2197, 2206, 2227, 3009, 3977
msg commands:	
\msg_error:nn	2289, 2294, 3350, 3354, 3441, 3501, 3826, 4067, 4074, 4390
\msg_error:nnn	560, 607, 624, 677, 1371, 1375, 1400, 1417, 1685, 1689, 1804, 2489, 2501, 2509, 2513, 2517, 4295, 4300, 4372, 4443
\msg_error:nnnn	2270, 2274, 2278, 2282, 3341, 3437, 3445, 4355
\msg_error:nnnnn	508, 528, 2032
\msg_fatal:nn	3175
\msg_fatal:nnn	419
\msg_info:nnn	13, 16, 21, 24, 369, 383
\msg_line_context:	4481, 4486, 4491, 4520, 4525, 4530, 4545, 4560, 4564, 4568, 4572, 4576, 4580, 4587,

4594, 4600, 4614, 4618, 4623, 4627, 4632, 4636, 4640,
4645, 4650, 4654, 4659, 4664, 4668, 4673, 4677, 4682,
4687, 4692, 4696, 4700

\msg_log:nnn 1824, 1829, 1834
\msg_log:nnnnn 334, 1962, 1967, 1972
\msg_log:nnnnnn 326
\msg_new:nnn 4444, 4448, 4452, 4456, 4461, 4478, 4483,
4488, 4493, 4502, 4510, 4514, 4518, 4523, 4528, 4543,
4558, 4562, 4566, 4570, 4574, 4578, 4582, 4591, 4597,
4603, 4607, 4611, 4616, 4621, 4625, 4630, 4634, 4638,
4643, 4648, 4652, 4657, 4662, 4666, 4671, 4675, 4680,
4685, 4690, 4694, 4698
\msg_new:nnnn 4469
\msg_term:nnnn . 1788, 1793, 3101, 3111, 3140, 3145
\msg_term:nnnnn 1943
\msg_warning:nn 3304, 3421
\msg_warning:nnnn 1980, 1986, 3049, 3054, 3594, 3607,
3643, 3656
\msg_warning:nnnnn 1938, 1948
\multicolsep 92, 95
\multicolsep 3273, 3394
\myenv 4467
\mypkg 4467

N

\NeedsTeXFormat 3
\newcounter 422
\NewDocumentCommand 1367, 2254, 3433, 4279, 4333, 4397
\NewDocumentEnvironment . 3150, 3319, 3474, 3796, 4040
\newenvsc 2522
\newlabel 34
\newlabel 405
no-store 1838
\noindent 103, 108
\noindent . 3250, 3374, 3702, 3757, 3809, 4020, 4052, 4258
\nointerlineskip 3250, 3374, 3702, 3757
noitemsep 747
\nopagebreak 1072, 1100, 1216, 1295, 1358, 1364
\normalfont 2463, 2836, 2849, 4176
nosep 747

P

Packages:

caption 102
enumext 24, 36, 61, 88, 96, 117
enumitem 34
expl3 99
footnotehyper 33
hyperref 28, 29, 33, 34, 73, 81, 106, 107, 117
lua-visual-debug 48
multicol 24, 117
scontents 24, 76, 79
shortlst 99
\par .. 1072, 1100, 1216, 1295, 1358, 1364, 1393, 1410, 2442,
3294, 3309, 3411, 3426, 3549, 3720, 3734, 3775, 3789,
4020, 4034, 4258, 4274
\parindent 3997, 4226
\parsep 47, 50, 97, 98
\parsep 2198, 2207, 3131, 3514, 3521, 3526
parsep 747
\parskip 3998, 4227
\partopsep 98
\partopsep 3132, 3519
partopsep 747

peek commands:

\peek_meaning:NTF 3887, 3901, 3918, 3929, 4105, 4119,
4136
\peek_meaning_remove:NTF 3894, 4112
\peek_remove_spaces:n 2995
\phantomsection 33
\phantomsection 394
prg commands:
\prg_do_nothing: 398
\prg_new_protected_conditional:Npnn ... 198
\prg_replicate:nn 215
\prg_return_false: 202
\prg_return_true: 201
\printkeyans 16, 114, 4333
prop commands:
\prop_count:N 328, 2148, 2430, 2466, 2736, 2839, 2852,
4179
\prop_gput_if_not_in:Nnn 2146
\prop_if_exist:NTF 1822, 4299
\prop_item:Nn 4302
\prop_new:N 1825
\ProvidesExplPackage 4

R

\raggedcolumns 3282, 3400
\ref 74, 80
ref 535, 582, 655
\refstepcounter 3957, 4199
regex commands:
\regex_match:nnTF 200, 704, 706, 718, 720
\regex_replace_once:nnN 208
\renewcommand 570, 615, 632, 685
\RenewDocumentCommand ... 2888, 2956, 2991, 3017, 3033
\RequirePackage 17, 25
resume 1530
resume* 1530
rightmargin 827
\Roman 35, 39, 40
\Roman 441
\roman 35, 39, 40
\roman 442, 553, 4323

S

save-ans 1777
save-key 2087
save-ref 2012
save-sep 2012
scan commands:
\scan_stop: 98, 3528, 3808, 4051, 4290, 4293
scontents internal commands:
\l_scontents_fname_out_tl 2559
__scontents_parse_environment_keys:n . 2565
__scontents_rescan_tokens:n 2572
\l_scontents_storing_bool 2557
\l_scontents_writing_bool 2558
seq commands:
\seq_clear:N 4404
\seq_const_from_clist:Nn 4392
\seq_count:N 329, 3487, 4408
\seq_gclear:N 2886, 2887
\seq_gput_right:Nn 2155, 2899, 2900
\seq_if_empty:NTF 2905, 4348, 4422
\seq_if_exist:NTF 1827, 4346
\seq_if_in:NnTF 4353
\seq_item:Nn 2594, 3546

©2024 by Pablo González L

\tl_trim_spaces:n	452, 4420, 4432, 4438	\use:n	1544, 2117, 4288
\tl_use:N	458, 461, 579, 644, 651, 694, 929, 933, 937, 941, 945, 949, 953, 957, 961, 965, 969, 973, 977, 981, 985, 989, 2233, 2374, 2382, 2393, 2407, 2412, 2424, 2922, 2928, 2952, 2970, 2974, 2982, 3019, 3020, 3027, 3035, 3036, 3042, 3157, 3325, 3471, 3730, 3785, 3984, 3995, 3999, 4213, 4224, 4230, 4235, 4336, 4337, 4338, 4339, 4340, 4358, 4416	\use_none:nn	397
token commands:		\usecounter	3091, 3133
\token_to_str:N	405	V	
\topsep	2199, 2208	\value	1611, 1617, 1624, 1630, 1638, 1644, 1651, 1657
topsep	<u>747</u>	vbox commands:	
\typeout	373, 376, 386, 387	\vbox_set_top:Nn	3728, 3783
U		\vspace	358, 808, 819, 1443, 1446, 1457, 1460, 1470, 1472, 1481, 1483, 1492, 1494, 1503, 1505, 1514, 1516, 1525, 1527, 2198, 2207, 3482, 3493, 4035, 4275
\u	209	W	
use commands:		widest	<u>727</u>
\use:N	216, 3024, 3159	wrap-ans	<u>2012</u>
		wrap-label	<u>465</u>
		wrap-label*	<u>465</u>
		wrap-opt	<u>2012</u>