

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-09^{*}

©2024 by Pablo González[†]

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

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

Abstract

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

Contents

1	Introduction	1	5	The storage system	10
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for <code>label</code> and <code>ref</code>	11
1.3	User interface	3	5.1.2	Keys for <code>wrap</code> and <code>display</code>	11
1.3.1	Internal counters	3	5.1.3	Keys for <code>debug</code> and <code>checking</code>	11
1.3.2	Support for <code>multicol</code>	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>minipage</code>	3	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	13
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
2	The environments provided	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	16
3	The command <code>\setenumext</code>	6	6	Full examples	17
4	The <code>keyval</code> system	6	7	The way of non-enumerated lists	19
4.1	Keys for <code>label</code> and <code>ref</code>	6	8	References	21
4.2	Keys for spaces	7	9	Change history	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 2024-06-09.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

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

1 Introduction

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

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

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

3. True False

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

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

* $(x - 1)^2$

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

* $3(x + y + z)$

3. True False

(a) $\alpha > \delta$

* False

(b) L^AT_EXze is cool?

* Very True!

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

* Yes

(b) Usually uses the package manager?

* Yes, dnf

(c) Rate the following package and class

i. xsim-exam

* doesn't exist for now :(

ii. xsim

* very good

iii. exsheets

* obsolete

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

* 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

E) value

4. Question with image and label below:

A

B

A

A) B) C)

A

duck

D) E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

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

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

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

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

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

2.3.1 Keys for `\item*`

`item-sym*` = { $\langle symbol \rangle$ }

default: $\$ \star \$$

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

`item-pos*` = { $\langle rigid length \rangle$ }

default: *by levels*

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

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

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

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

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

3 The command `\setenumext`

`\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 \TeX 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.

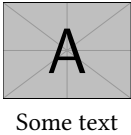
Example

```
\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}
```

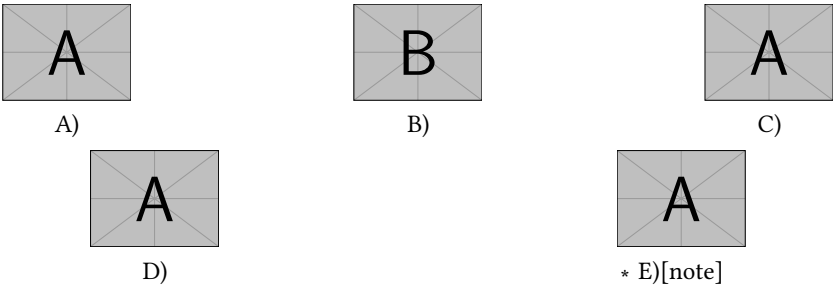
The `\anspic` command take three arguments, the *starred argument* `*` store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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) ~~ETEX~~e is cool?

[2] Very True!

3. Related to Linux

(a) You use linux?

[3] Yes

(b) Rate the following package and class

i. xsim

[4] very good

ii. exsheets

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

6 Full examples

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

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

and then you can use the excellent 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



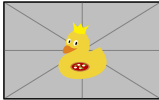
D



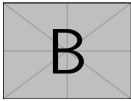
B



C



E



5. Question with image on left side:

- A

value
- B

value
- C

value
- D

correct
- E

value

- D

I and III only
- E

I, II, and III

- D

value
- 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] MITTELBAACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ \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 2024-06-09 – 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   {2024-06-09}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the `multicol` and `scontents` packages are loaded, if not we load it.

```
9 \hook_gput_code:nnn {begindocument} {enumext}
10 {
11   \IfPackageLoadedTF { multicol }
12   {
13     \msg_info:nnn { enumext } { package-load } { multicol }
14   }
15   {
16     \msg_info:nnn { enumext } { package-not-load } { multicol }
17     \RequirePackage{multicol}[2024-05-23]
18   }
19   \IfPackageLoadedTF { scontents }
20   {
21     \msg_info:nnn { enumext } { package-load } { scontents }
22   }
23   {
24     \msg_info:nnn { enumext } { package-not-load } { scontents }
25     \RequirePackage{scontents}
26   }
27 }
```

11.4 Definition of variables

Variables that do not appear in this section are created by means of `\keys_define:nn` or some function described below.

```
\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int
```

Integer variables will control the nesting levels of the environments and `\anskey` command.

```
28 \int_new:N \l__enumext_level_int
29 \int_new:N \l__enumext_level_h_int
30 \int_new:N \l__enumext_anskey_level_int
31 \int_new:N \l__enumext_keyans_level_int
32 \int_new:N \l__enumext_keyans_level_h_int
33 \int_new:N \l__enumext_keyans_pic_level_int
```

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

```
\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_first_bool
\l__enumext_anskey_env_bool
\l__enumext_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

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

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§11.16). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.34.1).

```

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

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

The integer variable `\l__enumext_start_X_int` are used by the `start` key (§11.13), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§11.16.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.11). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§11.18).

```

96 \cs_set_protected:Npn \__enumext_tmp:n #1
97 {
98   \bool_new:c { \l__enumext_wrap_label_#1_bool }
99   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
100   \int_new:c { \l__enumext_start_#1_int }
101   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
102   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
103   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
104   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
105   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
106 }
107 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```


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

```
\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_anskey_env_tl
\l__enumext_store_anskey_opt_tl
\l__enumext_store_current_label_tl
\l__enumext_store_current_opt_arg_tl
\l__enumext_store_current_label_tmp_tl
```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§11.24.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_name_tl` saves the $\langle \textit{store name} \rangle$ 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.

```
108 \bool_new:N \l__enumext_store_active_bool
109 \tl_new:N \l__enumext_store_name_tl
110 \tl_new:N \g__enumext_store_name_tl
111 \tl_new:N \l__enumext_store_anskey_arg_tl
112 \tl_new:N \l__enumext_store_anskey_env_tl
113 \tl_new:N \l__enumext_store_anskey_opt_tl
114 \tl_new:N \l__enumext_store_current_label_tl
115 \tl_new:N \l__enumext_store_current_opt_arg_tl
116 \tl_new:N \l__enumext_store_current_label_tmp_tl
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq
```

Internal variables used by the command `\setenumext` (§11.43).

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

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

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

Internal variables used by command `\printkeyans` (§11.42), `show-pos` key (§11.26), `item-sym*` key (§11.30), `save-key` key (§11.26.2) and “*storage level system*”.

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

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

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

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

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

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

```
\l__enumext_check_answers_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int
```

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.

```
138 \bool_new:N \l__enumext_check_answers_bool
139 \bool_new:N \g__enumext_check_ans_key_bool
140 \tl_new:N \l__enumext_check_start_line_env_tl
141 \int_new:N \g__enumext_check_starred_cmd_int
142 \int_new:N \g__enumext_item_anskey_int
143 \int_new:N \g__enumext_item_number_int
144 \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`.

```
145 \bool_new:N \l__enumext_hyperref_bool
146 \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.

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

```
155 \int_new:N \g__enumext_footnote_int
156 \seq_new:N \g__enumext_footnote_arg_seq
157 \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.

```
158 \cs_set_protected:Npn \__enumext_tmp:n #1
159 {
160   \bool_new:c { l__enumext_item_starred_#1_bool }
161   \int_new:c { l__enumext_item_column_pos_#1_int }
162   \int_new:c { g__enumext_item_count_all_#1_int }
163   \int_new:c { l__enumext_joined_item_#1_int }
164   \int_new:c { l__enumext_joined_item_aux_#1_int }
165   \int_new:c { l__enumext_tmpa_#1_int }
166   \box_new:c { l__enumext_item_text_#1_box }
167   \dim_new:c { l__enumext_joined_width_#1_dim }
168   \dim_new:c { l__enumext_item_width_#1_dim }
169   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
170   \str_new:c { l__enumext_align_label_#1_str }
171   \bool_new:c { g__enumext_minipage_active_#1_bool }
172   \box_new:c { l__enumext_miniright_code_#1_box }
173   \bool_new:c { g__enumext_minipage_center_#1_bool }
174   \dim_new:c { g__enumext_minipage_right_#1_dim }
175   \skip_new:c { g__enumext_minipage_right_#1_skip }
176 }
177 \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`.

```
178 \clist_const:Nn \c__enumext_all_envs_clist
179 {
180   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
181   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
182 }
```

(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.

```
183 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
184 {
185   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
186 }
```

(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.

```
187 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
188 {
189   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
190 }
191 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
192 {
193   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
194 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
195 \cs_new:Nn \__enumext_level:
196 {
197   \int_to_roman:n { \l__enumext_level_int }
198 }
```

(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?](#).

```
199 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
200 {
201   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
202   { \prg_return_true: }
203   { \prg_return_false: }
204 }
```

(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.

```
205 \cs_new_protected:Nn \__enumext_regex_counter_style:
206 {
207   \tl_map_inline:Nn \c__enumext_counter_style_tl
208   {
209     \regex_replace_once:nnN { \c{##1}\* }
210     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
211   }
212 }
```

(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.

```
213 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
214 {
215   * ~ #2
216   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
217   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
218 }
```

(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.

```

219 \cs_new_protected:Nn __enumext_is_not_nested:
220 {
221   \str_case:en { \@currentenv }
222   {
223     {enumext}
224     {
225       \bool_lazy_and:nnT
226       { \bool_not_p:n { \g__enumext_standar_bool } }
227       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
228       {
229         \bool_gset_true:N \g__enumext_standar_bool
230       }
231     }
232     {enumext*}
233     {
234       \bool_lazy_and:nnT
235       { \bool_not_p:n { \g__enumext_starred_bool } }
236       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
237       {
238         \bool_gset_true:N \g__enumext_starred_bool
239       }
240     }
241   }
242 }

```

The function `__enumext_is_on_first_level:` will set the variables `l__enumext_standar_first_bool` (§11.24.1), `l__enumext_starred_first_bool` (§11.24.1) and `l__enumext_anskey_env_bool` (§11.28) to “true” only if the environment is not nested and we are in the “first level” of it. We will also save the *start line number* of each environment in the variable `g__enumext_start_line_tl` and the *name* of each environment in the variable `g__enumext_envir_name_tl` to use in messages related to the `check-ans` key and `.log` file.

```

243 \cs_new_protected:Nn __enumext_is_on_first_level:
244 {
245   \bool_lazy_all:nT
246   {
247     { \bool_if_p:N \g__enumext_standar_bool }
248     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
249     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
250   }
251   {
252     \bool_set_true:N \l__enumext_standar_first_bool
253     \bool_set_true:N \l__enumext_anskey_env_bool
254     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
255     \tl_gset:Nn \g__enumext_start_line_tl
256     {
257       on ~ line ~ \exp_not:V \inputlineno
258     }
259   }
260   \bool_lazy_all:nT
261   {
262     { \bool_if_p:N \g__enumext_starred_bool }
263     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
264     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
265   }
266   {
267     \bool_set_true:N \l__enumext_starred_first_bool
268     \bool_set_true:N \l__enumext_anskey_env_bool
269     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
270     \tl_gset:Nn \g__enumext_start_line_tl
271     {
272       on ~ line ~ \exp_not:V \inputlineno
273     }
274   }
275 }

```

(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 \\enumext _check_start_line_env_tl to use in the \\enumext_check_starred_cmd:n function.

```

276 \cs_new_protected:Nn \\enumext_keyans_start_line:
277 {
278   \str_case:en { \@currentenvir }
279   {
280     {keyans}
281     {
282       \tl_set:Nc \\enumext_check_start_line_env_tl
283       {
284         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
285       }
286     }
287     {keyans*}
288     {
289       \tl_set:Nc \\enumext_check_start_line_env_tl
290       {
291         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
292       }
293     }
294     {keyanspic}
295     {
296       \tl_set:Nc \\enumext_check_start_line_env_tl
297       {
298         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
299       }
300     }
301   }
302 }
```

(End of definition for \\enumext_keyans_start_line:.)

11.5.2 Utilities for log and terminal

\\enumext_reset_global_vars:

The function \\enumext_reset_global_vars: will be passed to the function \\enumext_execute_after_env: and will return the global variables to their default values after being used.

\\enumext_reset_global_int:

\\enumext_reset_global_bool:

\\enumext_reset_global_tl:

```

303 \cs_new_protected:Nn \\enumext_reset_global_vars:
304 {
305   \\enumext_reset_global_int:
306   \\enumext_reset_global_bool:
307   \\enumext_reset_global_tl:
308 }
309 \cs_new_protected:Nn \\enumext_reset_global_int:
310 {
311   \int_gzero:N \\enumext_item_number_int
312   \int_gzero:N \\enumext_item_anskey_int
313   \int_gzero:N \\enumext_item_answer_diff_int
314 }
315 \cs_new_protected:Nn \\enumext_reset_global_bool:
316 {
317   \bool_gset_false:N \\enumext_check_ans_key_bool
318   \bool_gset_false:N \\enumext_standar_bool
319   \bool_gset_false:N \\enumext_starred_bool
320 }
321 \cs_new_protected:Nn \\enumext_reset_global_tl:
322 {
323   \tl_gclear:N \\enumext_store_name_tl
324   \tl_gclear:N \\enumext_start_line_tl
325   \tl_gclear:N \\enumext_envir_name_tl
326 }
```

(End of definition for \\enumext_reset_global_vars: and others.)

\\enumext_log_global_vars:

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

\\enumext_log_answer_vars:

```

327 \cs_new_protected:Nn \\enumext_log_global_vars:
328 {
329   \msg_log:nneeee { enumext } { prop-seq-int-hook }
330   { \\enumext_store_name_tl }
```

```

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

```

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.

```

335 \cs_new_protected:Nn \__enumext_log_answer_vars:
336 {
337     \msg_log:nneee { enumext } { item-answer-hook }
338     { \int_use:N \g__enumext_item_number_int }
339     { \int_use:N \g__enumext_item_anskey_int }
340     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
341 }

```

(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.

```

342 \__enumext_at_begin_document:n
343 {
344     \cs_new_eq:NN \__enumext_start_list:nn \list
345     \cs_new_eq:NN \__enumext_stop_list: \endlist
346     \cs_new_eq:NN \__enumext_item_std:w \item
347 }

```

(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.

```

348 \__enumext_at_begin_document:n
349 {
350     \cs_new_eq:NN \__enumext_minipage:w \minipage
351     \cs_new_eq:NN \__enumext_endminipage: \endminipage
352 }

```

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

11.7 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` environment (*custom version* of `minipage`) setting the `\ifminipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\vspace{0pt}` to maintain alignment on “top”. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§11.35) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.39)

```

353 \cs_new_protected:Nn \__enumext_internal_mini_page:
354 {
355     \int_compare:nNtT { \l__enumext_level_int } = { 0 }
356     {
357         \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
358         {

```



```

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

```

(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.

```

366 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
367 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

The function `__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyperref_bool` to “true” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `__enumext_footnotes_key_bool` to “true”.

```

368 \cs_new_protected:Nn \__enumext_after_hyperref:
369 {
370     \IfPackageLoadedTF { hyperref }
371     {
372         \msg_info:nnn { enumext } { package-load } { hyperref }
373         \bool_set_true:N \l__enumext_hyperref_bool
374         \IfHyperBoolean{hyperfootnotes}
375         {
376             \typeout{hyperfootnotes=true}
377             \bool_set_true:N \l__enumext_footnotes_key_bool
378         }
379         { \typeout{hyperfootnotes=false} }
380     }
381     { }

```

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`.

```

382 \bool_if:NT \l__enumext_footnotes_key_bool
383 {
384     \IfPackageLoadedTF { footnotehyper }
385     {
386         \msg_info:nnn { enumext } { package-load } { footnotehyper }
387     }
388     {
389         \typeout{No ~ footnotehyper ~ load}
390         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
391         \bool_set_false:N \l__enumext_footnotes_key_bool
392     }
393 }

```

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.

```

394 \bool_if:NTF \l__enumext_hyperref_bool
395 {
396     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
397     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
398 }
399 {
400     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
401     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
402 }
403 }

```

(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.

```

404 \cs_new_protected:Npn \l__enumext_newlabel:nn #1 #2
405 {
406   \protected@write \@auxout { }
407   {
408     \token_to_str:N \newlabel {#1}
409     {
410       {#2}
411       \bool_if:NT \l__enumext_hyperref_bool
412       { { \thepage } {#2} {#1} }
413       { }
414     }
415   }
416   \__enumext_hypertarget:nn {#1} { }
417   \__enumext_phantomsection:
418 }

```

(End of definition for `\l__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.

```

419 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
420 {
421   \cs_if_exist:cTF { c@ #2 }
422   { \msg_fatal:nnn { enumext } { counters } { #2 } }
423   {
424     \tl_set:Nn #1 { #2 }
425     \newcounter { #2 }
426   }
427 }

```

(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  428 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii 429 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii 430 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv 431 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXvii 432 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXviii 433 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
434 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
435 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

11.10 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```

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

```
446 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
447 {
448   \hbox_set:Nn \__enumext_label_width_by_box {#2}
449   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
450 }
451 \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.

```
452 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
453 {
454   \tl_clear_new:N #1
455   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
456   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
457   \tl_map_inline:Nn \g__enumext_counter_styles_tl
458   {
459     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
460     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
461     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
462   }
463   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
464   { \tl_use:N \g__enumext_widest_label_tl }
465   \tl_set_eq:cN { the #2 } #1
466 }
467 \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.

```
468 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
469 {
470   \keys_define:nn { enumext / #1 }
471   {
472     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
473     font .value_required:n = true,
474     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
475     labelsep .initial:n = {0.3333em},
476     labelsep .value_required:n = true,
477     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
478     labelwidth .value_required:n = true,
479     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
480     wrap-label .initial:n = {##1},
481     wrap-label .value_required:n = true,
482     wrap-label* .code:n = {
483       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
484       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
485     },
486     wrap-label* .value_required:n = true,
487   }
488 }
489 \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.

```

490 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
491 {
492   \keys_define:nn { enumext / #1 }
493   {
494     align .choice:,
495     align / left .code:n =
496       {
497         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
498         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
499       },
500     align / right .code:n =
501       {
502         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
503         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
504       },
505     align / center .code:n =
506       {
507         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
508         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
509       },
510     align / unknown .code:n =
511       \msg_error:nneee { enumext } { unknown-choice }
512       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
513     align .initial:n = left,
514     align .value_required:n = true,
515   }
516 }
517 \clist_map_inline:nn
518 {
519   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
520 }
521 { \__enumext_tmp:nn #1 }

522 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
523 {
524   \keys_define:nn { enumext / #1 }
525   {
526     align .choice:,
527     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
528     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
529     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
530     align / unknown .code:n =
531       \msg_error:nneee { enumext } { unknown-choice }
532       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
533     align .initial:n = left,
534     align .value_required:n = true,
535   }
536 }
537 \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

538 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
539 {
540   \keys_define:nn { enumext / #1 }
541   {
542     label .code:n = {
543       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
544       { l__enumext_counter_#2_tl } {##1}
545       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
546       \l__enumext_current_widest_dim

```

```

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

```

558 \cs_new_protected:Npn \__enumext_standar_ref:n #1
559 {
560     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
561     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
562     {
563         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
564     }
565     {
566         \tl_set_eq:Nc
567         \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
568         \__enumext_regex_counter_style:
569         \tl_set_eq:Nc
570         \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
571         \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
572         {
573             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
574             { \exp_not:V \l__enumext_ref_key_arg_tl }
575         }
576     }
577 }

```

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

```

578 \cs_new_protected:Npn \__enumext_standar_ref:
579 {
580     \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
581     {
582         \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
583     }
584 }

```

(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
585 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
586 {
587     \keys_define:nn { enumext / #1 }
588     {
589         label .code:n = {
590             \__enumext_label_style:cnv { \l__enumext_label_#2_tl }
591             { \l__enumext_counter_#2_tl } {##1}
592             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
593             \l__enumext_current_widest_dim
594         },
595         label .initial:n = #3,
596         label .value_required:n = true,
597         ref .code:n = \__enumext_starred_ref:n {##1},
598         ref .value_required:n = true,
599     }

```

```

600 }
601 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
602 \__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`.

```

603 \cs_new_protected:Npn \__enumext_starred_ref:n #1
604 {
605   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
606   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
607   {
608     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
609     {
610       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
611     }
612     {
613       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
614       \__enumext_regex_counter_style:
615       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
616       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
617       {
618         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
619         { \exp_not:V \l__enumext_ref_key_arg_tl }
620       }
621     }
622   }
623   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
624   {
625     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
626     {
627       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
628     }
629     {
630       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
631       \__enumext_regex_counter_style:
632       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
633       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
634       {
635         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
636         { \exp_not:V \l__enumext_ref_key_arg_tl }
637       }
638     }
639   }
640 }

```

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.

```

641 \cs_new_protected:Nn \__enumext_starred_ref:
642 {
643   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
644   {
645     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
646     {
647       \tl_use:N \l__enumext_renew_the_count_vii_tl
648     }
649   }
650   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
651   {
652     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
653     {
654       \tl_use:N \l__enumext_renew_the_count_viii_tl
655     }
656   }
657 }

```

(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 658 \keys_define:nn { enumext / keyans }
\__enumext_label_vi_tl 659 {
660   label .code:n = {
661     \__enumext_label_style:cnv { \__enumext_label_v_tl }
662     { \__enumext_counter_v_tl } {#1}
663     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
664     \__enumext_current_widest_dim
665     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
666     { \__enumext_counter_vi_tl } {#1}
667     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
668     \__enumext_current_widest_dim
669   },
670   label .initial:n = \Alph*,
671   label .value_required:n = true,
672   ref .code:n = \__enumext_keyans_ref:n {#1},
673   ref .value_required:n = true,
674 }

```

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

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:
675 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
676 {
677   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
678   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
679   {
680     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
681   }
682   {
683     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_v_tl
684     \__enumext_regex_counter_style:
685     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
686     \tl_put_right:Ne \__enumext_renew_the_count_v_tl
687     {
688       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
689       { \exp_not:V \__enumext_ref_key_arg_tl }
690     }
691   }
692 }

```

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

```

693 \cs_new_protected:Nn \__enumext_keyans_ref:
694 {
695   \tl_if_empty:NF \__enumext_renew_the_count_v_tl
696   {
697     \tl_use:N \__enumext_renew_the_count_v_tl
698   }
699 }

```

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

11.13 Setting start and widest keys

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

```

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

```

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

```

700 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
701 {
702   \__enumext_if_is_int:nTF { #3 }
703   {
704     \int_set:Nn #2 {#3}
705   }

```



```

706     {
707         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
708         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
709         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
710         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
711     }
712 }
713 \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.

```

714 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
715 {
716     \__enumext_if_is_int:nTF {#4}
717     {
718         \setcounter{enumX#1} { #4 }
719     }
720     {
721         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
722         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
723         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
724         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
725     }
726     \__enumext_label_width_by_box:cv
727     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
728 }
729 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for __enumext_widest_from:nNNn.)

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

```

start
widest
\l__enumext_start_X_int
730 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
731 {
732     \keys_define:nn { enumext / #1 }
733     {
734         start .code:n = {
735             \__enumext_start_from:ccn
736             { \l__enumext_label_#2_tl }
737             { \l__enumext_start_#2_int } {##1}
738         },
739         start .initial:n = 1,
740         widest .code:n = {
741             \__enumext_widest_from:nccn {#2}
742             { \l__enumext_label_#2_tl }
743             { \l__enumext_labelwidth_#2_dim } {##1}
744         },
745         widest .value_required:n = true,
746         start .value_required:n = true,
747     }
748 }
749 \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
750 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
751 {
752     \keys_define:nn { enumext / #1 }

```

```

753 {
754   topsep .skip_set:c = { l__enumext_topsep_#2_skip },
755   topsep .initial:n = {#3},
756   topsep .value_required:n = true,
757   partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
758   partopsep .initial:n = {#4},
759   partopsep .value_required:n = true,
760   parsep .skip_set:c = { l__enumext_parsep_#2_skip },
761   parsep .initial:n = {#5},
762   parsep .value_required:n = true,
763   itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
764   itemsep .initial:n = {#6},
765   itemsep .value_required:n = true,
766   noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
767   noitemsep .value_forbidden:n = true,
768   nosep .meta:n = {
769     itemsep = 0pt, parsep= 0pt,
770     topsep = 0pt, partopsep = 0pt,
771   },
772   nosep .value_forbidden:n = true,
773 }
774 }

```

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

```

775 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
776 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
777 { 4.0pt plus 2.0pt minus 1.0pt }
778 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
779 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
780 { 2.0pt plus 1.0pt minus 1.0pt }
781 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
782 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
783 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
784 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
785 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
786 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
787 { 2.0pt plus 1.0pt minus 1.0pt }
788 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
789 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
790 { 4.0pt plus 2.0pt minus 1.0pt }
791 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
792 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
793 { 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*`.

```

794 \cs_set_protected:Npn \__enumext_tmp:n #1
795 {
796   \keys_define:nn { enumext / #1 }
797   {
798     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
799     base-fix .initial:n = true,
800     base-fix .value_forbidden:n = true,
801   }
802 }
803 \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)

```

804 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
805 {
806   \bool_lazy_and:nnT

```

```

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

```

(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.

```

830 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
831 {
832   \keys_define:nn { enumext / #1 }
833   {
834     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
835     itemindent .value_required:n = true,
836     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
837     rightmargin .value_required:n = true,
838     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
839     listparindent .value_required:n = true,
840     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
841     list-offset .value_required:n = true,
842     list-indent .code:n =
843       \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
844       \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {##1},
845     list-indent .value_required:n = true,
846   }
847 }
848 \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.

```

849 \cs_set_protected:Npn \__enumext_tmp:n #1
850 {
851   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
852 }
853 \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.

```

854 \cs_set_protected:Nn \__enumext_fake_item:
855 {
856   \dim_compare:nNnT
857     { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
858     >
859     { \c_zero_dim }
860   {

```

```

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

```

(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.

```

906 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907 {
908     \keys_define:nn { enumext / #1 }
909     {
910         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
911         show-length .initial:n = false,
912     }
913 }
914 \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.

```

915 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
916 {

```

```

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

```

930 \cs_new_protected:Nn \__enumext_before_args_exec:
931 {
932     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
933 }

```

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 $\langle keys \rangle$.

```

934 \cs_new_protected:Nn \__enumext_before_keys_exec:
935 {
936     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
937 }

```

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

```

938 \cs_new_protected:Nn \__enumext_after_stop_list:
939 {
940     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
941 }

```

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*.

```

942 \cs_new_protected:Nn \__enumext_after_args_exec:
943 {
944     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
945 }

```

(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.

```

946 \cs_new_protected:Nn \__enumext_before_args_exec_v:
947 {
948     \tl_use:N \l__enumext_before_starred_key_v_tl
949 }

```

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 $\langle keys \rangle$.

```

950 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
951 {
952     \tl_use:N \l__enumext_before_no_starred_key_v_tl
953 }

```

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

```

954 \cs_new_protected:Nn \__enumext_after_stop_list_v:
955 {
956     \tl_use:N \l__enumext_after_stop_list_v_tl
957 }

```

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`.

```

958 \cs_new_protected:Nn \__enumext_after_args_exec_v:
959 {
960     \tl_use:N \l__enumext_after_list_args_v_tl
961 }

```

(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.

```

962 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
963 {
964     \tl_use:N \l__enumext_before_starred_key_vii_tl
965 }
966 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
967 {
968     \tl_use:N \l__enumext_before_starred_key_viii_tl
969 }

```

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⟩`.

```

970 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
971 {
972     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
973 }
974 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
975 {
976     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
977 }

```

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

```

978 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
979 {
980     \tl_use:N \l__enumext_after_stop_list_vii_tl
981 }
982 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
983 {
984     \tl_use:N \l__enumext_after_stop_list_viii_tl
985 }

```

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`.

```

986 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
987 {
988     \tl_use:N \l__enumext_after_list_args_vii_tl
989 }
990 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
991 {
992     \tl_use:N \l__enumext_after_list_args_viii_tl
993 }

```

(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.

```

994 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
995 {
996     \keys_define:nn { enumext / #1 }
997     {
998         mini-env    .dim_set:c = { l__enumext_minipage_right_#2_dim },
999         mini-env    .value_required:n = true,
1000         mini-sep    .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1001         mini-sep    .initial:n = 0.3333em,

```



```

1002     mini-sep    .value_required:n = true,
1003     columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1004     columns-sep .value_required:n = true,
1005     columns     .int_set:c  = { l__enumext_columns_#2_int },
1006     columns     .initial:n  = 1,
1007     columns     .value_required:n = true,
1008   }
1009 }
1010 \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.

```

1011 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1012 {
1013   \keys_define:nn { enumext / #1 }
1014   {
1015     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1016     mini-right .value_required:n = true,
1017     mini-right* .code:n = {
1018       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1019       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1020     },
1021     mini-right* .value_required:n = true,
1022   }
1023 }
1024 \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*.

```

1025 \cs_new_protected:Npn \__enumext_multi_set_vskip:
1026 {
1027   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1028   {
1029     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1030   }
1031   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1032   {
1033     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1034   }
1035   \__enumext_add_pre_parsep:
1036 }

```

(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*.

```

1037 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1038 {
1039   \int_case:nn { \l__enumext_level_int }
1040   {
1041     { 2 }{
1042       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1043       {
1044         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1045       }
1046     }
1047     { 3 }{
1048       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1049       {
1050         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1051       }
1052     }
1053     { 4 }{
1054       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1055       {
1056         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1057       }
1058     }
1059   }
1060 }

```

(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*.

```

1061 \cs_new_protected:Nn \__enumext_multi_addvspace:
1062 {
1063   \__enumext_multi_set_vskip:
1064   \mode_if_vertical:T
1065   {
1066     \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1067     {
1068       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1069     }
1070     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1071     {
1072       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1073     }
1074   }
1075   \par\nopagebreak
1076   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1077 }

```

(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`.

```

1078 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1079 {
1080   \skip_set:Nn \l__enumext_multicols_above_v_skip
1081   {
1082     \l__enumext_topsep_v_skip
1083   }
1084   \skip_set:Nn \l__enumext_multicols_below_v_skip
1085   {
1086     \l__enumext_topsep_v_skip
1087   }
1088 }

```

```

1089 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1090 {
1091   \__enumext_keyans_multi_set_vskip:
1092   \mode_if_vertical:T
1093   {
1094     \skip_add:Nn \l__enumext_multicols_above_v_skip
1095     {
1096       \skip_use:N \l__enumext_partopsep_v_skip
1097     }
1098     \skip_add:Nn \l__enumext_multicols_below_v_skip
1099     {
1100       \skip_use:N \l__enumext_partopsep_v_skip
1101     }
1102   }
1103   \par\nopagebreak
1104   \addvspace{ \l__enumext_multicols_above_v_skip }
1105 }

```

(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.

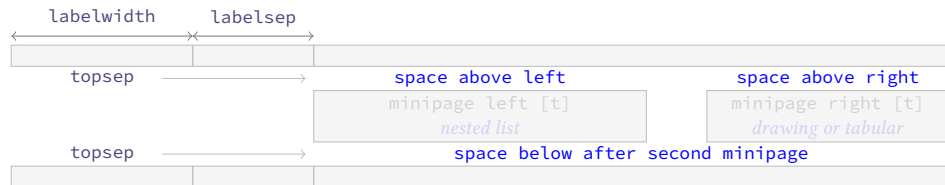


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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`.

```

1106 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1107 {
1108   \int_compare:nNtF
1109   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1110   {

```

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.

```

1111   \skip_if_eq:nnTF
1112   { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1113   {
1114     \skip_set:Nn \l__enumext_minipage_left_skip
1115     {
1116       -0.150\box_dp:N \strutbox

```

```

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

```

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.

```

1145     \skip_if_eq:nnTF
1146     { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1147     {
1148         \skip_set:Nn \l__enumext_minipage_left_skip
1149         {
1150             0.5\box_dp:N \strutbox
1151             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1152         }
1153         \skip_set:Nn \l__enumext_minipage_right_skip
1154         {
1155             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1156         }
1157         \skip_set:Nn \l__enumext_minipage_after_skip
1158         {
1159             1.6\box_dp:N \strutbox
1160         }
1161     }
1162     {
1163         \skip_set:Nn \l__enumext_minipage_left_skip
1164         {
1165             0.5875\box_dp:N \strutbox
1166             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1167         }
1168         \skip_set:Nn \l__enumext_minipage_right_skip
1169         {
1170             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1171             + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1172         }
1173         \skip_set:Nn \l__enumext_minipage_after_skip
1174         {
1175             0.325\box_dp:N \strutbox
1176             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1177         }
1178     }
1179 }
1180 }

```

(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.

```

1181 \cs_new_protected:Nn \__enumext_zero_parsep:
1182 {
1183   \int_case:nn { \l__enumext_level_int }
1184   {
1185     { 2 }{
1186       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1187       {
1188         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1189       }
1190     }
1191     { 3 }{
1192       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1193       {
1194         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1195       }
1196     }
1197     { 4 }{
1198       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1199       {
1200         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1201       }
1202     }
1203   }
1204 }

```

(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*.

```

1205 \cs_new_protected:Nn \__enumext_mini_addvspace:
1206 {
1207   \__enumext_mini_set_vskip:
1208   \mode_if_vertical:T
1209   {
1210     \skip_add:Nn \l__enumext_minipage_left_skip
1211     {
1212       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1213     }
1214     \skip_add:Nn \l__enumext_minipage_after_skip
1215     {
1216       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1217     }
1218   }
1219   \par\nopagebreak
1220   \addvspace { \l__enumext_minipage_left_skip }
1221 }

```

(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`.

```

1222 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1223 {
1224   \skip_zero_new:N \l__enumext_minipage_after_skip
1225   \skip_zero_new:N \l__enumext_minipage_left_skip
1226   \skip_zero_new:N \l__enumext_minipage_right_skip
1227   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1228   {
1229     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1230     {
1231       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1232       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1233       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }

```

```

1234     \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1235     {
1236       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1237     }
1238   }
1239   {
1240     \skip_set:Nn \l__enumext_minipage_left_skip
1241     {
1242       \skip_use:N \l__enumext_topsep_v_skip
1243     }
1244     \skip_set:Nn \l__enumext_minipage_right_skip
1245     {
1246       0.705\box_dp:N \strutbox
1247     }
1248     \skip_set:Nn \l__enumext_minipage_after_skip
1249     {
1250       1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1251     }
1252   }
1253 }
1254 {
1255   \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1256   {
1257     \skip_set:Nn \l__enumext_minipage_left_skip
1258     {
1259       0.5\box_dp:N \strutbox
1260       + \l__enumext_partopsep_v_skip
1261     }
1262     \skip_set:Nn \l__enumext_minipage_right_skip
1263     {
1264       \l__enumext_partopsep_v_skip
1265     }
1266     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1267   }
1268   {
1269     \skip_set:Nn \l__enumext_minipage_left_skip
1270     {
1271       0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1272     }
1273     \skip_set:Nn \l__enumext_minipage_right_skip
1274     {
1275       \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1276     }
1277     \skip_set:Nn \l__enumext_minipage_after_skip
1278     {
1279       0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1280     }
1281   }
1282 }
1283 }

```

(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`.

```

1284 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1285 {
1286   \l__enumext_keyans_mini_set_vskip:
1287   \mode_if_vertical:T
1288   {
1289     \skip_add:Nn \l__enumext_minipage_left_skip
1290     {
1291       \l__enumext_partopsep_v_skip
1292     }
1293     \skip_add:Nn \l__enumext_minipage_after_skip
1294     {
1295       \l__enumext_partopsep_v_skip
1296     }

```



```

1297     }
1298     \par\nopagebreak
1299     \addvspace { \l__enumext_minipage_left_skip }
1300 }

```

(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*`.

```

1301 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1302 {
1303   \skip_zero_new:N \l__enumext_minipage_left_skip
1304   \skip_gzero_new:N \g__enumext_minipage_right_skip
1305   \skip_gzero_new:N \g__enumext_minipage_after_skip
1306   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1307   {
1308     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1309     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1310   }
1311   {
1312     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1313     \skip_gset:Nn \g__enumext_minipage_right_skip
1314     {
1315       \l__enumext_topsep_vii_skip
1316     }
1317     \skip_gset:Nn \g__enumext_minipage_after_skip
1318     {
1319       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1320     }
1321   }
1322 }
1323 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1324 {
1325   \skip_zero_new:N \l__enumext_minipage_after_skip
1326   \skip_zero_new:N \l__enumext_minipage_left_skip
1327   \skip_zero_new:N \l__enumext_minipage_right_skip
1328   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1329   {
1330     \skip_set:Nn \l__enumext_minipage_left_skip
1331     {
1332       0.5\box_dp:N \strutbox
1333     }
1334     \skip_set:Nn \l__enumext_minipage_right_skip
1335     {
1336       \l__enumext_partopsep_viii_skip
1337     }
1338     \skip_set:Nn \l__enumext_minipage_after_skip
1339     {
1340       1.6\box_dp:N \strutbox
1341     }
1342   }
1343   {
1344     \skip_set:Nn \l__enumext_minipage_left_skip
1345     {
1346       0.5875\box_dp:N \strutbox
1347     }
1348     \skip_set:Nn \l__enumext_minipage_right_skip
1349     {
1350       \l__enumext_topsep_viii_skip
1351     }
1352     \skip_set:Nn \l__enumext_minipage_after_skip
1353     {
1354       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1355     }
1356   }
1357 }

```

(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.

```

1358 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1359 {
1360   \__enumext_mini_set_vskip_vii:
1361   \par\nopagebreak
1362   \addvspace { \l__enumext_minipage_left_skip }
1363 }
1364 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1365 {
1366   \__enumext_mini_set_vskip_viii:
1367   \par\nopagebreak
1368   \addvspace { \l__enumext_minipage_left_skip }
1369 }

```

(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.

```

1370 \NewDocumentCommand \miniright { s }
1371 {
1372   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1373   {
1374     \msg_error:nnn { enumext } { wrong-miniright-place }
1375   }
1376   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1377   {
1378     \msg_error:nnn { enumext } { wrong-miniright-place }
1379   }
1380   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1381   {
1382     \__enumext_keyans_mini_right_cmd:n {#1}
1383   }
1384   { \__enumext_mini_right_cmd:n {#1} }
1385 }

```

(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.

```

1386 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1387 {
1388   \dim_compare:nNnTF
1389   { \dim_use:c { \l__enumext_minipage_right_ } \__enumext_level: _dim } > { \c_zero_dim }
1390   {
1391     \__enumext_multicols_stop:
1392     \end{\__enumext_mini_env*}
1393     \hfill
1394     \begin{\__enumext_mini_env*}
1395     { \dim_use:c { \l__enumext_minipage_right_ } \__enumext_level: _dim } }
1396     \par\addvspace { \l__enumext_minipage_right_skip }
1397     \bool_if:nF {#1}
1398     {

```

```

1399         \centering
1400     }
1401     \int_gzero:N \g__enumext_minipage_stat_int
1402 }
1403 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1404 }

```

(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.

```

1405 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1406 {
1407     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1408     {
1409         \__enumext_keyans_multicols_stop:
1410         \end{__enumext_mini_env*}
1411         \hfill
1412         \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1413         \par\addvspace { \l__enumext_minipage_right_skip }
1414         \bool_if:nF {#1}
1415         {
1416             \centering
1417         }
1418         \int_gzero:N \g__enumext_minipage_stat_int
1419     }
1420     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1421 }

```

(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* \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below  {
below* \keys_define:nn { enumext / #1 }
{
1426     above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1427     above .value_required:n = true,
1428     above* .code:n      = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1429                     \keys_set:nn { enumext / #1 } { above = {#1} },
1430     above* .value_required:n = true,
1431     below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1432     below .value_required:n = true,
1433     below* .code:n      = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1434                     \keys_set:nn { enumext / #1 } { below = {#1} },
1435     below* .value_required:n = true,
1436 }
1437 }
1438 \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.

```

1439 \cs_new_protected:Nn \__enumext_vspace_above:
1440 {
1441     \skip_if_eq:nnF
1442     { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1443     {
1444         \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1445         {

```

```

1446         \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1447     }
1448 {
1449     \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1450 }
1451 }
1452 }

```

(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.

```

1453 \cs_new_protected:Nn \__enumext_vspace_below:
1454 {
1455     \skip_if_eq:nnF
1456     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1457     {
1458         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1459         {
1460             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1461         }
1462         {
1463             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1464         }
1465     }
1466 }

```

(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.

```

1467 \cs_new_protected:Nn \__enumext_vspace_above_v:
1468 {
1469     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1470     {
1471         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1472         {
1473             \vspace*{ \l__enumext_vspace_above_v_skip }
1474         }
1475         { \vspace { \l__enumext_vspace_above_v_skip } }
1476     }
1477 }

```

(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.

```

1478 \cs_new_protected:Nn \__enumext_vspace_below_v:
1479 {
1480     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1481     {
1482         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1483         {
1484             \vspace*{ \l__enumext_vspace_below_v_skip }
1485         }
1486         { \vspace { \l__enumext_vspace_below_v_skip } }
1487     }
1488 }

```

(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:

```

1489 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1490 {
1491     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1492     {
1493         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1494         {

```

```

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

```

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

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

```

1511 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1512 {
1513     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1514     {
1515         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1516         {
1517             \vspace*{ \l__enumext_vspace_below_vii_skip }
1518         }
1519         { \vspace { \l__enumext_vspace_below_vii_skip } }
1520     }
1521 }
1522 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1523 {
1524     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1525     {
1526         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1527         {
1528             \vspace*{ \l__enumext_vspace_below_viii_skip }
1529         }
1530         { \vspace { \l__enumext_vspace_below_viii_skip } }
1531     }
1532 }

```

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

11.23 Setting series, resume and resume* keys

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

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1533 \cs_set_protected:Npn \__enumext_tmp:n #1
1534 {
1535     \keys_define:nn { enumext / #1 }
1536     {
1537         series .str_set:N = \l__enumext_series_str,
1538         series .value_required:n = true,
1539         resume .code:n = \__enumext_resume_series:n {##1},
1540         resume* .code:n = \__enumext_resume_starred:,
1541         resume* .value_forbidden:n = true,
1542     }
1543 }
1544 \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.

```

1545 \cs_new:Npn __enumext_filter_series:n #1
1546 {
1547   \use:e
1548   {
1549     \keyval_parse:NNn
1550     __enumext_filter_series_key:n
1551     __enumext_filter_series_pair:nn {#1}
1552   }
1553 }

```

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.

```

1554 \cs_new:Npn __enumext_filter_series_key:n #1
1555 {
1556   \str_case:nnF {#1}
1557   {
1558     { resume } {}
1559     { resume* } {}
1560   }
1561   { , { \exp_not:n {#1} } }
1562 }

```

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.

```

1563 \cs_new:Npn __enumext_filter_series_pair:nn #1#2
1564 {
1565   \str_case:nnF {#1}
1566   {
1567     { series } {}
1568     { resume } {}
1569     { start } {}
1570     { save-ans } {}
1571     { save-key } {}
1572   }
1573   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1574 }

```

(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).

```

1575 \cs_new_protected:Npn __enumext_parse_series:n #1
1576 {
1577   \str_if_empty:NTF \l__enumext_series_str
1578   {
1579     \bool_if:NF \l__enumext_resume_active_bool
1580     {
1581       __enumext_resume_last:n {#1}
1582     }
1583   }
1584   {
1585     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1586     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1587     { __enumext_filter_series:n {#1} }
1588     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1589     {
1590       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1591     }
1592   }

```



```
1593 }
```

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.

```
1594 \cs_new_protected:Npn \__enumext_resume_last:n #1
1595 {
1596   \bool_if:NT \l__enumext_standar_first_bool
1597   {
1598     \tl_gclear:N \g__enumext_standar_series_tl
1599     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1600   }
1601   \bool_if:NT \l__enumext_starred_first_bool
1602   {
1603     \tl_gclear:N \g__enumext_starred_series_tl
1604     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1605   }
1606 }
```

(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).

```
1607 \cs_new_protected:Nn \__enumext_resume_save_counter:
1608 {
1609   \bool_if:NT \g__enumext_standar_bool
1610   {
1611     \tl_if_empty:NF \l__enumext_series_str
1612     {
1613       \int_gset_eq:cN
1614       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1615     }
1616     \tl_if_empty:NTF \l__enumext_resume_name_tl
1617     {
1618       \str_if_empty:NT \l__enumext_series_str
1619       {
1620         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1621       }
1622     }
1623     {
1624       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1625       {
1626         \int_gset_eq:cN
1627         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1628       }
1629     }
1630     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1631     {
1632       \int_gset_eq:cN
1633       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1634     }
1635   }
1636   \bool_if:NT \g__enumext_starred_bool
1637   {
1638     \tl_if_empty:NF \l__enumext_series_str
1639     {
1640       \int_gset_eq:cN
1641       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1642     }
1643   }
```

```

1643     \tl_if_empty:NTF \l__enumext_resume_name_tl
1644     {
1645         \str_if_empty:NT \l__enumext_series_str
1646         {
1647             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1648         }
1649     }
1650     {
1651         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1652         {
1653             \int_gset_eq:cN
1654             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1655         }
1656     }
1657     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1658     {
1659         \int_gset_eq:cN
1660         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1661     }
1662 }
1663 }

```

(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.

```

1664 \cs_new_protected:Npn \__enumext_resume_series:n #1
1665 {
1666     \tl_if_empty:NTF {#1}
1667     {
1668         \__enumext_resume_counter:n { }
1669     }
1670     {
1671         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1672         {
1673             \__enumext_resume_counter:n {#1}
1674             \bool_if:NT \g__enumext_standar_bool
1675             {
1676                 \keys_set:nv { enumext / level-1 }
1677                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1678             }
1679             \bool_if:NT \g__enumext_starred_bool
1680             {
1681                 \keys_set:nv { enumext / enumext* }
1682                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1683             }
1684         }
1685         {
1686             \bool_if:NT \g__enumext_standar_bool
1687             {
1688                 \msg_error:nnn { enumext } { unknown-series } {#1}
1689             }
1690             \bool_if:NT \g__enumext_starred_bool
1691             {
1692                 \msg_error:nnn { enumext } { unknown-series } {#1}
1693             }
1694         }
1695     }
1696 }

```

(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`.

```

1697 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1698 {
1699   \bool_set_true:N \l__enumext_resume_active_bool
1700   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1701   \tl_if_empty:NTF \l__enumext_resume_name_tl
1702   {
1703     \__enumext_resume_counter:
1704   }
1705   {
1706     \__enumext_resume_counter_series:
1707   }
1708   \__enumext_resume_counter_save_ans:
1709 }

```

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.

```

1710 \cs_new_protected:Nn \__enumext_resume_counter:
1711 {
1712   \bool_if:NT \g__enumext_standar_bool
1713   {
1714     \int_gincr:N \g__enumext_resume_int
1715     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1716   }
1717   \bool_if:NT \g__enumext_starred_bool
1718   {
1719     \int_gincr:N \g__enumext_resume_vii_int
1720     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1721   }
1722 }

```

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.

```

1723 \cs_new_protected:Nn \__enumext_resume_counter_series:
1724 {
1725   \bool_if:NT \g__enumext_standar_bool
1726   {
1727     \int_set:Nn \l__enumext_start_i_int
1728     {
1729       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1730     }
1731   }
1732   \bool_if:NT \g__enumext_starred_bool
1733   {
1734     \int_set:Nn \l__enumext_start_vii_int
1735     {
1736       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1737     }
1738   }
1739 }

```

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.

```

1740 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1741 {
1742   \bool_lazy_and:nnT
1743   { \bool_if_p:N \l__enumext_standar_first_bool }
1744   { \bool_if_p:N \l__enumext_store_active_bool }
1745   {
1746     \int_set:Nn \l__enumext_start_i_int
1747     {
1748       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1749     }
1750   }
1751   \bool_lazy_and:nnT
1752   { \bool_if_p:N \l__enumext_starred_first_bool }

```

```

1753     { \bool_if_p:N \__enumext_store_active_bool }
1754     {
1755         \int_set:Nn \__enumext_start_vii_int
1756         {
1757             \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1758         }
1759     }
1760 }

```

(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.

```

1761 \cs_new_protected:Nn \__enumext_resume_starred:
1762 {
1763     \bool_if:NT \g__enumext_standar_bool
1764     {
1765         \tl_if_empty:NF \g__enumext_standar_series_tl
1766         {
1767             \__enumext_resume_counter:n { }
1768             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1769         }
1770     }
1771     \bool_if:NT \g__enumext_starred_bool
1772     {
1773         \tl_if_empty:NF \g__enumext_starred_series_tl
1774         {
1775             \__enumext_resume_counter:n { }
1776             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1777         }
1778     }
1779 }

```

(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*`.

```

1780 \cs_set_protected:Npn \__enumext_tmp:n #1
1781 {
1782     \keys_define:nn { enumext / #1 }
1783     {
1784         save-ans .code:n = \__enumext_storing_set:n {##1},
1785         save-ans .value_required:n = true,
1786     }
1787 }
1788 \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:`.

```

1789 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1790 {
1791     \msg_term:nnVV { enumext } { save-ans-log }
1792     \g__enumext_envir_name_tl \__enumext_store_name_tl
1793 }
1794 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1795 {
1796     \msg_term:nnVV { enumext } { save-ans-log-hook }

```

```

1797     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1798 }

```

(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.

```

1799 \cs_new_protected:Npn \__enumext_storing_set:n #1
1800 {
1801   \tl_set:Nx \l__enumext_store_name_tl {#1}
1802   \tl_if_empty:NTF \l__enumext_store_name_tl
1803   {
1804     \bool_lazy_or:nnT
1805     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1806     {
1807       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1808     }
1809   }
1810   {
1811     \bool_lazy_or:nnT
1812     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1813     {
1814       \__enumext_start_save_ans_msg:
1815       \__enumext_storing_exec:
1816     }
1817   }
1818 }

```

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*`.

```

1819 \cs_new_protected:Nn \__enumext_storing_exec:
1820 {
1821   \bool_set_true:N \l__enumext_store_active_bool
1822   \bool_set_true:N \l__enumext_check_answers_bool
1823   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1824   \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1825   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1826   {
1827     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1828     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1829   }
1830   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1831   {
1832     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1833     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1834   }
1835   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1836   {
1837     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1838     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1839   }
1840 }

```

(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 1841 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1842 {
1843   \keys_define:nn { enumext / #1 }
1844   {
1845     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1846     check-ans .initial:n = false,
1847     check-ans .value_required:n = true,
1848     no-store .code:n = {
1849       \bool_set_false:N \__enumext_check_answers_bool
1850       \bool_set_false:N \__enumext_check_ans_key_bool
1851     },
1852     no-store .value_forbidden:n = true,
1853   }
1854 }
1855 \clist_map_inline:nn
1856 {
1857   level-1, level-2, level-3, level-4, enumext*
1858 }
1859 { \__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.

```
1860 \cs_new_protected:Nn \__enumext_check_ans_active:
1861 {
1862   \tl_if_empty:NF \l__enumext_store_name_tl
1863   {
1864     \bool_if:NT \l__enumext_check_answers_bool
1865     {
1866       \__enumext_check_ans_level:
1867     }
1868   }
1869 }
```

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.

```

1870 \cs_new_protected:Nn \__enumext_check_ans_level:
1871 {
1872   \int_case:nn { \l__enumext_level_int }
1873   {
1874     { 1 }{
1875       \bool_lazy_all:nT
1876       {
1877         { \bool_if_p:N \g__enumext_starred_bool }
1878         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1879       }
1880       {
1881         \int_gdecr:N \g__enumext_item_number_int
1882       }
1883     }
1884     { 2 }{
1885       \int_gdecr:N \g__enumext_item_number_int
1886     }
1887     { 3 }{
1888       \int_gdecr:N \g__enumext_item_number_int
1889     }
1890     { 4 }{
1891       \int_gdecr:N \g__enumext_item_number_int
1892     }
1893   }

```

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.

```

1894   \int_case:nn { \l__enumext_level_h_int }
1895   {
1896     { 1 }{
1897       \bool_lazy_all:nT
1898       {
1899         { \bool_if_p:N \g__enumext_standar_bool }
1900         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1901       }
1902       {
1903         \int_gdecr:N \g__enumext_item_number_int
1904       }
1905     }
1906   }
1907 }

```

(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.

```

1908 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1909 {
1910   \bool_lazy_and:nnT
1911   { \bool_if_p:N \l__enumext_check_ans_key_bool }
1912   { \bool_if_p:N \g__enumext_standar_bool }
1913   {
1914     \bool_gset_true:N \g__enumext_check_ans_key_bool
1915   }
1916   \bool_lazy_and:nnT
1917   { \bool_if_p:N \l__enumext_check_ans_key_bool }
1918   { \bool_if_p:N \g__enumext_starred_bool }
1919   {
1920     \bool_gset_true:N \g__enumext_check_ans_key_bool
1921   }
1922 }

```

(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:`.


```

1923 \cs_new_protected:Nn \__enumext_item_answer_diff:
1924 {
1925     \int_gset:Nn \g__enumext_item_answer_diff_int
1926     {
1927         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1928     }
1929 }

```

(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:.

```

1930 \cs_new_protected:Nn \__enumext_check_ans_show:
1931 {
1932     \int_case:nn { \g__enumext_item_answer_diff_int }
1933     {
1934         { -1 } { \__enumext_check_ans_msg_less: }
1935         { 0 } { \__enumext_check_ans_msg_same_ok: }
1936         { 1 } { \__enumext_check_ans_msg_greater: }
1937     }
1938 }
1939 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1940 {
1941     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1942     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1943 }
1944 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1945 {
1946     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1947     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1948 }
1949 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1950 {
1951     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1952     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1953 }

```

(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:.

```

1954 \cs_new_protected:Nn \__enumext_check_ans_log:
1955 {
1956     \int_case:nn { \g__enumext_item_answer_diff_int }
1957     {
1958         { -1 } { \__enumext_check_ans_log_msg_less: }
1959         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1960         { 1 } { \__enumext_check_ans_log_msg_greater: }
1961     }
1962 }
1963 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1964 {
1965     \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1966     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1967 }
1968 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1969 {
1970     \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1971     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1972 }
1973 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1974 {
1975     \msg_log:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1976     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1977 }

```

(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.

```

1978 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1979 {
1980   \int_compare:nNtT
1981     { \g__enumext_check_starred_cmd_int } = { 0 }
1982     {
1983       \msg_warning:nnnV
1984         { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1985     }
1986   \int_compare:nNtT
1987     { \g__enumext_check_starred_cmd_int } > { 1 }
1988     {
1989       \msg_warning:nnnV
1990         { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1991     }
1992   \int_gzero:N \g__enumext_check_starred_cmd_int
1993   \tl_clear:N \l__enumext_check_start_line_env_tl
1994 }

```

(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.

```

1995 \cs_new_protected:Npn \__enumext_execute_after_env:
1996 {
1997   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
1998   {
1999     \tl_if_empty:NF \g__enumext_store_name_tl
2000     {
2001       \__enumext_stop_save_ans_msg:
2002       \__enumext_item_answer_diff:
2003       \__enumext_log_global_vars:
2004       \__enumext_log_answer_vars:
2005       \bool_if:NTF \g__enumext_check_ans_key_bool
2006       {
2007         \__enumext_check_ans_show:
2008       }
2009       { \__enumext_check_ans_log: }
2010       \__enumext_undefine_anskey_env:
2011     }
2012     \__enumext_reset_global_vars:
2013   }
2014 }

```

• 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*`.

```

2015 \cs_set_protected:Npn \__enumext_tmp:n #1
2016 {
2017   \keys_define:nn { enumext / #1 }
2018   {
2019     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2020     wrap-ans .initial:n = \fbox{##1},
2021     wrap-ans .value_required:n = true,

```

```

2022     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2023     wrap-opt .initial:n = [{##1}],
2024     wrap-opt .value_required:n = true,
2025     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2026     save-sep .initial:n = {, ~ },
2027     save-sep .value_required:n = true,
2028     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2029     mark-ans .initial:n = \textasteriskcentered,
2030     mark-ans .value_required:n = true,
2031     mark-pos .choice:,
2032     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2033     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2034     mark-pos / unknown .code:n =
2035         \msg_error:nneee { enumext } { unknown-choice }
2036         { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2037     mark-pos .initial:n = right,
2038     mark-pos .value_required:n = true,
2039     show-ans .bool_set:N = \__enumext_show_answer_bool,
2040     show-ans .initial:n = false,
2041     show-ans .value_required:n = true,
2042     show-pos .bool_set:N = \__enumext_show_position_bool,
2043     show-pos .initial:n = false,
2044     show-pos .value_required:n = true,
2045     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2046     mark-ref .initial:n = \textasteriskcentered,
2047     mark-ref .value_required:n = true,
2048     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2049     save-ref .initial:n = false,
2050     save-ref .value_required:n = true,
2051 }
2052 }
2053 \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`.

```

2054 \cs_set_protected:Npn \__enumext_tmp:n #1
2055 {
2056     \keys_define:nn { enumext / #1 }
2057     {
2058         mark-pos .choice:,
2059         mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2060         mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2061         mark-pos .initial:n = right,
2062         mark-pos .value_required:n = true,
2063         show-ans .bool_set:N = \__enumext_show_answer_bool,
2064         show-ans .initial:n = false,
2065         show-ans .value_required:n = true,
2066         show-pos .bool_set:N = \__enumext_show_position_bool,
2067         show-pos .initial:n = false,
2068         show-pos .value_required:n = true,
2069     }
2070 }
2071 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

11.26.1 Store optional arguments of the environments

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

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

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

```

2072 \cs_new_protected:Npn \__enumext_store_active_keys:n #1

```

```

2073 {
2074   \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2075   {
2076     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2077     \tl_set:ce
2078       { l__enumext_store_save_key_ \__enumext_level: _tl }
2079       { \__enumext_filter_save_key:n {#1} }
2080   }
2081 }
2082 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2083 {
2084   \bool_if:NF \l__enumext_store_save_key_vii_bool
2085   {
2086     \tl_clear:N \l__enumext_store_save_key_vii_tl
2087     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2088   }
2089 }

```

(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.

```

2090 \cs_set_protected:Npn \__enumext_tmp:n #1
2091 {
2092   \keys_define:nn { enumext / enumext* }
2093   {
2094     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2095     save-key .value_required:n = true,
2096   }
2097   \keys_define:nn { enumext / #1 }
2098   {
2099     save-key .code:n = \__enumext_parse_save_key:n {##1},
2100     save-key .value_required:n = true,
2101   }
2102 }
2103 \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*`.

```

2104 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2105 {
2106   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2107   \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2108   \tl_set:ce
2109     { l__enumext_store_save_key_ \__enumext_level: _tl }
2110     { \__enumext_filter_save_key:n {#1} }
2111 }
2112 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2113 {
2114   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2115   \tl_clear:N \l__enumext_store_save_key_vii_tl
2116   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2117 }

```

(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.

```
2118 \cs_new:Npn \__enumext_filter_save_key:n #1
2119 {
2120   \use:e
2121   {
2122     \keyval_parse:NNn
2123     \__enumext_filter_save_key_key:n
2124     \__enumext_filter_save_key_pair:nn {#1}
2125   }
2126 }
```

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.

```
2127 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2128 {
2129   \str_case:nnF {#1}
2130   {
2131     { resume } {} { resume* } {} { no-store } {}
2132   }
2133   { , { \exp_not:n {#1} } }
2134 }
```

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.

```
2135 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2136 {
2137   \str_case:nnF {#1}
2138   {
2139     { series } {} { resume } {} { save-ans } {}
2140     { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2141     { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2142     { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2143     { mini-right } {} { mini-right* } {}
2144   }
2145   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2146 }
```

(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.

```
2147 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2148 {
2149   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2150   {
2151     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2152   }
2153   { #1 }
2154 }
2155 \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.

```
2156 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2157 {
```

```

2158   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2159 }
2160 \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.

```

2161 \cs_new_protected:Nn \__enumext_store_level_open:
2162 {
2163   \bool_if:NT \l__enumext_check_answers_bool
2164   {
2165     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2166     {
2167       \__enumext_store_addto_seq:n
2168       {
2169         \item \begin{enumext}
2170       }
2171     }
2172   {
2173     \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2174     {
2175       \item \begin{enumext} [
2176     }
2177     \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2178     {
2179       ]
2180     }
2181     \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2182   }
2183 }
2184 }
2185 \cs_new_protected:Nn \__enumext_store_level_close:
2186 {
2187   \bool_if:NT \l__enumext_check_answers_bool
2188   {
2189     \__enumext_store_addto_seq:n { \end{enumext} }
2190   }
2191 }

```

(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.

```

2192 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2193 {
2194   \bool_if:NT \l__enumext_check_answers_bool
2195   {
2196     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2197     {
2198       \__enumext_store_addto_seq:n
2199       {
2200         \item \begin{enumext*}[base-fix]
2201       }
2202     }
2203   {
2204     \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2205     {
2206       \item \begin{enumext*}[base-fix,
2207     }
2208     \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2209     {
2210       ]
2211     }
2212     \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2213   }
2214 }

```

```

2215 }
2216 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2217 {
2218   \bool_if:NT \l__enumext_check_answers_bool
2219   {
2220     \__enumext_store_addto_seq:n { \end{enumext*} }
2221   }
2222 }

```

(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

```

2223 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2224 {
2225   \mode_leave_vertical:
2226   \skip_horizontal:n { -\dim_use:N #2 }
2227   \makebox[0pt][ r ]
2228   {
2229     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2230     {
2231       \tl_use:N \l__enumext_mark_answer_sym_tl
2232     }
2233   }
2234   \skip_horizontal:n { \dim_use:N #2 }
2235 }
2236 \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*.

```

2237 \keys_define:nn { enumext / anskey }
2238 {
2239   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2240   break-col .default:n = true,
2241   break-col .value_forbidden:n = true,
2242   item-join .int_set:N = \l__enumext_store_item_join_int,
2243   item-join .value_required:n = true,
2244   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2245   item-star .default:n = true,
2246   item-star .value_forbidden:n = true,
2247   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2248   item-sym* .value_required:n = true,
2249   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2250   item-pos* .value_required:n = true,
2251 }

```

🔗 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.

```

2252 \NewDocumentCommand \anskey { o +m }
2253 {
2254   \__enumext_anskey_safe_outer:
2255   \group_begin:

```



```

2256     \bool_if:NT \l__enumext_check_answers_bool
2257     {
2258         \int_gincr:N \g__enumext_item_anskey_int
2259         \__enumext_anskey_safe_inner:n {#2}
2260         \__enumext_store_anskey_code:nn {#1} {#2}
2261     }
2262 \group_end:
2263 }

```

(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.

```

2264 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2265 {
2266     \bool_if:NF \l__enumext_store_active_bool
2267     {
2268         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2269     }
2270     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2271     {
2272         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2273     }
2274     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2275     {
2276         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2277     }
2278     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2279     {
2280         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2281     }
2282 }

```

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.

```

2283 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2284 {
2285     \tl_if_empty:nT {#1}
2286     {
2287         \msg_error:nn { enumext } { anskey-empty-arg }
2288     }
2289     \int_incr:N \l__enumext_anskey_level_int
2290     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2291     {
2292         \msg_error:nn { enumext } { anskey-nested }
2293     }
2294 }

```

(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.

```

2295 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2296 {
2297     \__enumext_store_addto_prop:n {#2}
2298     \bool_if:NT \l__enumext_store_ref_key_bool
2299     {
2300         \__enumext_store_internal_ref:
2301     }
2302     \__enumext_anskey_show_wrap_left:n { #2 }

```

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

```

2303     \tl_if_novalue:nF {#1}
2304     {

```

```

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

```

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

```

2315   \bool_lazy_and:nnT
2316     { \bool_not_p:n { \l__enumext_starred_bool } }
2317     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2318   {
2319     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2320       {
2321         ( \exp_not:V \l__enumext_store_item_join_int )
2322       }
2323   }

```

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

```

2324   \bool_if:NTF \l__enumext_store_item_star_bool
2325   {
2326     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2327     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2328     {
2329       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2330       {
2331         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2332       }
2333     }
2334   \dim_compare:nT
2335   {
2336     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2337   }
2338   {
2339     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2340     {
2341       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2342     }
2343   }
2344   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2345 }
2346 {
2347   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2348 }

```

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

```

2349   \bool_lazy_and:nnT
2350     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2351     { \bool_if_p:N \l__enumext_hyperref_bool }
2352   {
2353     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2354     {
2355       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2356       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2357     }
2358   }
2359   \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2360 }

```

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

`\l__enumext_store_internal_ref:` The function `\l__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 $\langle labels \rangle$, we do not want to get double dots in our references, then we will place this in the variable `__enumext_newlabel_arg_two_tl`.

```

2361 \cs_new_protected:Nn \__enumext_store_internal_ref:
2362 {
2363   \cs_set_protected:Npn \__enumext_tmp:n ##1
2364   {
2365     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2366     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2367     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2368     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2369   }
2370   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2371   \cs_set:Npn \__enumext_tmp:n ##1
2372   { . \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.

```

2373 \bool_lazy_all:nT
2374 {
2375   { \bool_if_p:N \g__enumext_starred_bool }
2376   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2377 }
2378 {
2379   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2380   { \tl_use:N \__enumext_label_copy_vii_tl }
2381 }
2382 \bool_lazy_all:nT
2383 {
2384   { \bool_if_p:N \l__enumext_standar_bool }
2385   { \bool_if_p:N \g__enumext_starred_bool }
2386   { \int_compare_p:nNn { \__enumext_level_int } > { 0 } }
2387 }
2388 {
2389   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2390   {
2391     \tl_use:N \__enumext_label_copy_vii_tl
2392     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2393   }
2394 }

```

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.

```

2395 \bool_lazy_all:nT
2396 {
2397   { \bool_if_p:N \l__enumext_standar_bool }
2398   { \int_compare_p:nNn { \__enumext_level_int } > { 0 } }
2399   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2400   { \bool_not_p:n { \__enumext_starred_bool } }
2401 }
2402 {
2403   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2404   {
2405     \tl_use:N \__enumext_label_copy_i_tl
2406     \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
2407   }
2408 }
2409 \cs_set:Npn \__enumext_tmp:n ##1
2410 { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2411 \bool_lazy_all:nT
2412 {
2413   { \bool_if_p:N \l__enumext_standar_bool }
2414   { \int_compare_p:nNn { \__enumext_level_int } > { 0 } }
2415   { \bool_not_p:n { \g__enumext_starred_bool } }
2416   { \int_compare_p:nNn { \__enumext_level_h_int } > { 0 } }
2417 }
2418 {
2419   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2420   {
2421     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2422     . \tl_use:N \__enumext_label_copy_vii_tl
2423   }
2424 }

```

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

```

2425 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2426 {
2427     \l__enumext_store_name_tl \c_colon_str
2428     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2429 }
```

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.

```

2430 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2431 {
2432     \__enumext_newlabel:nn
2433     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2434     { \l__enumext_newlabel_arg_two_tl }
2435 }
2436 \l__enumext_write_aux_file_tl
2437 }
```

(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.

```

2438 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2439 {
2440     \par
2441     \bool_if:NT \l__enumext_starred_bool
2442     {
2443         \cs_set:Nn \__enumext_level: { vii }
2444     }
2445     \__enumext_print_keyans_box:cc
2446     { \l__enumext_labelwidth_ \__enumext_level: _dim }
2447     { \l__enumext_labelsep_ \__enumext_level: _dim }
2448     \__enumext_anskey_wrapper:n { #1 }
2449 }
```

(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.

```

2450 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2451 {
2452     \bool_if:NT \l__enumext_show_answer_bool
2453     {
2454         \__enumext_anskey_show_wrap_arg:n { #1 }
2455     }
2456     \bool_if:NT \l__enumext_show_position_bool
2457     {
2458         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2459         {
2460             \group_begin:
2461             \exp_not:N \normalfont
2462             \exp_not:N \footnotesize [ \int_eval:n
2463             {
2464                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2465             }
2466             ]
2467             \group_end:
2468         }
2469         \__enumext_anskey_show_wrap_arg:n { #1 }
2470     }
2471 }
```

(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.

```

2472 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2473 {
2474   \cs_undefine:c { anskey* }
2475   \cs_undefine:c { endanskey* }
2476   \cs_undefine:c { __scontents_anskey*_env_begin: }
2477   \cs_undefine:c { __scontents_anskey*_env_end: }
2478 }

```

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

```

2479 \__enumext_before_env:nn { enumext }
2480 {
2481   \bool_lazy_and:nnT
2482     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2483     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2484     {
2485       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2486       {
2487         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2488       }
2489     }
2490 }
2491 \__enumext_before_env:nn { enumext* }
2492 {
2493   \bool_lazy_and:nnT
2494     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2495     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2496     {
2497       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2498       {
2499         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2500       }
2501     }
2502 }

```

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

```

2503 \__enumext_before_env:nn { anskey* }
2504 {
2505   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2506   {
2507     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2508   }
2509   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2510   {
2511     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2512   }
2513   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2514   {
2515     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2516   }
2517 }

```

(End of definition for `__enumext_undefine_anskey_env:`.)

anskey*

The function `__enumext_anskey_env_make:n` creates the environment **anskey*** (*custom version of `scontents` environment*) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`. To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec: (§11.24.1)` and we will execute it only if the variable `__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package **scontents**.

```

2518 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2519 {
2520   \bool_if:NT \__enumext_anskey_env_bool
2521   {
2522     \newenvsc{anskey*}[store-env=#1,print-env=false]
2523     \__enumext_anskey_env_exec:
2524   }
2525 }
2526 \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`.

```

2527 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2528 {
2529   \keys_define:nn { scontents / scontents }
2530   {
2531     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2532     break-col .default:n = true,
2533     break-col .value_forbidden:n = true,
2534     item-join .int_gset:N = \g__enumext_store_item_join_int,
2535     item-join .value_required:n = true,
2536     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2537     item-star .default:n = true,
2538     item-star .value_forbidden:n = true,
2539     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2540     item-sym* .value_required:n = true,
2541     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2542     item-pos* .value_required:n = true,
2543     print-env .undefine:,
2544     store-env .undefine:,
2545     write-out .undefine:,
2546   }
2547 }

```

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`.

```

2548 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2549 {
2550   \keys_define:nn { scontents / scontents }
2551   {
2552     break-col .undefine:,
2553     item-join .undefine:,
2554     item-star .undefine:,
2555     item-sym* .undefine:,
2556     item-pos* .undefine:,
2557     write-out .code:n = {
2558       \bool_set_false:N \l__scontents_storing_bool
2559       \bool_set_true:N \l__scontents_writing_bool
2560       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2561     },
2562     write-out .value_required:n = true,
2563     print-env .meta:nn = { scontents } { print-env = ##1 },
2564     print-env .default:n = true,
2565     store-env .meta:nn = { scontents } { store-env = ##1 },
2566     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} }
2567   }
2568 }

```

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*.

```

2569 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2570 {
2571   \group_begin:
2572   \int_set:Nn \tex_newlinechar:D { ``^^J }
2573   \__scontents_rescan_tokens:x
2574   {
2575     \endgroup % This assumes \catcode``=0... Things might go off otherwise.
2576     #1
2577   }
2578 }

```

(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)*.

```

2579 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2580 {
2581   \__enumext_before_env:nn { anskey* }
2582   {
2583     \__enumext_anskey_env_define_keys:
2584   }

```

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.

```

2585   \hook_if_empty:nF {env/anskey*/after}
2586   {
2587     \hook_gremove_code:nn {env/anskey*/after} { * }
2588   }
2589   \__enumext_after_env:nn { anskey* }
2590   {
2591     \tl_clear:N \l__enumext_store_anskey_env_tl
2592     \tl_clear:N \l__enumext_store_anskey_opt_tl
2593     \tl_gset:Ne \l__enumext_store_anskey_env_tl
2594     {
2595       \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2596     }
2597     \__enumext_anskey_env_keys:
2598     \__enumext_anskey_env_store:
2599     \__enumext_anskey_env_clean:
2600     \__enumext_anskey_env_undefine_keys:
2601   }
2602 }

```

🔗 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`.

`__enumext_anskey_env_store:`

`__enumext_anskey_env_clean:`

```

2603 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2604 {
2605   \bool_lazy_and:nnT
2606   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2607   { \bool_not_p:n { \l__enumext_starred_bool } }
2608   {
2609     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2610   }

```

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

```

2611   \bool_lazy_and:nnT
2612   { \bool_not_p:n { \l__enumext_starred_bool } }
2613   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2614   {
2615     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2616     {
2617       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2618     }
2619   }

```


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

```

2620   \bool_if:NT \g__enumext_store_item_star_bool
2621   {
2622     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2623     {
2624       ,item-star,
2625     }
2626     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2627     {
2628       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2629       {
2630         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2631       }
2632     }
2633     \dim_compare:nT
2634     {
2635       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2636     }
2637     {
2638       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2639       {
2640         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2641       }
2642     }
2643   }
2644 }

```

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`.

```

2645 \cs_new_protected:Nn \__enumext_anskey_env_store:
2646 {
2647   \group_begin:
2648   \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2649   {
2650     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2651     {
2652       \exp_args:Ne
2653       \anskey
2654       {
2655         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2656       }
2657     }
2658     {
2659       \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2660       \exp_args:Ne
2661       \anskey
2662       {
2663         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2664       }
2665     }
2666   }
2667   \group_end:
2668 }

```

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

```

2669 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2670 {
2671   \bool_gset_false:N \g__enumext_store_columns_break_bool
2672   \int_gzero:N       \g__enumext_store_item_join_int
2673   \bool_gset_false:N \g__enumext_store_item_star_bool
2674   \tl_gclear:N       \g__enumext_store_item_symbol_tl
2675   \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2676 }

```

(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 $\langle label \rangle$ `\l__enumext_label_v_tl` for the `keyans` environment and the current $\langle label \rangle$ `\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 $\langle prop list \rangle$ defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

2677 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2678 {
2679   \tl_clear:N \l__enumext_store_current_label_tl
2680   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2681   {
2682     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2683   }
2684   {
2685     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2686   }
2687   \tl_if_novalue:nF { #1 }
2688   {
2689     % Set save-sep
2690     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2691     {
2692       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2693     }
2694     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2695   }
2696   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2697 }

```

(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 $\langle label \rangle$. 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 $\langle labels \rangle$ and remove the dots “.” from them, we do not want to get double dots in our references.

```

2698 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2699 {
2700   \bool_if:NT \l__enumext_store_ref_key_bool
2701   {
2702     \cs_set_protected:Npn \__enumext_tmp:n ##1
2703     {
2704       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2705       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2706       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2707       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2708     }
2709     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2710     \__enumext_keyans_store_ref_aux_i:
2711   }
2712 }

```

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

```

2713 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2714 {
2715   \bool_if:NT \g__enumext_starred_bool
2716   {
2717     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2718   }
2719   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2720   {
2721     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

2722     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2723   }
2724   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2725   {
2726     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2727     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2728   }
2729   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2730   {
2731     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2732     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2733   }
2734   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2735   {
2736     \l__enumext_store_name_tl \c_colon_str
2737     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2738   }
2739   \__enumext_keyans_store_ref_aux_ii:
2740 }

```

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.

```

2741 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2742 {
2743   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2744   {
2745     \__enumext_newlabel:nn
2746     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2747     { \l__enumext_newlabel_arg_two_tl }
2748   }
2749   \l__enumext_write_aux_file_tl
2750 }

```

(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.

```

2751 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2752 {
2753   \tl_clear:N \l__enumext_store_current_label_tl
2754   \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
2755   {
2756     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2757   }
2758   {
2759     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2760   }
2761   \tl_if_novalue:nF { #1 }
2762   {
2763     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2764     {
2765       \tl_put_right:Ne \l__enumext_store_current_label_tl
2766       {
2767         \l__enumext_store_keyans_item_opt_sep_tl
2768       }
2769     }
2770     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2771   }
2772   \__enumext_keyans_addto_seq_link:
2773 }

```

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.

```

2774 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2775 {
2776   \bool_lazy_and:nnT
2777     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2778     { \bool_if_p:N \l__enumext_hyperref_bool }
2779     {
2780       \tl_put_right:Ne \l__enumext_store_current_label_tl
2781       {
2782         \hfill \exp_not:N \hyperlink
2783         {
2784           \exp_not:V \l__enumext_newlabel_arg_one_tl
2785         }
2786         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2787       }
2788     }
2789     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2790     \bool_if:NT \l__enumext_check_answers_bool
2791     {
2792       \int_gincr:N \g__enumext_item_anskey_int
2793     }
2794 }

```

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

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

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

```

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

```

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

```

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

```

```

2834     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2835     {
2836         \group_begin:
2837         \exp_not:N \normalfont
2838         \exp_not:N \footnotesize [ \int_eval:n
2839         {
2840             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2841         }
2842         ]
2843         \group_end:
2844     }
2845 }
2846 {
2847     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2848     {
2849         \group_begin:
2850         \exp_not:N \normalfont
2851         \exp_not:N \footnotesize [ \int_eval:n
2852         {
2853             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2854         }
2855         ]
2856         \group_end:
2857     }
2858 }
2859 \tl_put_left:Nn \l__enumext_label_v_tl
2860 {
2861     \__enumext_print_keyans_box:NN
2862     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2863 }
2864 }

```

(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*`.

```

2865 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2866 {
2867     \keys_define:nn { enumext / #1 }
2868     {
2869         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2870         item-sym* .value_required:n = true,
2871         item-sym* .initial:n = { $\star$ },
2872         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2873         item-pos* .value_required:n = true,
2874     }
2875 }
2876 \clist_map_inline:nn
2877 {
2878     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2879 }
2880 { \__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](#).

```

2881 \cs_new_protected:Nn \__enumext_footnotetext:nn
2882 {
2883     \footnotetext[#1]{#2}
2884 }
2885 \cs_new_protected:Nn \__enumext_renew_footnote:
2886 {
2887     \seq_gclear:N \g__enumext_footnote_arg_seq
2888     \seq_gclear:N \g__enumext_footnote_int_seq

```

```

2889 \RenewDocumentCommand \footnote { o +m }
2890 {
2891   \tl_if_novalue:nTF {##1}
2892   {
2893     \stepcounter{footnote}
2894     \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2895   }
2896   {
2897     \int_gset:Nn \g__enumext_footnote_int { ##1 }
2898   }
2899   \footnotemark [ \g__enumext_footnote_int ]
2900   \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2901   \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2902 }
2903 }
2904 \cs_new_protected:Nn \__enumext_print_footnote:
2905 {
2906   \seq_if_empty:NF \g__enumext_footnote_int_seq
2907   {
2908     \seq_map_pairwise_function:NNN
2909     \g__enumext_footnote_int_seq
2910     \g__enumext_footnote_arg_seq
2911     \__enumext_footnotetext:nn
2912   }
2913 }

```

(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)`.

```

2914 \cs_new_protected:Npn \__enumext_default_item:n #1
2915 {
2916   \tl_if_novalue:nTF {##1}
2917   {
2918     \bool_if:NT \l__enumext_check_answers_bool
2919     {
2920       \int_gincr:N \g__enumext_item_number_int
2921     }
2922     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2923     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2924   }
2925   {
2926     \bool_set_eq:cc
2927     { l__enumext_wrap_label_ \__enumext_level: _bool }
2928     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2929     \__enumext_item_std:w [##1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2930   }
2931 }

```

(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_#_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_#_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_#_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_#_bool` and the vars `\l__enumext_item_symbol_#_sep_#_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label:` (§11.33).

```

2932 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2933 {
2934   \tl_if_novalue:nF {#1}
2935   {
2936     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2937   }
2938   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2939   \tl_if_novalue:nTF {#2}
2940   {
2941     \dim_set_eq:cc
2942     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2943     { \__enumext_labelsep_ \__enumext_level: _dim }
2944   }
2945   {
2946     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2947   }
2948   \bool_if:NT \l__enumext_check_answers_bool
2949   {
2950     \int_gincr:N \g__enumext_item_number_int
2951   }
2952   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2953   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2954 }

```

(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).

```

2955 \cs_new_protected:Nn \__enumext_redefine_item:
2956 {
2957   \RenewDocumentCommand \item { s o o }
2958   {
2959     \bool_if:nTF {##1}
2960     {
2961       \__enumext_starred_item:nn {##2} {##3}
2962     }
2963     { \__enumext_default_item:n {##2} }
2964   }
2965 }

```

(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`.

```

2966 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2967 {
2968   \tl_if_novalue:nTF { #1 }
2969   {
2970     \bool_set_true:N \__enumext_wrap_label_v_bool
2971     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```



```

2972     }
2973     {
2974         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2975         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2976     }
2977 }

```

(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 $\langle label \rangle$.

```

2978 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2979 {
2980     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
2981     \__enumext_keyans_show_left:n { #1 }
2982     \bool_set_true:N \l__enumext_wrap_label_v_bool
2983     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

2984     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
2985     \__enumext_keyans_addto_prop:n { #1 }
2986     \__enumext_keyans_store_ref:
2987     \__enumext_keyans_addto_seq:n { #1 }
2988     \int_gincr:N \g__enumext_check_starred_cmd_int
2989 }

```

(End of definition for __enumext_keyans_starred_item:n.)

\item*
__enumext_keyans_redefine_item:

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

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

```

2990 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2991 {
2992     \RenewDocumentCommand \item { s o }
2993     {
2994         \bool_if:nTF {##1}
2995         {
2996             \peek_remove_spaces:n
2997             {
2998                 \__enumext_keyans_starred_item:n {##2}
2999             }
3000         }
3001         {
3002             \__enumext_keyans_default_item:n {##2}
3003         }
3004     }
3005 }

```

(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.

```

3006 \cs_new_protected:Nn \__enumext_item_starred:
3007 {
3008     \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
3009     {
3010         \mode_leave_vertical:
3011         \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }

```

```

3012         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
3013         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3014     }
3015 }

```

(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).

```

3016 \cs_new_protected:Nn \__enumext_make_label:
3017 {
3018     \RenewDocumentCommand \makelabel { m }
3019     {
3020         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3021         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3022         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3023         {
3024             \__enumext_item_starred:
3025             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3026         }
3027         { ##1 }
3028         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3029         \tl_gclear:N \g__enumext_item_symbol_tl
3030     }
3031 }

```

(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).

```

3032 \cs_new_protected:Nn \__enumext_keyans_make_label:
3033 {
3034     \RenewDocumentCommand \makelabel { m }
3035     {
3036         \tl_use:N \l__enumext_label_fill_left_v_tl
3037         \tl_use:N \l__enumext_label_font_style_v_tl
3038         \bool_if:NTF \l__enumext_wrap_label_v_bool
3039         {
3040             \__enumext_wrapper_label_v:n { ##1 }
3041         }
3042         { ##1 }
3043         \tl_use:N \l__enumext_label_fill_right_v_tl
3044     }
3045 }

```

(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.

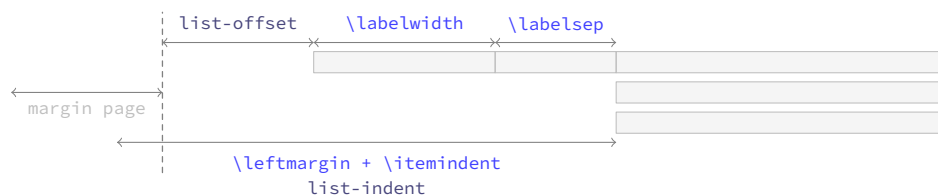
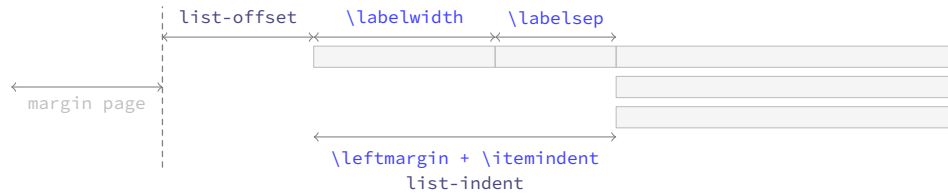
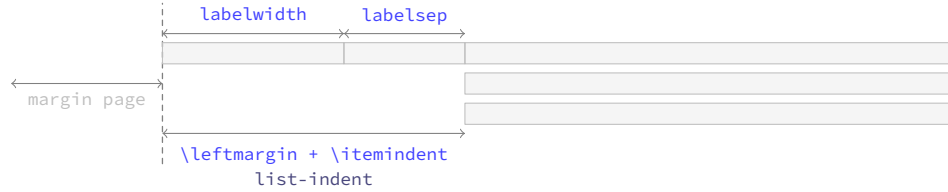


Figure 9: Representation of standard horizontal lengths in `list` environment.

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`,

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

so that the left edge of the *label box* is at $\text{\leftmargin} + \text{\itemindent}$ minus $\text{\labelwidth} + \text{\labelsep}$. 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.

Figure 11: Default horizontal lengths in `enumext`.

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

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

```
#1: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__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).

```
3046 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3047 {
3048   \dim_compare:nNt { #1 } < { \c_zero_dim }
3049   {
3050     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3051     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3052   }
3053   \dim_compare:nNt { #2 } < { \c_zero_dim }
3054   {
3055     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3056     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3057   }
3058 }
```

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

```
3058 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3059 \dim_compare:nNtF { #4 } < { \c_zero_dim }
3060 {
3061   \dim_set:Nn #6 { #1 + #2 - #4 }
3062   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3063 }
3064 {
3065   \dim_compare:nNt { #4 } = { #1 + #2 }
3066   { \dim_set:Nn #6 { \c_zero_dim } }
3067   \dim_compare:nNt { #4 } < { #1 + #2 }
3068   { \dim_set:Nn #6 { #1 + #2 - #4 } }
3069   \dim_compare:nNt { #4 } > { #1 + #2 }
3070   {
3071     \dim_set:Nn #6 { -#1 - #2 + #4 }
3072     \dim_set:Nn #6 { #6*-1 }
3073   }
3074   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3075 }
3076 }
3077 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

11.34.2 Setting second argument of the lists

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

```

3078 \cs_set_protected:Npn \__enumext_tmp:n #1
3079 {
3080   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3081   {
3082     \__enumext_calc_hspace:cccccc
3083     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3084     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3085     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3086     { l__enumext_leftmargin_tmp_#1_bool }
3087     \clist_map_inline:nn
3088       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3089       { \dim_set_eq:cc {###1} { l__enumext_###1_#1_dim } }
3090     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3091       { \skip_set_eq:cc {###1} { l__enumext_###1_#1_skip } }
3092     \usecounter { enumX#1 }
3093     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3094     \str_if_eq:nnTF {#1} { v }
3095     {
3096       \__enumext_keyans_redefine_item:
3097       \__enumext_keyans_make_label:
3098       \__enumext_keyans_ref:
3099       \__enumext_keyans_fake_item:
3100       \bool_if:cT { l__enumext_show_length_#1_bool }
3101       {
3102         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3103       }
3104     }
3105     {
3106       \__enumext_redefine_item:
3107       \__enumext_make_label:
3108       \__enumext_standar_ref:
3109       \__enumext_fake_item:
3110       \bool_if:cT { l__enumext_show_length_#1_bool }
3111       {
3112         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_int }
3113       }
3114     }
3115   }
3116 }
3117 \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.

```

3118 \cs_set_protected:Npn \__enumext_tmp:n #1
3119 {
3120   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3121   {
3122     \__enumext_calc_hspace:cccccc
3123     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3124     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3125     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3126     { l__enumext_leftmargin_tmp_#1_bool }
3127     \clist_map_inline:nn
3128       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3129       { \dim_set_eq:cc {###1} { l__enumext_###1_#1_dim } }
3130     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3131       { \skip_set_eq:cc {###1} { l__enumext_###1_#1_skip } }
3132     \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3133     \skip_zero:N \partopsep
3134     \usecounter { enumX#1 }
3135     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3136     \__enumext_starred_ref:
3137     \str_if_eq:nnTF {#1} { vii }

```

```

3138         {
3139             \__enumext_fake_item_vii:
3140             \bool_if:cT { l__enumext_show_length_vii_bool }
3141             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3142         }
3143         {
3144             \__enumext_fake_item_viii:
3145             \bool_if:cT { l__enumext_show_length_#1_bool }
3146             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3147         }
3148     }
3149 }
3150 \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.

```

3151 \NewDocumentEnvironment{enumext}{0}{}
3152 {
3153     \__enumext_safe_exec:
3154     \__enumext_parse_keys:n {#1}
3155     \__enumext_before_list:
3156     \__enumext_start_store_level:
3157     \__enumext_start_list:nn
3158     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3159     {
3160         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3161         \__enumext_before_keys_exec:
3162     }
3163     \__enumext_after_args_exec:
3164 }
3165 {
3166     \__enumext_stop_list:
3167     \__enumext_stop_store_level:
3168     \__enumext_after_list:
3169 }

```

(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:**.

```

3170 \cs_new_protected:Nn \__enumext_safe_exec:
3171 {
3172     \__enumext_internal_mini_page:
3173     \__enumext_is_not_nested:
3174     \int_incr:N \l__enumext_level_int
3175     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3176     { \msg_fatal:nn { enumext } { list-too-deep } }
3177     \bool_set_true:N \l__enumext_standar_bool
3178     \__enumext_is_on_first_level:
3179 }

```

(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**.

```

3180 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3181 {
3182     \tl_if_novalue:nF {#1}
3183     {
3184         \str_clear:N \l__enumext_series_str

```

```

3185         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3186         {
3187             \keys_set:nn { enumext / level-1 } {#1}
3188             \__enumext_parse_series:n {#1}
3189             \__enumext_nested_base_line_fix:
3190         }
3191         {
3192             \exp_args:Ne \keys_set:nn
3193             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3194         }
3195     \__enumext_store_active_keys:n {#1}
3196 }
3197 }

```

(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.

```

3198 \cs_new_protected:Nn \__enumext_start_store_level:
3199 {
3200     \bool_lazy_all:nT
3201     {
3202         { \bool_if_p:N \l__enumext_store_active_bool }
3203         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3204         { \bool_not_p:n { \g__enumext_starred_bool } }
3205     }
3206     {
3207         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3208         {
3209             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3210             \__enumext_store_level_open:
3211         }
3212     }
3213     \bool_lazy_all:nT
3214     {
3215         { \bool_if_p:N \l__enumext_store_active_bool }
3216         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3217         { \bool_if_p:N \g__enumext_starred_bool }
3218     }
3219     {
3220         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3221         {
3222             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3223             \__enumext_store_level_open:
3224         }
3225     }
3226 }
3227 \cs_new_protected:Nn \__enumext_stop_store_level:
3228 {
3229     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3230     {
3231         \__enumext_store_level_close:
3232     }
3233 }

```

(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.

```

3234 \cs_new_protected:Nn \__enumext_before_list:
3235 {
3236     \__enumext_vspace_above:
3237     \__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.

```

3238     \__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.

```

3239 \dim_compare:nNt
3240 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3241 {
3242   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3243   {
3244     \linewidth
3245     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3246     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3247   }

```

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*.

```

3248 \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3249 \int_gincr:N \g__enumext_minipage_stat_int
3250 \__enumext_mini_addvspace:
3251 \nointerlineskip\noindent
3252 \begin{__enumext_mini_env*}
3253   { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3254 }
3255 \__enumext_multicols_start:
3256 }

```

(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.

```

3257 \cs_new_protected:Nn \__enumext_multicols_start:
3258 {
3259   \int_compare:nNt
3260   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3261   {
3262     \dim_compare:nNt
3263     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3264     {
3265       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3266       {
3267         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3268         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3269         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3270         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3271       }
3272     }
3273     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3274     \skip_zero:N \multicolsep
3275     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3276     {
3277       \dim_zero:N \columnseprule
3278     }

```

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.

```

3279 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3280 {
3281   \__enumext_multi_addvspace:
3282 }
3283 \raggedcolumns

```



```

3284         \begin{multicols}{\int_use:c { \l__enumext_columns_ \l__enumext_level: _int } }
3285     }
3286 }

```

(End of definition for `\l__enumext_multicols_start:`)

`\l__enumext_multicols_stop:` The function `\l__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.

```

3287 \cs_new_protected:Nn \l__enumext_multicols_stop:
3288 {
3289     \int_compare:nNtT
3290     { \int_use:c { \l__enumext_columns_ \l__enumext_level: _int } } > { 1 }
3291     {
3292         \end{multicols}
3293         \bool_if:cF { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3294         {
3295             \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \l__enumext_level: _skip } }
3296         }
3297     }
3298 }

```

(End of definition for `\l__enumext_multicols_stop:`)

`\l__enumext_after_list:` The function `\l__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.

```

3299 \cs_new_protected:Nn \l__enumext_after_list:
3300 {
3301     \bool_if:cTF { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3302     {
3303         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3304         {
3305             \msg_warning:nn { enumext } { missing-miniright }
3306             \miniright
3307         }
3308         \int_gzero:N \g__enumext_minipage_stat_int
3309         \end{__enumext_mini_env*}
3310         \par\addvspace { \l__enumext_minipage_after_skip }
3311     }
3312     { \l__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`.

```

3313     \l__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.

```

3314     \l__enumext_after_stop_list:
3315     \l__enumext_vspace_below:
3316     \bool_set_false:N \l__enumext_standar_bool
3317     \l__enumext_resume_save_counter:
3318 }

```

(End of definition for `\l__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 `\l__enumext_after_env:nn`.

```

3319 \l__enumext_after_env:nn {enumext} { \l__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.

```

3320 \NewDocumentEnvironment{keyans}{ 0{ } }
3321 {
3322   \__enumext_keyans_safe_exec:
3323   \__enumext_keyans_parse_keys:n {#1}
3324   \__enumext_before_list_v:
3325   \__enumext_start_list:nn
3326   { \tl_use:N \__enumext_label_v_tl }
3327   {
3328     \__enumext_list_arg_two_v:
3329     \__enumext_before_keys_exec_v:
3330   }
3331   \__enumext_after_args_exec_v:
3332 }
3333 {
3334   \__enumext_check_starred_cmd:n { item }
3335   \__enumext_stop_list:
3336   \__enumext_after_list_v:
3337 }

```

(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.

```

3338 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3339 {
3340   \bool_if:NF \l__enumext_store_active_bool
3341   {
3342     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3343   }
3344   \int_incr:N \l__enumext_keyans_level_int
3345   \bool_set_true:N \l__enumext_keyans_env_bool
3346   \__enumext_keyans_start_line:
3347   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3348   \bool_set_false:N \l__enumext_store_active_bool
3349   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3350   {
3351     \msg_error:nn { enumext } { keyans-nested }
3352   }
3353   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3354   {
3355     \msg_error:nn { enumext } { keyans-wrong-level }
3356   }
3357 }

```

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

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

```

3358 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3359 {
3360   \keys_set:nn { enumext / keyans } {#1}
3361 }

```

(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.

```

3362 \cs_new_protected:Nn \__enumext_before_list_v:
3363 {
3364   \__enumext_vspace_above_v:
3365   \__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.

```

3366   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3367   {
3368     \dim_set:Nn \l__enumext_minipage_left_v_dim

```

```

3369         {
3370             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3371         }

```

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*.

```

3372         \bool_set_true:N \l__enumext_minipage_active_v_bool
3373         \int_gincr:N \g__enumext_minipage_stat_int
3374         \__enumext_keyans_mini_addvspace:
3375         \nointerlineskip\noindent
3376         \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3377     }

```

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

```

3378     \__enumext_keyans_multicols_start:
3379 }

```

(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.

```

3380 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3381 {
3382     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3383     {

```

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

```

3384         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3385         {
3386             \dim_set:Nn \l__enumext_columns_sep_v_dim
3387             {
3388                 (
3389                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3390                 ) / \l__enumext_columns_v_int
3391                 - \l__enumext_listoffset_v_dim
3392             }
3393         }
3394         \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).

```

3395         \skip_zero:N \multicolsep
3396         \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.

```

3397         \bool_if:NF \l__enumext_minipage_active_v_bool
3398         {
3399             \__enumext_keyans_multi_addvspace:
3400         }
3401         \raggedcolumns
3402         \begin{multicols}{ \l__enumext_columns_v_int }
3403     }
3404 }

```

(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.

```

3405 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3406 {
3407     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3408     {
3409         \end{multicols}
3410         \bool_if:NF \l__enumext_minipage_active_v_bool

```

```

3411         {
3412         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3413         }
3414     }
3415 }

```

(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.

```

3416 \cs_new_protected:Nn \__enumext_after_list_v:
3417 {
3418     \bool_if:NTF \l__enumext_minipage_active_v_bool
3419     {
3420         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3421         {
3422             \msg_warning:nn { enumext } { missing-miniright }
3423             \miniright
3424         }
3425         \int_gzero:N \g__enumext_minipage_stat_int
3426         \end{\__enumext_mini_env*}
3427         \par\addvspace{ \l__enumext_minipage_after_skip }
3428     }
3429     { \__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.

```

3430     \bool_set_false:N \l__enumext_keyans_env_bool
3431     \__enumext_after_stop_list_v:
3432     \__enumext_vspace_below_v:
3433 }

```

(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.

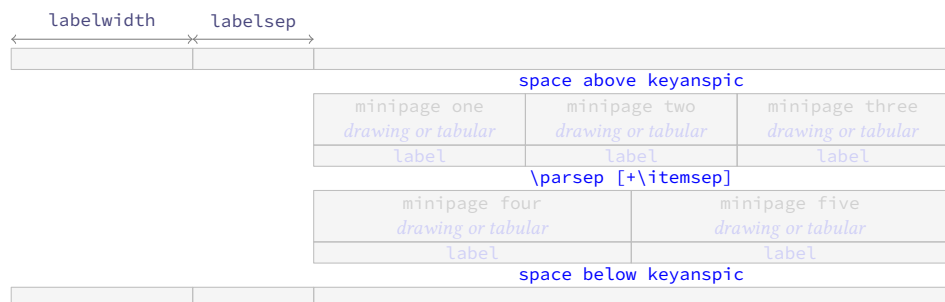


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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*[\langle content \rangle]` store the current `\label` next to the `[\langle content \rangle]` if it is present in the `\langle sequence \rangle` and `\langle prop list \rangle` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```

3434 \NewDocumentCommand \anspic { s o +m }
3435 {

```

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

```

3436 \bool_if:NF \__enumext_store_active_bool
3437 {
3438   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3439 }
3440 \int_compare:nNt { \__enumext_level_int } > { 1 }
3441 {
3442   \msg_error:nn { enumext } { keyanspic-wrong-level }
3443 }
3444 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
3445 {
3446   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3447 }

```

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.

```

3448 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3449 {
3450   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3451 }
3452 }

```

(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.

```

3453 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3454 {
3455   \stepcounter { enumXvi }
3456   #3 \\\
3457   \bool_if:nT { #1 }
3458   {
3459     \__enumext_keyans_addto_prop:n { #2 }
3460     \__enumext_keyans_store_ref:
3461     \__enumext_keyans_addto_seq:n { #2 }
3462     \int_gincr:N \__enumext_check_starred_cmd_int
3463     \bool_lazy_or:nnT
3464       { \bool_if_p:N \__enumext_show_answer_bool }
3465       { \bool_if_p:N \__enumext_show_position_bool }
3466     {
3467       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3468       \__enumext_keyans_show_left:n { #2 }
3469       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3470     }
3471   }
3472   \tl_use:N \__enumext_label_font_style_v_tl
3473   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3474 }

```

(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.

```

3475 \NewDocumentEnvironment{keyanspic}{ o }
3476 {
3477   \__enumext_keyans_pic_safe_exec:
3478   \__enumext_start_list:nn
3479   { }
3480   {
3481     \__enumext_keyans_pic_arg_two:
3482   }

```

We apply the “adjusted” vertical spacing above the environment

```

3483 \vspace { \__enumext_keyans_pic_above_skip }
3484 }

```

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.

```

3485 {
3486   \tl_if_novalue:nTF { #1 }
3487   {
3488     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3489   }
3490   { \__enumext_keyans_pic_do:n { #1 } }
3491   \__enumext_stop_list:
3492   \__enumext_check_starred_cmd:n { anspic }
3493   \setcounter { enumXvi } { 0 }
3494   \vspace { \l__enumext_topsep_v_skip }
3495   %\bool_set_false:N \l__enumext_store_active_bool
3496 }

```

(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.

```

3497 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3498 {
3499   \int_incr:N \l__enumext_keyans_pic_level_int
3500   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3501   {
3502     \msg_error:nn { enumext } { keyanspic-nested }
3503   }
3504   \__enumext_keyans_start_line:
3505 }

```

(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`.

```

3506 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3507 {
3508   \dim_compare:nNnT { #1 } < { 0pt }
3509   { \skip_set:Nn #1 { -#1 } }
3510 }

```

(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.

```

3511 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3512 {

```

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.

```

3513   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3514   \__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.

```

3515   \skip_add:Nn \parsep { \itemsep }
3516   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3517   \dim_zero:N \labelwidth
3518   \dim_zero:N \listparindent
3519   \dim_zero:N \labelsep
3520   \skip_zero:N \partopsep
3521   \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.

```

3522   \__enumext_keyans_pic_skip_abs:N \parsep
3523   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3524   {

```

```

3525         \box_dp:N \strutbox
3526         + \l__enumext_topsep_v_skip
3527         - \parsep
3528     }
3529     \__enumext_item_std:w \scan_stop:
3530 }

```

(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`.

```

3531 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3532 {
3533     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3534 }
3535 \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.

```

3536 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3537 {
3538     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3539     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3540     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3541     \int_step_inline:nnn
3542     { \l__enumext_keyans_pic_above_int + 1 }
3543     { \l__enumext_keyans_pic_below_int }
3544     {
3545         \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3546         \centering
3547         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3548         \__enumext_endminipage:
3549     }
3550     \par
3551 }

```

(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.

```

3552 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3553 {
3554     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3555     {
3556         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3557         {
3558             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3559             / \l__enumext_columns_vii_int
3560         }
3561     }
3562     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3563     \dim_set:Nn \l__enumext_item_width_vii_dim
3564     {
3565         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )

```



```

3566         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3567         - \l__enumext_labelsep_vii_dim
3568     }
3569     \dim_zero_new:N \itemwidth
3570 }
3571 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3572 {
3573     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3574     {
3575         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3576         {
3577             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3578             / \l__enumext_columns_viii_int
3579         }
3580     }
3581     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3582     \dim_set:Nn \l__enumext_item_width_viii_dim
3583     {
3584         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3585         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3586         - \l__enumext_labelsep_viii_dim
3587     }
3588     \dim_zero_new:N \itemwidth
3589 }

```

(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.

```

3590 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3591 {
3592     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3593     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3594     {
3595         \msg_warning:nnee { enumext } { item-joined }
3596         { \int_use:N \l__enumext_joined_item_vii_int }
3597         { \int_use:N \l__enumext_columns_vii_int }
3598         \int_set:Nn \l__enumext_joined_item_vii_int
3599         {
3600             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3601         }
3602     }
3603     \int_compare:nNnT
3604     { \l__enumext_joined_item_vii_int }
3605     >
3606     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3607     {
3608         \msg_warning:nnee { enumext } { item-joined-columns }
3609         { \int_use:N \l__enumext_joined_item_vii_int }
3610         {
3611             \int_eval:n
3612             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3613         }
3614         \int_set:Nn \l__enumext_joined_item_vii_int
3615         {
3616             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3617         }
3618     }
3619     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3620     {
3621         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3622         \int_decr:N \l__enumext_joined_item_aux_vii_int
3623         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3624         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3625         \dim_set:Nn \l__enumext_joined_width_vii_dim
3626         {
3627             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3628             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3629                 + \l__enumext_columns_sep_vii_dim

```

```

3630         )*\l__enumext_joined_item_aux_vii_int
3631     }
3632     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3633 }
3634 {
3635     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3636     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3637 }
3638 }
3639 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3640 {
3641     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3642     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3643     {
3644         \msg_warning:nnee { enumext } { item-joined }
3645         { \int_use:N \l__enumext_joined_item_viii_int }
3646         { \int_use:N \l__enumext_columns_viii_int }
3647         \int_set:Nn \l__enumext_joined_item_viii_int
3648         {
3649             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3650         }
3651     }
3652     \int_compare:nNnT
3653     { \l__enumext_joined_item_viii_int }
3654     >
3655     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3656     {
3657         \msg_warning:nnee { enumext } { item-joined-columns }
3658         { \int_use:N \l__enumext_joined_item_viii_int }
3659         {
3660             \int_eval:n
3661             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3662         }
3663         \int_set:Nn \l__enumext_joined_item_viii_int
3664         {
3665             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3666         }
3667     }
3668     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3669     {
3670         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3671         \int_decr:N \l__enumext_joined_item_aux_viii_int
3672         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3673         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3674         \dim_set:Nn \l__enumext_joined_width_viii_dim
3675         {
3676             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3677             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3678                 + \l__enumext_columns_sep_viii_dim
3679             )*\l__enumext_joined_item_aux_viii_int
3680         }
3681         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3682     }
3683     {
3684         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3685         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3686     }
3687 }

```

(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 `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

3688 \cs_new_protected:Nn \__enumext_start_mini_vii:
3689 {
3690     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3691     {
3692         \dim_set:Nn \l__enumext_minipage_left_vii_dim

```

```

3693     {
3694         \linewidth
3695         - \l__enumext_minipage_right_vii_dim
3696         - \l__enumext_minipage_hsep_vii_dim
3697     }
3698     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3699     \dim_gset_eq:NN
3700         \g__enumext_minipage_right_vii_dim
3701         \l__enumext_minipage_right_vii_dim
3702     \__enumext_mini_addvspace_vii:
3703     \nointerlineskip\noindent
3704     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3705 }
3706 }

```

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”.

```

3707 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3708 {
3709     \bool_if:NT \l__enumext_minipage_active_vii_bool
3710     {
3711         \end{__enumext_mini_env*}
3712         \hfill
3713         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3714     }
3715 }

```

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.

```

3716 \__enumext_after_env:nn {enumext*}
3717 {
3718     \bool_if:NT \g__enumext_minipage_active_vii_bool
3719     {
3720         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3721         \par\addvspace { \g__enumext_minipage_right_skip }
3722         \bool_if:NF \g__enumext_minipage_center_vii_bool
3723         {
3724             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3725             {
3726                 \centering
3727             }
3728         }
3729         \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3730         {
3731             \tl_use:N \g__enumext_miniright_code_vii_tl
3732         }
3733         \box_use_drop:N \l__enumext_miniright_code_vii_box
3734         \end{__enumext_mini_env*}
3735         \par\addvspace{ \g__enumext_minipage_after_skip }
3736     }
3737     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3738     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3739     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3740     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3741     \bool_gset_false:N \g__enumext_starred_bool
3742 }

```

(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.

```

3743 \cs_new_protected:Nn \__enumext_start_mini_viii:
3744 {
3745     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3746     {
3747         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3748         {

```

```

3749         \linewidth
3750         - \l__enumext_minipage_right_viii_dim
3751         - \l__enumext_minipage_hsep_viii_dim
3752     }
3753     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3754     \dim_gset_eq:NN
3755         \g__enumext_minipage_right_viii_dim
3756         \l__enumext_minipage_right_viii_dim
3757     \__enumext_mini_addvspace_viii:
3758     \nointerlineskip\noindent
3759     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3760 }
3761 }
3762 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3763 {
3764     \bool_if:NT \l__enumext_minipage_active_viii_bool
3765     {
3766         \end{__enumext_mini_env*}
3767         \hfill
3768         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3769     }
3770 }
3771 \__enumext_after_env:nn {keyans*}
3772 {
3773     \bool_if:NT \g__enumext_minipage_active_viii_bool
3774     {
3775         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3776         \par\addvspace { \g__enumext_minipage_right_skip }
3777         \bool_if:NF \g__enumext_minipage_center_viii_bool
3778         {
3779             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3780             {
3781                 \centering
3782             }
3783         }
3784         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3785         {
3786             \tl_use:N \g__enumext_miniright_code_viii_tl
3787         }
3788         \box_use_drop:N \l__enumext_miniright_code_viii_box
3789         \end{__enumext_mini_env*}
3790         \par\addvspace{ \g__enumext_minipage_after_skip }
3791     }
3792     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3793     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3794     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3795     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3796 }

```

(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.

```

3797 \NewDocumentEnvironment{enumext*}{ o }
3798 {
3799     \__enumext_safe_exec_vii:
3800     \__enumext_parse_keys_vii:n {#1}
3801     \__enumext_before_list_vii:
3802     \__enumext_start_store_level_vii:
3803     \__enumext_start_list:nn { }
3804     {
3805         \__enumext_list_arg_two_vii:
3806         \__enumext_before_keys_exec_vii:
3807     }
3808     \__enumext_starred_columns_set_vii:
3809     \item[] \scan_stop:
3810     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3811     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3812 }

```

```

3813 {
3814     \__enumext_stop_item_tmp_vii:
3815     \__enumext_remove_extra_parsep_vii:
3816     \__enumext_stop_list:
3817     \__enumext_stop_store_level_vii:
3818     \__enumext_after_list_vii:
3819 }

```

(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`.

```

3820 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3821 {
3822     \__enumext_internal_mini_page:
3823     \__enumext_is_not_nested:
3824     \int_incr:N \l__enumext_level_h_int
3825     \int_compare:nNtT { \l__enumext_level_h_int } > { 1 }
3826     {
3827         \msg_error:nn { enumext } { nested }
3828     }
3829     \bool_set_true:N \l__enumext_starred_bool
3830     \__enumext_is_on_first_level:
3831 }

```

(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>`.

```

3832 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3833 {
3834     \tl_if_novalue:nF {#1}
3835     {
3836         \str_clear:N \l__enumext_series_str
3837         \keys_set:nn { enumext / enumext* } {#1}
3838         \__enumext_parse_series:n {#1}
3839         \__enumext_store_active_keys_vii:n {#1}
3840         \__enumext_nested_base_line_fix:
3841     }
3842 }

```

(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`.

```

3843 \cs_new_protected:Nn \__enumext_before_list_vii:
3844 {
3845     \__enumext_vspace_above_vii:
3846     \__enumext_check_ans_active:
3847     \__enumext_before_args_exec_vii:
3848     \__enumext_start_mini_vii:
3849 }

```

(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.

```

3850 \cs_new_protected:Nn \__enumext_after_list_vii:
3851 {
3852     \__enumext_stop_mini_vii:
3853     \__enumext_after_stop_list_vii:
3854     \__enumext_check_ans_key_hook:
3855     \__enumext_vspace_below_vii:
3856     \bool_set_false:N \l__enumext_starred_bool
3857     \__enumext_resume_save_counter:
3858 }

```

(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`.

```

3859 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3860 {
3861   \bool_if:NT \l__enumext_store_active_bool
3862   {
3863     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3864     {
3865       \__enumext_store_level_open_vii:
3866     }
3867   }
3868 }
3869 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3870 {
3871   \bool_if:NT \l__enumext_store_active_bool
3872   {
3873     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3874     {
3875       \__enumext_store_level_close_vii:
3876     }
3877   }
3878 }

```

(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`.

```

3879 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3880 {
3881   \__enumext_stop_item_tmp_vii:
3882   \int_incr:N \l__enumext_item_column_pos_vii_int
3883   \int_gincr:N \g__enumext_item_count_all_vii_int
3884   \__enumext_item_peek_args_vii:
3885 }

```

(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).

```

3886 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3887 {
3888   \peek_meaning:NTF (
3889     { \__enumext_joined_item_vii:w }
3890     { \__enumext_joined_item_vii:w (1) }
3891   }

```

(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`.

```

3892 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3893 {
3894   \__enumext_starred_joined_item_vii:n {#1}
3895   \peek_meaning_remove:NTF *
3896     { \__enumext_starred_item_vii:w }
3897     { \__enumext_standar_item_vii:w }
3898 }

```

(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].

```

3899 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3900 {
3901   \bool_set_false:N \l__enumext_item_starred_vii_bool
3902   \peek_meaning:NTF [
3903     {
3904       \bool_set_eq:NN
3905         \l__enumext_wrap_label_vii_bool
3906         \l__enumext_wrap_label_opt_vii_bool
3907       \__enumext_start_item_vii:w
3908     }
3909     {
3910       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3911       \legacy_if_set_true:n { @noitemarg }
3912       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3913     }
3914   }

```

(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

```

3915 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3916 {
3917   \bool_set_true:N \l__enumext_item_starred_vii_bool
3918   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3919   \peek_meaning:NTF [
3920     { \__enumext_starred_item_vii_aux_i:w }
3921     { \__enumext_starred_item_vii_aux_ii:w }
3922   }
3923   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3924   {
3925     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3926     \__enumext_starred_item_vii_aux_ii:w
3927   }
3928   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3929   {
3930     \peek_meaning:NTF [
3931       { \__enumext_starred_item_vii_aux_iii:w }
3932       {
3933         \dim_set_eq:NN
3934           \l__enumext_item_symbol_sep_vii_dim
3935           \l__enumext_labelsep_vii_dim
3936         \legacy_if_set_true:n { @noitemarg }
3937         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3938       }
3939     }
3940   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3941   {
3942     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3943     \legacy_if_set_true:n { @noitemarg }
3944     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3945   }

```

(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.

```

3946 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3947 {
3948   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3949   \legacy_if:nT { @noitemarg }
3950   {
3951     \legacy_if_set_false:n { @noitemarg }
3952     \legacy_if:nT { @nmbrlist }
3953     {
3954       \bool_if:NT \l__enumext_hyperref_bool
3955       {
3956         \legacy_if_set_true:n { @hyper@item }
3957       }
3958       \refstepcounter{enumXvii}
3959       \bool_if:NT \l__enumext_check_answers_bool
3960       {
3961         \int_gincr:N \g__enumext_item_number_int
3962       }
3963     }
3964   }

```

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

```

3965   \group_begin:
3966   \lrbox{ \l__enumext_item_text_vii_box }
3967   \bool_if:NF \l__enumext_footnotes_key_bool
3968   {
3969     \__enumext_renew_footnote:
3970   }
3971   \bool_if:NT \l__enumext_item_starred_vii_bool
3972   {
3973     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3974     {
3975       \tl_gset_eq:NN
3976       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3977     }
3978     \mode_leave_vertical:
3979     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3980     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3981     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3982     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3983   }
3984   \group_begin:
3985   \tl_use:N \l__enumext_label_font_style_vii_tl
3986   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3987   {
3988     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3989     { \__enumext_wrapper_label_vii:n {#1} }
3990   }
3991   {
3992     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3993   }
3994   \group_end:
3995   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3996   \tl_use:N \l__enumext_after_list_args_vii_tl
3997   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3998   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3999   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4000   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4001 }

```

(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.

```

4002 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4003 {
4004     \__enumext_endminipage:
4005     \endlrbox
4006     \group_end:
4007     \box_set_wd:Nn \l__enumext_item_text_vii_box
4008     {
4009         \l__enumext_joined_width_vii_dim
4010         + \l__enumext_labelwidth_vii_dim
4011         + \l__enumext_labelsep_vii_dim
4012     }
4013     \int_set:Nn \hbadness { 10000 }
4014     \box_use_drop:N \l__enumext_item_text_vii_box
4015     \bool_if:NF \l__enumext_footnotes_key_bool
4016     {
4017         \__enumext_print_footnote:
4018     }
4019     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4020     {
4021         \par\noindent
4022         \int_zero:N \l__enumext_item_column_pos_vii_int
4023     }
4024     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4025 }

```

(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.

```

4026 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4027 {
4028     \int_compare:nNnT
4029     {
4030         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4031     }
4032     =
4033     { 0 }
4034     {
4035         \par
4036         \vspace{ -\l__enumext_itemsep_vii_skip }
4037         \int_gzero:N \g__enumext_item_count_all_vii_int
4038     }
4039 }

```

(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`.

```

4040 \__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.

```

4041 \NewDocumentEnvironment{keyans*}{ o }
4042 {
4043     \__enumext_safe_exec_viii:
4044     \__enumext_parse_keys_viii:n {#1}
4045     \__enumext_before_list_viii:
4046     \__enumext_start_list:nn { }
4047     {
4048         \__enumext_list_arg_two_viii:
4049         \__enumext_before_keys_exec_viii:
4050     }
4051     \__enumext_starred_columns_set_viii:
4052     \item[] \scan_stop:
4053     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4054     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4055 }

```

```

4056 {
4057   \__enumext_stop_item_tmp_viii:
4058   \__enumext_remove_extra_parsep_viii:
4059   \__enumext_check_starred_cmd:n { item }
4060   \__enumext_stop_list:
4061   \__enumext_after_list_viii:
4062 }

```

(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.

```

4063 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4064 {
4065   \int_incr:N \l__enumext_keyans_level_h_int
4066   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4067   {
4068     \msg_error:nn { enumext } { nested }
4069   }
4070   \__enumext_keyans_start_line:
4071   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4072   \bool_set_false:N \l__enumext_store_active_bool
4073   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4074   {
4075     \msg_error:nn { enumext } { keyans-wrong-level }
4076   }
4077 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

4078 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4079 {
4080   \tl_if_novalue:nF {#1}
4081   {
4082     \keys_set:nn { enumext / keyans* } {#1}
4083   }
4084 }

```

(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`.

```

4085 \cs_new_protected:Nn \__enumext_before_list_viii:
4086 {
4087   \__enumext_vspace_above_viii:
4088   \__enumext_before_args_exec_viii:
4089   \__enumext_start_mini_viii:
4090 }

```

(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.

```

4091 \cs_new_protected:Nn \__enumext_after_list_viii:
4092 {
4093   \__enumext_stop_mini_viii:
4094   \__enumext_after_stop_list_viii:
4095   \__enumext_vspace_below_viii:
4096 }

```

(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`.

```
4097 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4098 {
4099     \__enumext_stop_item_tmp_viii:
4100     \int_incr:N \l__enumext_item_column_pos_viii_int
4101     \int_gincr:N \g__enumext_item_count_all_viii_int
4102     \__enumext_item_peek_args_viii:
4103 }
```

(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).

```
4104 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4105 {
4106     \peek_meaning:NTF (
4107         { \__enumext_joined_item_viii:w }
4108         { \__enumext_joined_item_viii:w (1) }
4109     }
```

(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`.

```
4110 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4111 {
4112     \__enumext_starred_joined_item_viii:n {#1}
4113     \peek_meaning_remove:NTF *
4114         { \__enumext_starred_item_viii:w }
4115         { \__enumext_standar_item_viii:w }
4116 }
```

(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]`.

```
4117 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4118 {
4119     \bool_set_false:N \l__enumext_item_starred_viii_bool
4120     \peek_meaning:NTF [
4121         {
4122             \bool_set_eq:NN
4123             \l__enumext_wrap_label_viii_bool
4124             \l__enumext_wrap_label_opt_viii_bool
4125             \__enumext_start_item_viii:w
4126         }
4127         {
4128             \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

4129         \legacy_if_set_true:n { @noitemarg }
4130         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4131     }
4132 }

```

(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].

```

4133 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4134 {
4135     \bool_set_true:N \__enumext_item_starred_viii_bool
4136     \bool_set_true:N \__enumext_wrap_label_viii_bool
4137     \peek_meaning:NTF [
4138         { \__enumext_starred_item_viii_aux_i:w }
4139         { \__enumext_starred_item_viii_aux_ii:w }
4140     }

```

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.

```

4141 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4142 {
4143     \tl_clear:N \l__enumext_store_current_label_tl
4144     \tl_if_no_value:nF { #1 }
4145     {
4146         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4147         {
4148             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4149             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4150         }
4151         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4152     }
4153     \__enumext_starred_item_viii_aux_ii:w
4154 }
4155 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4156 {
4157     \legacy_if_set_true:n { @noitemarg }
4158     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4159 }

```

(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.

```

4160 \cs_new_protected:Nn \__enumext_starred_item_exec:
4161 {
4162     \tl_put_left:Ne \l__enumext_store_current_label_tl { \__enumext_label_viii_tl }
4163     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4164     \__enumext_keyans_store_ref:
4165     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4166     \__enumext_keyans_addto_seq_link:
4167     \int_gincr:N \g__enumext_check_starred_cmd_int
4168     \bool_if:NT \l__enumext_show_answer_bool
4169     {
4170         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4171     }
4172     \bool_if:NT \l__enumext_show_position_bool
4173     {
4174         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4175         {
4176             \group_begin:
4177             \exp_not:N \normalfont
4178             \exp_not:N \footnotesize [ \int_eval:n
4179                 {
4180                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4181                 }

```

```

4182         ]
4183     \group_end:
4184 }
4185 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
4186 }
4187 }

```

(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.

```

4188 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4189 {
4190     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4191     \legacy_if:nT { @noitemarg }
4192     {
4193         \legacy_if_set_false:n { @noitemarg }
4194         \legacy_if:nT { @nmbrlist }
4195         {
4196             \bool_if:NT \l__enumext_hyperref_bool
4197             {
4198                 \legacy_if_set_true:n { @hyper@item }
4199             }
4200             \refstepcounter{enumXviii}
4201         }
4202     }

```

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

```

4203     \group_begin:
4204     \lrbox{ \l__enumext_item_text_viii_box }
4205     \bool_if:NF \l__enumext_footnotes_key_bool
4206     {
4207         \__enumext_renew_footnote:
4208     }
4209     \bool_if:NT \l__enumext_item_starred_viii_bool
4210     {
4211         \__enumext_starred_item_exec:
4212     }
4213     \group_begin:
4214     \tl_use:N \l__enumext_label_font_style_viii_tl
4215     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4216     {
4217         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4218         { \__enumext_wrapper_label_viii:n {#1} }
4219     }
4220     {
4221         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4222     }
4223     \group_end:
4224     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4225     \tl_use:N \l__enumext_after_list_args_viii_tl
4226     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4227     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4228     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4229     \bool_if:NT \l__enumext_item_starred_viii_bool
4230     {
4231         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4232         \__enumext_keyans_show_item_opt:
4233         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4234     }
4235     {
4236         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4237     }
4238 }

```

(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.

```

4239 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:

```

```

4240 {
4241   \__enumext_endminipage:
4242   \endlrbox
4243   \group_end:
4244   \box_set_wd:Nn \l__enumext_item_text_viii_box
4245   {
4246     \l__enumext_joined_width_viii_dim
4247     + \l__enumext_labelwidth_viii_dim
4248     + \l__enumext_labelsep_viii_dim
4249   }
4250   \int_set:Nn \hbadness { 10000 }
4251   \box_use_drop:N \l__enumext_item_text_viii_box
4252   \bool_if:NF \l__enumext_footnotes_key_bool
4253   {
4254     \__enumext_print_footnote:
4255   }
4256   \int_compare:nNnTF
4257   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4258   {
4259     \par\noindent
4260     \int_zero:N \l__enumext_item_column_pos_viii_int
4261   }
4262   { \hspace{ \l__enumext_columns_sep_viii_dim } }
4263 }

```

(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.

```

4264 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4265 {
4266   \int_compare:nNnT
4267   {
4268     \int_mod:nn
4269     { \g__enumext_item_count_all_viii_int }
4270     { \l__enumext_columns_viii_int }
4271   }
4272   =
4273   { 0 }
4274   {
4275     \par
4276     \vspace{ -\l__enumext_itemsep_viii_skip }
4277     \int_gzero:N \g__enumext_item_count_all_viii_int
4278   }
4279 }

```

(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.

```

4280 \NewDocumentCommand \getkeyans { m }
4281 {
4282   \exp_args:Ne \__enumext_getkeyans_aux:n
4283   { \tl_to_str:e { \text_expand:n {#1} } }
4284 }

```

(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.

```

4285 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4286 {
4287   \str_if_in:nnTF {#1} { : }
4288   {
4289     \use:e
4290     {
4291       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4292       { {##1} {##2} }
4293     }

```



```

4294         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4295     }
4296     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4297 }

```

(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⟩*.

```

4298 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4299 {
4300     \prop_if_exist:cF { g__enumext_#1_prop }
4301     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4302     \group_begin:
4303     \prop_item:cn { g__enumext_#1_prop }{#2}
4304     \group_end:
4305 }

```

(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⟩]

```

4306 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4307 \keys_define:nn { enumext / print }
4308 {
4309     print* .code:n = \keys_precompile:neN { enumext / enumext* }
4310                 { \__enumext_filter_save_key:n {#1} }
4311                 \__enumext_print_keyans_starred_tl, % starred cmd
4312     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4313     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4314                 { \__enumext_filter_save_key:n {#1} }
4315                 \__enumext_print_keyans_i_tl,
4316     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4317     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4318                 { \__enumext_filter_save_key:n {#1} }
4319                 \__enumext_print_keyans_ii_tl,
4320     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4321     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4322                 { \__enumext_filter_save_key:n {#1} }
4323                 \__enumext_print_keyans_iii_tl,
4324     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4325     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4326                 { \__enumext_filter_save_key:n {#1} }
4327                 \__enumext_print_keyans_iv_tl,
4328     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4329     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4330                 { \__enumext_filter_save_key:n {#1} }
4331                 \__enumext_print_keyans_vii_tl, % starred nested
4332     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4333 }

```

🔗 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.

```

4334 \NewDocumentCommand \printkeyans { s O{} m }
4335 {
4336     \group_begin:

```

```

4337 \tl_use:N \l__enumext_print_keyans_i_tl
4338 \tl_use:N \l__enumext_print_keyans_ii_tl
4339 \tl_use:N \l__enumext_print_keyans_iii_tl
4340 \tl_use:N \l__enumext_print_keyans_iv_tl
4341 \tl_use:N \l__enumext_print_keyans_vii_tl
4342 \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4343 \group_end:
4344 }

```

(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.

```

4345 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4346 {
4347   \seq_if_exist:cTF { g__enumext_#3_seq }
4348   {
4349     \seq_if_empty:cF { g__enumext_#3_seq }
4350     {
4351       %%\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`

```

4352   \bool_if:nTF {#1}
4353   {
4354     \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4355     {
4356       \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4357     }
4358     {
4359       \tl_use:N \l__enumext_print_keyans_starred_tl
4360       \begin{enumext*}[#2]
4361       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4362       \end{enumext*}
4363     }
4364   }

```

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

```

4365   {
4366     \begin{enumext}[#2]
4367     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4368     \end{enumext}
4369   }
4370 }
4371 }
4372 {
4373   \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4374 }
4375 }

```

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

11.43 The command `\setenumext`

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

```

4376 \keys_define:nn { enumext / meta-families }
4377 {
4378   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4379   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4380   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4381   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4382   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4383   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4384   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4385   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4386   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4387   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4388   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4389   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,

```

```

4390     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4391     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4392 }

```

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

```

4393 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4394 {
4395     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4396     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4397 }

```

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

```

4398 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4399 {
4400     \tl_if_novalue:nTF {#1}
4401     {
4402         \seq_map_inline:Nn \c__enumext_all_families_seq
4403     }
4404     {
4405         \seq_clear:N \l__enumext_setkey_tmpa_seq
4406         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4407         \int_set:Nn \l__enumext_setkey_tmpa_int
4408         {
4409             \seq_count:N \l__enumext_setkey_tmpb_seq
4410         }
4411         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4412         {
4413             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4414             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4415             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4416             {
4417                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4418             }
4419         }
4420         {
4421             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4422         }
4423         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4424         { \seq_map_inline:Nn \c__enumext_all_families_seq }
4425         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4426     }
4427     {
4428         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4429     }
4430 }

```

(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.

```

4431 \cs_new_protected:Npn \__enumext_set_parse:n #1
4432 {
4433     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4434     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } %<- max level
4435     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4436     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4437     {
4438         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4439         { \tl_trim_spaces:n {#1} }
4440     }
4441     { \__enumext_set_error:nn {#1} { } }
4442 }
4443 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4444 { \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.

```

4445 \msg_new:nnn { enumext } { package-load }
4446 {
4447   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4448 }
4449 \msg_new:nnn { enumext } { package-not-load }
4450 {
4451   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4452 }
4453 \msg_new:nnn { enumext } { package-load-foot }
4454 {
4455   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4456 }
```

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

```

4457 \msg_new:nnn { enumext } { counters }
4458 {
4459   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4460   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4461 }
```

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

```

4462 \msg_new:nnn { enumext } { unknown-choice }
4463 {
4464   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4465 }
4466 % \begin{macrocode}
4467 %
4468 % Message used by reserved \myenv*{anskey*} environment by \mypkg*{enumext} package.
4469 % \begin{macrocode}
4470 \msg_new:nnnn { enumext } { anskey-env-error }
4471 {
4472   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4473   'enumext' ~ package, ~ It~ is~ already~ defined.
4474 }
4475 {
4476   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4477   for ~ the ~ 'save-ans' ~ key.\\
4478 }
```

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

```

4479 \msg_new:nnn { enumext } { store-prop }
4480 {
4481   * ~ Package ~ enumext: ~ Creating ~
4482   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4483 }
4484 \msg_new:nnn { enumext } { store-seq }
4485 {
4486   * ~ Package ~ enumext: ~ Creating ~
4487   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4488 }
4489 \msg_new:nnn { enumext } { store-int }
4490 {
4491   * ~ Package ~ enumext: ~ Creating ~
4492   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4493 }
4494 \msg_new:nnn { enumext } { prop-seq-int-hook }
4495 {
4496   * ~ Package ~ enumext: ~ Elements ~ in ~
4497   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4498   * ~ Package ~ enumext: ~ Elements ~ in ~
4499   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4500   * ~ Package ~ enumext: ~ Value ~ off ~
4501   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4502 }
4503 \msg_new:nnn { enumext } { item-answer-hook }
4504 {
4505   * ~ Package ~ enumext: ~ Value ~ off ~
4506   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4507   * ~ Package ~ enumext: ~ Value ~ off ~
4508   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
```

```

4509     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4510 }

```

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

```

4511 \msg_new:nnn { enumext } { invalid-key }
4512 {
4513     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4514 }
4515 \msg_new:nnn { enumext } { unknown-key-family }
4516 {
4517     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4518 }

```

Messages used in length calculation.

```

4519 \msg_new:nnn { enumext } { width-negative }
4520 {
4521     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4522     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4523 }
4524 \msg_new:nnn { enumext } { width-zero }
4525 {
4526     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4527     The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4528 }

```

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

```

4529 \msg_new:nnn { enumext } { list-lengths }
4530 {
4531     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4532     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4533     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4534     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4535     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4536     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4537     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4538     \__enumext_show_length:nnn { skip } { topsep } {#1}
4539     \__enumext_show_length:nnn { skip } { parsep } {#1}
4540     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4541     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4542     ****
4543 }

```

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

```

4544 \msg_new:nnn { enumext } { list-lengths-not-nested }
4545 {
4546     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4547     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4548     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4549     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4550     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4551     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4552     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4553     \__enumext_show_length:nnn { skip } { topsep } {#1}
4554     \__enumext_show_length:nnn { skip } { parsep } {#1}
4555     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4556     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4557     ****
4558 }

```

Messages used by `ref` key.

```

4559 \msg_new:nnn { enumext } { key-ref-empty }
4560 {
4561     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4562 }

```

Messages used by `save-ans` key.

```

4563 \msg_new:nnn { enumext } { save-ans-empty }
4564 {
4565     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4566 }
4567 \msg_new:nnn { enumext } { save-ans-log }
4568 {
4569     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4570 }

```

```

4571 \msg_new:nnn { enumext } { save-ans-log-hook }
4572 {
4573   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4574 }
4575 \msg_new:nnn { enumext } { save-ans-hook }
4576 {
4577   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4578 }

```

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

```

4579 \msg_new:nnn { enumext } { need-save-ans }
4580 {
4581   Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4582 }
4583 \msg_new:nnn { enumext } { items-same-answer }
4584 {
4585   *****\\
4586   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4587   for ~ \c_left_brace_str #2 \c_right_brace_str\\
4588   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4589   'OK', ~ all ~ items ~ with ~ answer.\\
4590   *****
4591 }
4592 \msg_new:nnn { enumext } { item-greater-answer }
4593 {
4594   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4595   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4596   Items ~ > ~ Answers.
4597 }
4598 \msg_new:nnn { enumext } { item-less-answer }
4599 {
4600   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4601   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4602   Items ~ < ~ Answers.
4603 }

```

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

```

4604 \msg_new:nnn { enumext } { missing-starred }
4605 {
4606   Missing ~ '\c_backslash_str #1*' ~ #2.
4607 }
4608 \msg_new:nnn { enumext } { many-starred }
4609 {
4610   Many ~ '\c_backslash_str #1*' ~ #2.
4611 }

```

Messages used by `\printkeyans*` command.

```

4612 \msg_new:nnn { enumext } { print-starred }
4613 {
4614   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4615   #2 ~ environment ~ \msg_line_context:.
4616 }

```

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

```

4617 \msg_new:nnn { enumext } { list-too-deep }
4618 {
4619   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4620   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4621 }

```

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

```

4622 \msg_new:nnn { enumext } { anskey-empty-arg }
4623 {
4624   Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4625 }
4626 \msg_new:nnn { enumext } { anskey-wrong-place }
4627 {
4628   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4629   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4630 }
4631 \msg_new:nnn { enumext } { anskey-nested }
4632 {
4633   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.

```

```

4634 }
4635 \msg_new:nnn { enumext } { anskey-env-wrong }
4636 {
4637   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4638 }
4639 \msg_new:nnn { enumext } { anspic-wrong-place }
4640 {
4641   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4642   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4643 }
4644 \msg_new:nnn { enumext } { command-wrong-place }
4645 {
4646   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4647   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4648 }

```

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

```

4649 \msg_new:nnn { enumext } { keyans-nested }
4650 {
4651   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4652 }
4653 \msg_new:nnn { enumext } { keyans-wrong-level }
4654 {
4655   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4656   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4657 }
4658 \msg_new:nnn { enumext } { wrong-place }
4659 {
4660   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4661   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4662 }
4663 \msg_new:nnn { enumext } { keyanspic-nested }
4664 {
4665   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4666 }
4667 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4668 {
4669   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4670   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4671 }

```

Messages used by `\getkeyans` command.

```

4672 \msg_new:nnn { enumext } { undefined-storage-anskey }
4673 {
4674   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4675 }

```

Messages used by `\miniright` command.

```

4676 \msg_new:nnn { enumext } { missing-miniright }
4677 {
4678   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4679   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4680 }
4681 \msg_new:nnn { enumext } { wrong-miniright-place }
4682 {
4683   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4684   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4685 }
4686 \msg_new:nnn { enumext } { wrong-miniright-use }
4687 {
4688   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4689   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4690 }

```

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

```

4691 \msg_new:nnn { enumext } { nested }
4692 {
4693   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4694 }
4695 \msg_new:nnn { enumext } { item-joined }
4696 {
4697   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4698 }

```



```
4699 \msg_new:nnn { enumext } { item-joined-columns }  
4700 {  
4701   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4702 }
```

11.45 Finish package

Finish package implementation.

```
4703 \file_input_stop:  
4704 </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>	209
<code>\+</code>	201
<code>\-</code>	201
<code>\\</code>	217, 2575, 3456, 4459, 4472, 4477, 4497, 4499, 4506, 4508, 4521, 4526, 4531, 4546, 4585, 4587, 4589, 4594, 4595, 4600, 4601, 4619, 4628, 4641, 4646, 4655, 4660, 4669, 4678, 4683, 4688
A	
above	<u>1422</u>
above*	<u>1422</u>
<code>\addvspace</code>	1076, 1104, 1220, 1299, 1362, 1368, 1396, 1413, 3295, 3310, 3412, 3427, 3721, 3735, 3776, 3790
after	<u>915</u>
align	<u>490</u>
<code>\Alph</code>	35, 39, <u>40</u>
<code>\Alph</code>	442, 557, 602, 670, 4328
<code>\alph</code>	35, 39, <u>40</u>
<code>\alph</code>	443, 555, 4320
<code>\anskey</code>	12, 71, <u>2252</u> , 2653, 2661
anskey*	13, <u>2518</u>
<code>\anspic</code>	15, 96, <u>3434</u>
<code>\anspic*</code>	66
<code>\arabic</code>	29, 35
<code>\arabic</code>	441, 554, 601, 4312, 4316, 4332
B	
base-fix	<u>794</u>
<code>\baselineskip</code>	48
<code>\baselineskip</code>	811, 822
before	<u>915</u>
before*	<u>915</u>
below	<u>1422</u>
below*	<u>1422</u>
bool commands:	
<code>\bool_gset_false:N</code>	317, 318, 319, 2671, 2673, 3737, 3741, 3792
<code>\bool_gset_true:N</code>	229, 238, 1018, 1914, 1920, 3713, 3738, 3768, 3793
<code>\bool_if:NTF</code>	382, 394, 411, 1444, 1458, 1471, 1482, 1493, 1504, 1515, 1526, 1579, 1596, 1601, 1609, 1636, 1674, 1679, 1686, 1690, 1712, 1717, 1725, 1732, 1763, 1771, 1864, 2005, 2074, 2084, 2163, 2187, 2194, 2218, 2256, 2266, 2298, 2324, 2441, 2452, 2456, 2520, 2620, 2700, 2715, 2790, 2801, 2805, 2918, 2948, 3022, 3038, 3100, 3110, 3140, 3145, 3229, 3279, 3293, 3301, 3340, 3397, 3410, 3418, 3436, 3709, 3718, 3722, 3764, 3773, 3777, 3861, 3871, 3954, 3959, 3967, 3971, 3986, 4015, 4168, 4172, 4196, 4205, 4209, 4215, 4229, 4252
<code>\bool_if:nTF</code>	1397, 1414, 2959, 2994, 3058, 3457, 4352
<code>\bool_if_p:N</code>	247, 262, 807, 808, 818, 819, 1743, 1744, 1752, 1753, 1877, 1899, 1911, 1912, 1917, 1918, 2309, 2350, 2351, 2375, 2384, 2385, 2397, 2413, 2606, 2777, 2778, 2815, 2816, 3202, 3215, 3217, 3464, 3465
<code>\bool_lazy_all:nTF</code>	245, 260, 1875, 1897, 2373, 2382, 2395, 2411, 3200, 3213
<code>\bool_lazy_and:nnTF</code>	225, 234, 806, 817, 1742, 1751, 1910, 1916, 2308, 2315, 2349, 2481, 2493, 2605, 2611, 2776
<code>\bool_lazy_or:nnTF</code>	1804, 1811, 2814, 3463
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 63, 72, 93, 98, 99, 104, 105, 108, 129, 130, 138, 139, 145, 146, 160, 171, 173
<code>\bool_not_p:n</code>	226, 235, 2310, 2316, 2400, 2415, 2607, 2612, 3203, 3204, 3216
<code>\bool_set_eq:NN</code>	2926, 2974, 3904, 4122
<code>\bool_set_false:N</code>	391, 828, 1849, 1850, 2558, 3316, 3348, 3430, 3495, 3513, 3856, 3901, 4072, 4119
<code>\bool_set_true:N</code>	252, 253, 267, 268, 373, 377, 483, 843, 1428, 1433, 1699, 1821, 1822, 2106, 2114, 2559, 2922, 2952, 2970, 2982, 3177, 3209, 3222, 3248, 3345, 3372, 3698, 3753, 3829, 3910, 3917, 3918, 4128, 4135, 4136
box commands:	
<code>\box_dp:N</code>	1116, 1120, 1124, 1135, 1139, 1150, 1159, 1165, 1175, 1188, 1194, 1200, 1231, 1232, 1233, 1236, 1246, 1250, 1259, 1266, 1271, 1279, 1308, 1309, 1312, 1319, 1332, 1340, 1346, 1354, 3525
<code>\box_new:N</code>	69, 166, 172
<code>\box_set_wd:Nn</code>	4007, 4244
<code>\box_use_drop:N</code>	3733, 3788, 4014, 4251
<code>\box_wd:N</code>	449
C	
<code>\c</code>	209, 210, 707, 709, 721, 723
<code>\catcode</code>	2575
<code>\cB</code>	210
<code>\cE</code>	210
<code>\centering</code>	1399, 1416, 3546, 3726, 3781
check-ans	<u>1841</u>
Document class:	
article	41
clist commands:	
<code>\clist_const:Nn</code>	178
<code>\clist_map_function:nN</code>	3533
<code>\clist_map_inline:Nn</code>	489, 749, 848, 914, 929, 1010, 1438
<code>\clist_map_inline:nn</code>	48, 59, 77, 83, 95, 107, 132, 154, 177, 517, 537, 803, 853, 1024, 1544, 1788, 1855, 2053, 2071, 2103, 2370, 2709, 2876, 3087, 3090, 3117, 3127, 3130, 3150, 4434
<code>\columnbreak</code>	72
<code>\columnbreak</code>	2312
columns	<u>994</u>
columns-sep	<u>994</u>
<code>\columnsep</code>	92, 95
<code>\columnsep</code>	3273, 3394
<code>\columnseprule</code>	92, 95
<code>\columnseprule</code>	3277, 3396
Commands provide by enumext:	
<code>\anskey</code>	27, 62, 63, 68, 69, 71, 73–75, 80, 82, 91, 105, 113, 114, 119
<code>\anspic*</code>	27, 66, 69, 80, 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, 80, 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

\printkeyans*	114	\dim_compare:nNnTF	856, 872, 884, 896, 1388, 1407, 3048, 3053, 3059, 3065, 3067, 3069, 3239, 3262, 3366, 3384, 3508, 3554, 3573, 3690, 3745
\printkeyans	27, 69, 114	\dim_compare:nTF	2334, 2633
\setenumext	27, 114–116, 118	\dim_gset_eq:NN	3699, 3754
Counters defined by enumext :		\dim_gzero:N	2675, 3740, 3795
enumXiii	25, 34	\dim_new:N	66, 73, 74, 75, 92, 134, 167, 168, 174
enumXii	25, 34	\dim_set:Nn	449, 844, 2946, 3051, 3056, 3058, 3061, 3062, 3066, 3068, 3071, 3072, 3074, 3242, 3265, 3368, 3386, 3538, 3556, 3563, 3575, 3582, 3625, 3674, 3692, 3747, 3942
enumXiv	25, 34	\dim_set_eq:NN	545, 592, 663, 667, 2941, 3089, 3129, 3273, 3394, 3632, 3635, 3636, 3681, 3684, 3685, 3933
enumXi	25, 34	\dim_use:N	857, 865, 1389, 1395, 2226, 2229, 2234, 3011, 3013, 3240, 3245, 3246, 3253, 3263, 3267, 3268, 3270
enumXviii	25, 34	\dim_zero:N	3277, 3396, 3517, 3518, 3519
enumXvii	25, 34, 106	\dim_zero_new:N	3569, 3588
enumXvi	25, 34	\c_zero_dim	859, 873, 885, 897, 1389, 1407, 2336, 2635, 3048, 3053, 3059, 3066, 3240, 3263, 3366, 3384, 3554, 3573, 3690, 3745
enumXv	25, 34		
cs commands:		E	
\cs_generate_variant:Nn	451, 467, 713, 729, 2155, 2160, 2236, 2526, 3077, 3535, 4306	\end	1392, 1410, 2189, 2220, 3292, 3309, 3409, 3426, 3711, 3734, 3766, 3789, 4354, 4362, 4368
\cs_if_exist:NTF	421	\endgroup	2575
\cs_if_free:NTF	2485, 2497	\endlist	32
\cs_new:Nn	195	\endlist	345
\cs_new:Npn	213, 1545, 1554, 1563, 2118, 2127, 2135	\endlrbox	4005, 4242
\cs_new_eq:NN	344, 345, 346, 350, 351, 396, 397, 400, 401	\endminipage	32
\cs_new_protected:Nn	205, 219, 243, 276, 303, 309, 315, 321, 327, 335, 353, 368, 578, 641, 693, 804, 930, 934, 938, 942, 946, 950, 954, 958, 962, 966, 970, 974, 978, 982, 986, 990, 1025, 1037, 1061, 1078, 1089, 1106, 1181, 1205, 1222, 1284, 1301, 1323, 1358, 1364, 1439, 1453, 1467, 1478, 1489, 1500, 1511, 1522, 1607, 1710, 1723, 1740, 1761, 1789, 1794, 1819, 1860, 1870, 1908, 1923, 1930, 1939, 1944, 1949, 1954, 1963, 1968, 1973, 1995, 2161, 2185, 2192, 2216, 2223, 2264, 2361, 2472, 2527, 2548, 2579, 2603, 2645, 2669, 2698, 2713, 2741, 2774, 2810, 2822, 2830, 2881, 2885, 2904, 2955, 2990, 3006, 3016, 3032, 3170, 3198, 3227, 3234, 3257, 3287, 3299, 3338, 3362, 3380, 3405, 3416, 3453, 3497, 3511, 3531, 3536, 3552, 3571, 3688, 3707, 3743, 3762, 3820, 3843, 3850, 3859, 3869, 3886, 4026, 4063, 4085, 4091, 4104, 4160, 4264	\endminipage	351
\cs_new_protected:Npn	183, 187, 191, 404, 419, 436, 446, 452, 558, 603, 675, 700, 714, 1386, 1405, 1575, 1594, 1664, 1697, 1799, 1978, 2072, 2082, 2104, 2112, 2147, 2156, 2283, 2295, 2438, 2450, 2518, 2569, 2677, 2751, 2795, 2914, 2932, 2966, 2978, 3046, 3080, 3120, 3180, 3358, 3506, 3590, 3639, 3832, 3892, 3899, 3915, 3923, 3928, 3940, 4078, 4110, 4117, 4133, 4141, 4155, 4285, 4298, 4345, 4431, 4443	enumext	5, <u>3151</u>
\cs_new_protected_nopar:Nn	3879, 4002, 4097, 4239	enumext internal commands:	
\cs_new_protected_nopar:Npn	3946, 4188	\l__enumext__check_start_line_env_tl	31
\cs_set:Nn	2443	\l__enumext__ref_the_count_tl	37
\cs_set:Npn	2371, 2409, 4291	\l__enumext__resume_name_tl	58
\cs_set_eq:NN	3810, 3811, 3948, 4053, 4054, 4190	__enumext_add_pre_parsep:	47, 1035, <u>1037</u> , 1037
\cs_set_protected:Nn	854, 870, 882, 894	__enumext_after_args_exec:	44, <u>930</u> , 942, 3163
\cs_set_protected:Npn	44, 53, 70, 78, 90, 96, 125, 150, 158, 468, 490, 522, 538, 585, 730, 750, 794, 830, 849, 906, 915, 994, 1011, 1422, 1533, 1780, 1841, 2015, 2054, 2090, 2363, 2702, 2865, 3078, 3118	__enumext_after_args_exec_v:	45, <u>946</u> , 958, 3331
\cs_to_str:N	438, 461	__enumext_after_args_exec_vii:	962, 986
\cs_undefine:N	2474, 2475, 2476, 2477	__enumext_after_args_exec_viii:	990
		__enumext_after_env:nn	66, 77, 78, 93, 102, 108, <u>187</u> , 187, 2589, 3319, 3716, 3771, 4040
D		__enumext_after_hyperref:	33, 366, <u>368</u> , 368
\d	201	__enumext_after_list:	93, 104, 109, 3168, <u>3299</u> , 3299
\DeclareDocumentEnvironment	357	\l__enumext_after_list_args_v_tl	960
dim commands:		\l__enumext_after_list_args_vii_tl	988, 3996
\dim_abs:n	3051, 3056	\l__enumext_after_list_args_viii_tl	992, 4225
\dim_add:Nn	3516	__enumext_after_list_v:	96, 3336, <u>3416</u> , 3416
		__enumext_after_list_vii:	3818, <u>3850</u> , 3850
		__enumext_after_list_viii:	4061, <u>4091</u> , 4091
		__enumext_after_stop_list:	44, 45, <u>930</u> , 938, 3314
		__enumext_after_stop_list_v:	44, <u>946</u> , 954, 3431
		\l__enumext_after_stop_list_v_tl	956
		__enumext_after_stop_list_vii:	<u>962</u> , 978, 3853
		\l__enumext_after_stop_list_vii_tl	980
		__enumext_after_stop_list_viii:	982, 4094
		\l__enumext_after_stop_list_viii_tl	984
		\l__enumext_align_label_vii_str	3988, 3992
		\l__enumext_align_label_viii_str	4217, 4221
		\l__enumext_align_label_X_str	<u>158</u>
		\c__enumext_all_envs_clist	<u>178</u> , 489, 749, 848, 914, 929, <u>1010</u> , <u>1438</u>

```

\c__enumext_all_families_seq .. 116, 4393, 4402,
    4424
\__enumext_anskey__env_keys: ..... 78
\l__enumext_anskey_env_bool 30, 77, 34, 253, 268,
    2520
\__enumext_anskey_env_clean: .. 79, 2599, 2603,
    2669
\__enumext_anskey_env_define_keys: 77, 2518,
    2527, 2583
\__enumext_anskey_env_exec: 78, 2523, 2579, 2579
\__enumext_anskey_env_keys: .. 2597, 2603, 2603
\__enumext_anskey_env_make:n 62, 77, 1824, 2518,
    2518, 2526
\__enumext_anskey_env_store: .. 79, 2598, 2603,
    2645
\__enumext_anskey_env_undefine_keys: . 77, 78,
    2548, 2600
\__enumext_anskey_env_undefine_keys:\__-
    enumext_rescan_anskey_env:n ..... 2518
\l__enumext_anskey_level_int .. 28, 2289, 2290
\__enumext_anskey_safe_inner:n .. 71, 72, 2259,
    2264, 2283
\__enumext_anskey_safe_outer: . 71, 2254, 2264,
    2264
\__enumext_anskey_show_wrap_arg:n . 75, 2438,
    2438, 2454, 2469
\__enumext_anskey_show_wrap_left:n 75, 2302,
    2450, 2450
\__enumext_anskey_wrapper:n ..... 2019, 2448
\__enumext_at_begin_document:n .. 32, 183, 183,
    342, 348
\l__enumext_base_line_fix_bool . 798, 808, 819,
    828
\__enumext_before_args_exec: 44, 930, 930, 3237
\__enumext_before_args_exec_v: 44, 45, 946, 946,
    3365
\__enumext_before_args_exec_vii: .. 962, 962,
    3847
\__enumext_before_args_exec_viii: 966, 4088
\__enumext_before_env:nn 77, 187, 191, 2479, 2491,
    2503, 2581
\__enumext_before_keys_exec: 44, 930, 934, 3161
\__enumext_before_keys_exec_v: .. 44, 946, 950,
    3329
\__enumext_before_keys_exec_vii ..... 962
\__enumext_before_keys_exec_vii: 45, 970, 3806
\__enumext_before_keys_exec_viii: .. 45, 974,
    4049
\__enumext_before_list: ... 91, 3155, 3234, 3234
\__enumext_before_list_v: . 94, 3324, 3362, 3362
\__enumext_before_list_vii: .. 104, 3801, 3843,
    3843
\__enumext_before_list_viii: .. 109, 4045, 4085,
    4085
\l__enumext_before_no_starred_key_v_tl 952
\l__enumext_before_no_starred_key_vii_-
    tl ..... 972
\l__enumext_before_no_starred_key_viii_-
    tl ..... 976
\l__enumext_before_starred_key_v_tl ... 948
\l__enumext_before_starred_key_vii_tl . 964
\l__enumext_before_starred_key_viii_tl 968
\__enumext_calc_hspace:NNNNNNN 88, 3046, 3046,
    3077, 3082, 3122
\__enumext_check_ans_active: 63, 91, 1860, 1860,
    3238, 3846
\g__enumext_check_ans_item_tl ..... 81
\g__enumext_check_ans_key_bool 64, 65, 138, 317,
    1914, 1920, 2005
\l__enumext_check_ans_key_bool 64, 84, 85, 1845,
    1850, 1911, 1917
\__enumext_check_ans_key_hook: 64, 1908, 1908,
    3313, 3854
\__enumext_check_ans_level: 63, 1860, 1866, 1870
\__enumext_check_ans_log: .. 64-66, 1954, 1954,
    2009
\__enumext_check_ans_log_msg_greater: 1954,
    1960, 1973
\__enumext_check_ans_log_msg_less: 1954, 1958,
    1963
\__enumext_check_ans_log_msg_same_ok: 1954,
    1959, 1968
\__enumext_check_ans_msg_greater: 1930, 1936,
    1949
\__enumext_check_ans_msg_less: 1930, 1934, 1939
\__enumext_check_ans_msg_same_ok: 1930, 1935,
    1944
\__enumext_check_ans_show: .. 64-66, 1930, 1930,
    2007
\g__enumext_check_ans_show_bool ..... 93
\l__enumext_check_answers_bool 62, 63, 71, 138,
    1822, 1849, 1864, 2163, 2187, 2194, 2218, 2256, 2790,
    2918, 2948, 3959
\__enumext_check_starred_cmd:n 31, 66, 81, 1978,
    1978, 3334, 3492, 4059
\g__enumext_check_starred_cmd_int 138, 1981,
    1987, 1992, 2988, 3462, 4167
\l__enumext_check_start_line_env_tl 138, 282,
    289, 296, 1984, 1990, 1993
\l__enumext_columns_sep_v_dim 3384, 3386, 3394
\l__enumext_columns_sep_vii_dim .. 3554, 3556,
    3565, 3629, 4024
\l__enumext_columns_sep_viii_dim . 3573, 3575,
    3584, 3678, 4262
\l__enumext_columns_v_int 1227, 3382, 3390, 3402,
    3407
\l__enumext_columns_vii_int .. 3559, 3562, 3566,
    3593, 3597, 3600, 3606, 3612, 3616, 4019, 4030
\l__enumext_columns_viii_int . 3578, 3581, 3585,
    3642, 3646, 3649, 3655, 3661, 3665, 4257, 4270
\l__enumext_counter_i_tl ..... 44, 428
\l__enumext_counter_ii_tl ..... 44, 429
\l__enumext_counter_iii_tl ..... 44, 430
\l__enumext_counter_iv_tl ..... 44, 431
\c__enumext_counter_style_tl ..... 29, 49, 207
\g__enumext_counter_styles_tl . 26, 35, 66, 439,
    457
\l__enumext_counter_v_tl ..... 44, 432, 683
\l__enumext_counter_vi_tl ..... 44, 433
\l__enumext_counter_vii_tl ..... 44, 434, 613
\l__enumext_counter_viii_tl ..... 44, 435, 630
\l__enumext_current_widest_dim 26, 66, 463, 546,
    593, 664, 668
\__enumext_default_item:n ... 2914, 2914, 2963
\__enumext_define_counters:Nn 25, 419, 419, 428,
    429, 430, 431, 432, 433, 434, 435
\__enumext_endminipage: . 32, 348, 351, 363, 3548,
    4004, 4241
\g__enumext_envir_name_tl 30, 34, 254, 269, 325,

```

1792, 1797, 1807, 1942, 1947, 1952, 1966, 1971, 1976
 __enumext_execute_after_env: 31, 32, 61, 64–66, 76, 1995, 1995, 3319, 4040
 __enumext_fake_item: 854, 854, 3109
 \l__enumext_fake_item_indent_v_dim 873, 878
 \l__enumext_fake_item_indent_v_tl 875, 2971, 2975, 2983
 \l__enumext_fake_item_indent_vii_dim 885, 890
 \l__enumext_fake_item_indent_vii_tl 887, 4000
 \l__enumext_fake_item_indent_viii_dim . 897, 902, 4233
 \l__enumext_fake_item_indent_viii_tl .. 899, 4231, 4236
 \l__enumext_fake_item_indent_X_tl 96
 __enumext_fake_item_vii: 854, 882, 3139
 __enumext_fake_item_viii: 854, 894, 3144
 __enumext_filter_save_key:n .. 69, 2079, 2087, 2110, 2116, 2118, 2118, 4310, 4314, 4318, 4322, 4326, 4330
 __enumext_filter_save_key_key:n .. 69, 2118, 2123, 2127
 __enumext_filter_save_key_pair:nn 69, 2118, 2124, 2135
 __enumext_filter_series:n 57, 1545, 1545, 1587, 1599, 1604
 __enumext_filter_series_key:n 57, 1545, 1550, 1554
 __enumext_filter_series_pair:nn .. 57, 1545, 1551, 1563
 \g__enumext_footnote_arg_seq . 155, 2887, 2900, 2910
 \g__enumext_footnote_int . 155, 2894, 2897, 2899, 2901
 \g__enumext_footnote_int_seq . 155, 2888, 2901, 2906, 2909
 __enumext_footnotes_key_bool 33
 \l__enumext_footnotes_key_bool 28, 33, 107, 145, 377, 382, 391, 3967, 4015, 4205, 4252
 __enumext_footnotetext:nn ... 2881, 2881, 2911
 __enumext_getkeyans:nn .. 114, 4294, 4298, 4298
 __enumext_getkeyans_aux:n 113, 4282, 4285, 4285
 \l__enumext_hyperref_bool 28, 33, 145, 373, 394, 411, 2351, 2778, 3954, 4196
 __enumext_hypertarget:nn 33, 368, 396, 400, 416
 __enumext_if_is_int:n 199
 __enumext_if_is_int:nTF 199, 702, 716
 __enumext_internal_mini_page: .. 32, 353, 353, 3172, 3822
 __enumext_is_not_nested: . 25, 30, 90, 219, 219, 3173, 3823
 __enumext_is_on_first_level: . 25, 30, 90, 219, 243, 3178, 3830
 \g__enumext_item_answer_key_int 71, 82, 138, 312, 339, 340, 1927, 2258, 2792
 __enumext_item_answer_diff: 64–66, 1923, 1923, 2002
 \g__enumext_item_answer_diff_int . 64, 65, 144, 313, 1925, 1932, 1956
 \l__enumext_item_column_pos_vii_int 105, 3600, 3606, 3612, 3616, 3623, 3882, 4019, 4022
 \l__enumext_item_column_pos_viii_int .. 110, 3649, 3655, 3661, 3665, 3672, 4100, 4257, 4260
 \l__enumext_item_column_pos_X_int 158
 \g__enumext_item_count_all_vii_int 105, 3624, 3883, 4030, 4037
 \g__enumext_item_count_all_viii_int 110, 3673, 4101, 4269, 4277
 \g__enumext_item_count_all_X_int 158
 \g__enumext_item_number_int 63, 64, 138, 311, 338, 340, 1881, 1885, 1888, 1891, 1903, 1927, 2920, 2950, 3961
 __enumext_item_peek_args_vii: 105, 3884, 3886, 3886
 __enumext_item_peek_args_viii: .. 110, 4102, 4104, 4104
 __enumext_item_starred: .. 86, 3006, 3006, 3024
 \l__enumext_item_starred_vii_bool 3901, 3917, 3971
 \l__enumext_item_starred_viii_bool 4119, 4135, 4209, 4229
 \l__enumext_item_starred_X_bool 158
 __enumext_item_std:w 32, 84–86, 98, 342, 346, 2923, 2929, 2953, 2971, 2975, 2983, 3529
 \g__enumext_item_symbol_aux_vii_tl 3925, 3973, 3976, 3980, 3982
 \g__enumext_item_symbol_aux_X_tl 158
 \l__enumext_item_symbol_sep_vii_dim .. 3934, 3942, 3979, 3981
 \g__enumext_item_symbol_tl ... 85, 60, 122, 2938, 3012, 3029
 \l__enumext_item_symbol_vii_tl 3976
 \l__enumext_item_text_vii_box 3966, 4007, 4014
 \l__enumext_item_text_viii_box 4204, 4244, 4251
 \l__enumext_item_text_X_box 158
 \l__enumext_item_width_vii_dim ... 3563, 3627, 3635, 3636
 \l__enumext_item_width_viii_dim .. 3582, 3676, 3684, 3685
 \l__enumext_item_width_X_dim 158
 \l__enumext_itemindent_X_dim 70
 \l__enumext_itemsep_vii_skip 4036
 \l__enumext_itemsep_viii_skip 4276
 \l__enumext_joined_item_aux_vii_int .. 3621, 3622, 3623, 3624, 3630
 \l__enumext_joined_item_aux_viii_int . 3670, 3671, 3672, 3673, 3679
 \l__enumext_joined_item_aux_X_int 158
 __enumext_joined_item_vii:w .. 105, 3889, 3890, 3892, 3892
 \l__enumext_joined_item_vii_int .. 3592, 3593, 3596, 3598, 3604, 3609, 3614, 3619, 3621, 3627
 __enumext_joined_item_viii:w . 110, 4107, 4108, 4110, 4110
 \l__enumext_joined_item_viii_int . 3641, 3642, 3645, 3647, 3653, 3658, 3663, 3668, 3670, 3676
 \l__enumext_joined_item_X_int 158
 \l__enumext_joined_width_vii_dim . 3625, 3632, 3635, 3997, 4009
 \l__enumext_joined_width_viii_dim 3674, 3681, 3684, 4226, 4246
 \l__enumext_joined_width_X_dim 158
 __enumext_keyans_addto_prop:n 80, 2677, 2677, 2985, 3459
 __enumext_keyans_addto_seq:n . 81, 2751, 2751, 2987, 3461
 __enumext_keyans_addto_seq_link: 2751, 2772, 2774, 4166
 __enumext_keyans_anspic_code:nnn . 97, 3450,

[3453](#), [3453](#)
`__enumext_keyans_default_item:n` .. [85](#), [2966](#),
[2966](#), [3002](#)
`\l__enumext_keyans_env_bool` [34](#), [3203](#), [3216](#), [3345](#),
[3430](#)
`__enumext_keyans_fake_item:` .. [854](#), [870](#), [3099](#)
`\l__enumext_keyans_level_h_int` .. [28](#), [623](#), [650](#),
[2274](#), [2509](#), [2729](#), [4065](#), [4066](#)
`\l__enumext_keyans_level_int` .. [28](#), [1380](#), [2270](#),
[2505](#), [2724](#), [3344](#), [3349](#), [3444](#)
`__enumext_keyans_make_label:` [35](#), [87](#), [3032](#), [3032](#),
[3097](#)
`__enumext_keyans_mini_addvspace:` [51](#), [95](#), [1284](#),
[1284](#), [3374](#)
`__enumext_keyans_mini_right_cmd:n` [54](#), [1382](#),
[1405](#), [1405](#)
`__enumext_keyans_mini_set_vskip:` .. [50](#), [1222](#),
[1222](#), [1286](#)
`__enumext_keyans_multi_addvspace:` [95](#), [1078](#),
[1089](#), [3399](#)
`__enumext_keyans_multi_set_vskip:` [47](#), [1078](#),
[1078](#), [1091](#)
`__enumext_keyans_multicols_start:` [95](#), [3378](#),
[3380](#), [3380](#)
`__enumext_keyans_multicols_stop:` .. [95](#), [1409](#),
[3405](#), [3405](#), [3429](#)
`__enumext_keyans_parse_keys:n` [3323](#), [3358](#), [3358](#)
`\l__enumext_keyans_pic_above_int` .. [133](#), [3539](#),
[3540](#), [3542](#)
`\l__enumext_keyans_pic_above_skip` .. [98](#), [133](#),
[3483](#), [3523](#)
`__enumext_keyans_pic_arg_two:` [98](#), [3481](#), [3511](#),
[3511](#)
`\l__enumext_keyans_pic_below_int` .. [133](#), [3539](#),
[3540](#), [3543](#)
`\l__enumext_keyans_pic_body_seq` .. [97-99](#), [133](#),
[3448](#), [3488](#), [3547](#)
`__enumext_keyans_pic_do:n` [99](#), [3488](#), [3490](#), [3531](#),
[3531](#), [3535](#)
`\l__enumext_keyans_pic_level_int` .. [28](#), [1372](#),
[2278](#), [2513](#), [2680](#), [2719](#), [2754](#), [2832](#), [3499](#), [3500](#)
`__enumext_keyans_pic_row:n` [99](#), [3533](#), [3536](#), [3536](#)
`__enumext_keyans_pic_safe_exec:` .. [98](#), [3477](#),
[3497](#), [3497](#)
`__enumext_keyans_pic_skip_abs:N` .. [98](#), [3506](#),
[3506](#), [3522](#)
`\l__enumext_keyans_pic_width_dim` .. [133](#), [3538](#),
[3545](#)
`__enumext_keyans_redefine_item:` .. [86](#), [2990](#),
[2990](#), [3096](#)
`__enumext_keyans_ref:` .. [39](#), [675](#), [693](#), [3098](#)
`__enumext_keyans_ref:n` .. [39](#), [672](#), [675](#), [675](#)
`__enumext_keyans_safe_exec:` .. [3322](#), [3338](#), [3338](#)
`__enumext_keyans_show_ans:` .. [2795](#), [2803](#), [2822](#)
`__enumext_keyans_show_item_opt:` .. [2795](#), [2810](#),
[2983](#), [3473](#), [4232](#)
`__enumext_keyans_show_left:n` .. [86](#), [2795](#), [2795](#),
[2981](#), [3468](#)
`__enumext_keyans_show_pos:` .. [2795](#), [2807](#), [2830](#)
`__enumext_keyans_starred_item:n` .. [86](#), [2978](#),
[2978](#), [2998](#)
`__enumext_keyans_start_line:` .. [25](#), [31](#), [276](#), [276](#),
[3346](#), [3504](#), [4070](#)
`__enumext_keyans_store_ref:` .. [80](#), [2698](#), [2698](#),
[2986](#), [3460](#), [4164](#)
`__enumext_keyans_store_ref_aux_i:` [80](#), [2698](#),
[2710](#), [2713](#)
`__enumext_keyans_store_ref_aux_ii:` [81](#), [2698](#),
[2739](#), [2741](#)
`__enumext_keyans_wrapper_opt:n` .. [2022](#), [2818](#)
`\l__enumext_label_copy_i_tl` .. [2405](#), [2717](#), [2722](#),
[2727](#), [2732](#)
`\l__enumext_label_copy_v_tl` .. [2727](#)
`\l__enumext_label_copy_vi_tl` .. [2722](#)
`\l__enumext_label_copy_vii_tl` [2380](#), [2391](#), [2422](#),
[2717](#)
`\l__enumext_label_copy_viii_tl` .. [2732](#)
`\l__enumext_label_copy_X_tl` .. [147](#)
`\l__enumext_label_fill_left_v_tl` .. [3036](#)
`\l__enumext_label_fill_left_X_tl` .. [96](#)
`\l__enumext_label_fill_right_v_tl` .. [3043](#)
`\l__enumext_label_fill_right_X_tl` .. [96](#)
`\l__enumext_label_font_style_v_tl` [3037](#), [3472](#)
`\l__enumext_label_font_style_vii_tl` .. [3985](#)
`\l__enumext_label_font_style_viii_tl` .. [4214](#)
`\l__enumext_label_i_tl` .. [538](#)
`\l__enumext_label_ii_tl` .. [538](#)
`\l__enumext_label_iii_tl` .. [538](#)
`\l__enumext_label_iv_tl` .. [538](#)
`__enumext_label_style:Nnn` [25](#), [35](#), [452](#), [452](#), [467](#),
[543](#), [590](#), [661](#), [665](#)
`\l__enumext_label_v_tl` .. [80](#), [81](#), [658](#), [2685](#), [2759](#),
[2824](#), [2859](#), [2980](#), [2984](#), [3326](#), [3467](#), [3469](#)
`\l__enumext_label_vi_tl` .. [80](#), [81](#), [658](#), [2682](#), [2756](#),
[3467](#), [3469](#), [3473](#)
`\l__enumext_label_vii_tl` .. [585](#), [3912](#), [3937](#), [3944](#)
`\l__enumext_label_viii_tl` [585](#), [4130](#), [4158](#), [4162](#)
`\l__enumext_label_width_by_box` .. [66](#), [448](#), [449](#)
`__enumext_label_width_by_box:Nn` [35](#), [446](#), [446](#),
[451](#), [463](#), [726](#)
`\l__enumext_labelsep_i_dim` .. [2827](#), [2862](#), [4170](#),
[4185](#)
`\l__enumext_labelsep_v_dim` .. [3389](#)
`\l__enumext_labelsep_vii_dim` .. [3558](#), [3567](#), [3628](#),
[3935](#), [3995](#), [4011](#)
`\l__enumext_labelsep_viii_dim` [3577](#), [3586](#), [3677](#),
[4224](#), [4233](#), [4248](#)
`\l__enumext_labelwidth_i_dim` .. [2827](#), [2862](#), [4170](#),
[4185](#)
`\l__enumext_labelwidth_v_dim` .. [3389](#)
`\l__enumext_labelwidth_vii_dim` .. [3558](#), [3566](#),
[3628](#), [3988](#), [3992](#), [4010](#)
`\l__enumext_labelwidth_viii_dim` .. [3577](#), [3585](#),
[3677](#), [4217](#), [4221](#), [4247](#)
`\l__enumext_leftmargin_tmp_v_bool` .. [98](#), [3513](#)
`\l__enumext_leftmargin_tmp_X_bool` .. [70](#)
`\l__enumext_leftmargin_tmp_X_dim` .. [70](#)
`\l__enumext_leftmargin_X_dim` .. [70](#)
`__enumext_level:` [195](#), [195](#), [567](#), [570](#), [571](#), [580](#), [582](#),
[857](#), [861](#), [865](#), [932](#), [936](#), [940](#), [944](#), [1027](#), [1029](#), [1031](#),
[1033](#), [1066](#), [1068](#), [1070](#), [1072](#), [1076](#), [1109](#), [1112](#), [1131](#),
[1140](#), [1146](#), [1151](#), [1155](#), [1166](#), [1170](#), [1171](#), [1176](#), [1212](#),
[1216](#), [1389](#), [1395](#), [1442](#), [1444](#), [1446](#), [1449](#), [1456](#), [1458](#),
[1460](#), [1463](#), [2074](#), [2076](#), [2078](#), [2106](#), [2107](#), [2109](#), [2165](#),
[2173](#), [2177](#), [2181](#), [2443](#), [2446](#), [2447](#), [2922](#), [2923](#), [2927](#),
[2928](#), [2929](#), [2936](#), [2938](#), [2942](#), [2943](#), [2946](#), [2952](#), [2953](#),
[3008](#), [3011](#), [3013](#), [3020](#), [3021](#), [3022](#), [3025](#), [3028](#), [3158](#),
[3160](#), [3209](#), [3222](#), [3229](#), [3240](#), [3242](#), [3245](#), [3246](#), [3248](#),

3253, 3260, 3263, 3265, 3267, 3268, 3269, 3270, 3273,
 3279, 3284, 3290, 3293, 3295, 3301
 \l__enumext_level_h_int .. 28, 227, 249, 263, 606,
 643, 1878, 1894, 2399, 2416, 2483, 2495, 3824, 3825
 \l__enumext_level_int . 90, 28, 197, 236, 248, 264,
 355, 1039, 1183, 1376, 1872, 1900, 1997, 2376, 2386,
 2392, 2398, 2406, 2414, 2421, 2482, 2494, 3112, 3174,
 3175, 3185, 3193, 3207, 3220, 3275, 3353, 3440, 3863,
 3873, 4073
 __enumext_list_arg_two_i: 3078
 __enumext_list_arg_two_ii: 3078
 __enumext_list_arg_two_iii: 3078
 __enumext_list_arg_two_iv: 3078
 __enumext_list_arg_two_v: . 86, 3078, 3328, 3514
 __enumext_list_arg_two_vii: 3118, 3805
 __enumext_list_arg_two_viii: 3118, 4048
 \l__enumext_listoffset_v_dim 3391
 \l__enumext_listparindent_vii_dim 3998
 \l__enumext_listparindent_viii_dim ... 4227
 __enumext_log_answer_vars: . 32, 327, 335, 2004
 __enumext_log_global_vars: . 31, 327, 327, 2003
 __enumext_make_label: 35, 84, 85, 87, 3016, 3016,
 3107
 \l__enumext_mark_answer_sym_tl 71, 2028, 2231,
 2458, 2834, 2847, 4174
 \l__enumext_mark_position_str 122, 2032, 2033,
 2059, 2060, 2229
 \l__enumext_mark_ref_sym_tl .. 2045, 2356, 2786
 __enumext_mini_addvspace: .. 50, 92, 1205, 1205,
 3250
 __enumext_mini_addvspace_vii: 53, 1358, 1358,
 3702
 __enumext_mini_addvspace_viii: 53, 1358, 1364,
 3757
 __enumext_mini_env* 353
 __enumext_mini_right_cmd:n . 53, 54, 1384, 1386,
 1386
 __enumext_mini_set_vskip: . 48, 1106, 1106, 1207
 __enumext_mini_set_vskip_vii: 52, 1301, 1301,
 1360
 __enumext_mini_set_vskip_viii: 52, 1301, 1323,
 1366
 __enumext_minipage:w 32, 348, 350, 359, 3545, 3997,
 4226
 \l__enumext_minipage_active_v_bool .. 95, 96,
 3372, 3397, 3410, 3418
 \g__enumext_minipage_active_vii_bool .. 102,
 3713, 3718, 3737
 \l__enumext_minipage_active_vii_bool . 3698,
 3709
 \g__enumext_minipage_active_viii_bool 3768,
 3773, 3792
 \l__enumext_minipage_active_viii_bool 3753,
 3764
 \g__enumext_minipage_active_X_bool ... 158
 \l__enumext_minipage_active_X_bool 84
 \g__enumext_minipage_after_skip 84, 1305, 1317,
 3735, 3790
 \l__enumext_minipage_after_skip 48, 50, 93, 96,
 84, 1122, 1137, 1157, 1173, 1188, 1194, 1200, 1214,
 1224, 1233, 1236, 1248, 1266, 1277, 1293, 1325, 1338,
 1352, 3310, 3427
 \g__enumext_minipage_center_vii_bool . 3722,
 3738
 \g__enumext_minipage_center_viii_bool 3777,
 3793
 \g__enumext_minipage_center_X_bool ... 158
 \l__enumext_minipage_hsep_v_dim ... 94, 3370
 \l__enumext_minipage_hsep_vii_dim 3696
 \l__enumext_minipage_hsep_viii_dim ... 3751
 \l__enumext_minipage_left_skip 48, 95, 84, 1114,
 1129, 1148, 1163, 1210, 1220, 1225, 1231, 1240, 1257,
 1269, 1289, 1299, 1303, 1308, 1312, 1326, 1330, 1344,
 1362, 1368
 \l__enumext_minipage_left_v_dim 94, 3368, 3376
 \l__enumext_minipage_left_vii_dim 3692, 3704
 \l__enumext_minipage_left_viii_dim 3747, 3759
 \l__enumext_minipage_left_X_dim 84
 \g__enumext_minipage_right_skip 84, 1304, 1309,
 1313, 3721, 3776
 \l__enumext_minipage_right_skip . 48, 84, 1118,
 1133, 1153, 1168, 1226, 1232, 1244, 1262, 1273, 1327,
 1334, 1348, 1396, 1413
 \l__enumext_minipage_right_v_dim .. 94, 1407,
 1412, 3366, 3370
 \g__enumext_minipage_right_vii_dim 101, 3700,
 3720, 3740
 \l__enumext_minipage_right_vii_dim 101, 3690,
 3695, 3701
 \g__enumext_minipage_right_viii_dim .. 3755,
 3775, 3795
 \l__enumext_minipage_right_viii_dim .. 3745,
 3750, 3756
 \g__enumext_minipage_right_X_dim 158
 \g__enumext_minipage_right_X_skip 158
 \g__enumext_minipage_stat_int . 92, 95, 84, 1401,
 1418, 3249, 3303, 3308, 3373, 3420, 3425
 \l__enumext_miniright_code_vii_box 3729, 3733
 \g__enumext_miniright_code_vii_tl 102, 3724,
 3731, 3739
 \l__enumext_miniright_code_viii_box .. 3784,
 3788
 \g__enumext_miniright_code_viii_tl 3779, 3786,
 3794
 \l__enumext_miniright_code_X_box 158
 __enumext_multi_addvspace: . 47, 92, 1061, 1061,
 3281
 __enumext_multi_set_vskip: 46, 1025, 1025, 1063
 \l__enumext_multicols_above_ii_skip ... 1044
 \l__enumext_multicols_above_iii_skip .. 1050
 \l__enumext_multicols_above_iv_skip ... 1056
 \l__enumext_multicols_above_v_skip 1080, 1094,
 1104
 \l__enumext_multicols_above_X_skip 78
 \l__enumext_multicols_below_v_skip 1084, 1098,
 3412
 \l__enumext_multicols_below_X_skip 78
 __enumext_multicols_start: 92, 3255, 3257, 3257
 __enumext_multicols_stop: 93, 1391, 3287, 3287,
 3312
 __enumext_nested_base_line_fix: 41, 794, 804,
 3189, 3840
 __enumext_newlabel:nn 28, 33, 75, 404, 404, 2432,
 2745
 \l__enumext_newlabel_arg_one_tl 28, 33, 75, 80,
 147, 2355, 2425, 2433, 2734, 2746, 2784
 \l__enumext_newlabel_arg_two_tl 28, 34, 74, 147,
 2379, 2389, 2403, 2419, 2434, 2721, 2726, 2731, 2747


```

__enumext_parse_keys:n 41, 57, 3154, 3180, 3180
__enumext_parse_keys_vii:n . 41, 57, 3800, 3832,
3832
__enumext_parse_keys_viii:n . 4044, 4078, 4078
__enumext_parse_save_key:n 68, 2099, 2104, 2104
__enumext_parse_save_key_vii:n 68, 2094, 2104,
2112
__enumext_parse_serie:n . . . . . 104
__enumext_parse_series:n . . 57, 90, 1575, 1575,
3188, 3838
__enumext_parse_store_keys:n . . . . . 90
\l__enumext_parsep_i_skip 1042, 1044, 1186, 1234
\l__enumext_parsep_ii_skip . . 1048, 1050, 1192
\l__enumext_parsep_iii_skip . . 1054, 1056, 1198
\l__enumext_parsep_vii_skip . . . . . 3999
\l__enumext_parsep_viii_skip . . . . . 4228
\l__enumext_partopsep_v_skip . 1096, 1100, 1260,
1264, 1271, 1275, 1291, 1295
\l__enumext_partopsep_viii_skip . . . . . 1336
__enumext_phantomsection: 33, 368, 397, 401, 417
__enumext_print_footnote: . . 2881, 2904, 4017,
4254
__enumext_print_keyans_box:NN 71, 2223, 2223,
2236, 2445, 2826, 2861, 4170, 4185
\l__enumext_print_keyans_i_tl . . . 4315, 4337
\l__enumext_print_keyans_ii_tl . . 4319, 4338
\l__enumext_print_keyans_iii_tl . . 4323, 4339
\l__enumext_print_keyans_iv_tl . . 4327, 4340
\l__enumext_print_keyans_starred_tl 114, 115,
122, 4311, 4359
\l__enumext_print_keyans_vii_tl 114, 4331, 4341
\l__enumext_print_keyans_X_tl . . . . . 122
__enumext_printkeyans:nnn 114, 115, 4342, 4345,
4345
__enumext_redefine_item: . 85, 2955, 2955, 3106
\l__enumext_ref_key_arg_tl 37, 49, 210, 560, 561,
574, 605, 608, 619, 625, 636, 677, 678, 689
\l__enumext_ref_the_count_tl . 37, 49, 567, 570,
573, 613, 615, 618, 630, 632, 635, 683, 685, 688
__enumext_regex_counter_style: . . 29, 37, 205,
205, 568, 614, 631, 684
__enumext_register_counter_style:Nn . . 436,
436, 441, 442, 443, 444, 445
__enumext_remove_extra_parsep_vii: . . 3815,
4026, 4026
__enumext_remove_extra_parsep_viii: . 4058,
4264, 4264
__enumext_renew_footnote: . . 2881, 2885, 3969,
4207
\l__enumext_renew_the_count_v_tl 686, 695, 697
\l__enumext_renew_the_count_vii_tl 616, 645,
647
\l__enumext_renew_the_count_viii_tl 633, 652,
654
\l__enumext_renew_the_count_X_tl . . . . . 49
__enumext_rescan_anskey_env:n . . 77, 79, 2569,
2655, 2663
__enumext_reset_global_bool: . . 303, 306, 315
__enumext_reset_global_int: . . 303, 305, 309
__enumext_reset_global_tl: . . . 303, 307, 321
__enumext_reset_global_vars: . 31, 66, 303, 303,
2012
\l__enumext_resume_active_bool 57, 59, 60, 1579,
1699
__enumext_resume_counter: . . 59, 60, 1697, 1703,
1710
__enumext_resume_counter:n . 57, 59, 1668, 1673,
1697, 1697, 1767, 1775
__enumext_resume_counter_save_ans: 60, 1697,
1708, 1740
__enumext_resume_counter_series: . 60, 1697,
1706, 1723
\g__enumext_resume_int . . . 60, 1620, 1714, 1715
__enumext_resume_last:n 57, 58, 1575, 1581, 1594
\l__enumext_resume_name_tl 60, 1616, 1624, 1627,
1643, 1651, 1654, 1700, 1701, 1729, 1736
__enumext_resume_save_counter: 58, 1607, 1607,
3317, 3857
__enumext_resume_series:n . 59, 1539, 1664, 1664
__enumext_resume_starred: . 61, 1540, 1761, 1761
\g__enumext_resume_vii_int . 104, 60, 1647, 1719,
1720
__enumext_safe_exec: . . 32, 90, 3153, 3170, 3170
__enumext_safe_exec_vii: . 32, 3799, 3820, 3820
__enumext_safe_exec_viii: . . 4043, 4063, 4063
\l__enumext_series_name_tl . . . . . 59
\l__enumext_series_str . 58, 90, 1537, 1577, 1585,
1586, 1588, 1590, 1611, 1614, 1618, 1638, 1641, 1645,
3184, 3836
__enumext_set_error:nn . . . . 4431, 4441, 4443
__enumext_set_parse:n . . . . . 4414, 4431, 4431
\l__enumext_setkey_tmpa_int . . 117, 4407, 4411
\l__enumext_setkey_tmpa_seq . . 117, 4405, 4415,
4421, 4423, 4425, 4438
\l__enumext_setkey_tmpa_tl . . . 117, 4413, 4417
\l__enumext_setkey_tmpb_seq . . 117, 4406, 4409,
4413, 4414
\l__enumext_setkey_tmpb_tl 117, 4433, 4435, 4436
\l__enumext_show_answer_bool . 2039, 2063, 2452,
2801, 2815, 3464, 4168
__enumext_show_length:nnn . . 43, 213, 213, 4532,
4533, 4534, 4535, 4536, 4537, 4538, 4539, 4540, 4541,
4547, 4548, 4549, 4550, 4551, 4552, 4553, 4554, 4555,
4556
\l__enumext_show_position_bool . . 2042, 2066,
2456, 2805, 2816, 3465, 4172
\g__enumext_standar_bool 30, 90, 34, 226, 229, 247,
318, 1609, 1674, 1686, 1712, 1725, 1763, 1899, 1912
\l__enumext_standar_bool . 90, 93, 34, 2384, 2397,
2413, 3177, 3316
\l__enumext_standar_first_bool 30, 90, 34, 252,
807, 1596, 1743, 1805, 1812
__enumext_standar_item_vii:w . 105, 106, 3897,
3899, 3899
__enumext_standar_item_viii:w 110, 4115, 4117,
4117
__enumext_standar_ref: . . . 37, 558, 578, 3108
__enumext_standar_ref:n . . . 37, 550, 558, 558
\g__enumext_standar_series_tl . 60, 1598, 1599,
1765, 1768
\g__enumext_starred_bool 30, 104, 34, 235, 238, 262,
319, 1636, 1679, 1690, 1717, 1732, 1771, 1877, 1918,
2375, 2385, 2415, 2715, 3204, 3217, 3741
\l__enumext_starred_bool . . 104, 34, 2310, 2316,
2400, 2441, 2607, 2612, 3829, 3856
__enumext_starred_columns_set_vii: . . 3552,
3552, 3808
__enumext_starred_columns_set_viii: . 3552,

```

```

    3571, 4051
\l__enumext_starred_first_bool 30, 34, 267, 818,
    1601, 1752, 1805, 1812
\__enumext_starred_item:nn ... 2932, 2932, 2961
\__enumext_starred_item_exec: . 111, 4160, 4160,
    4211
\__enumext_starred_item_vii:w . 105, 106, 3896,
    3915, 3915
\__enumext_starred_item_vii_aux_i:w .. 3915,
    3920, 3923
\__enumext_starred_item_vii_aux_ii:w . 3915,
    3921, 3926, 3928
\__enumext_starred_item_vii_aux_iii:w 3915,
    3931, 3940
\__enumext_starred_item_viii:w 110, 111, 4114,
    4133, 4133
\__enumext_starred_item_viii_aux_i:w .. 111,
    4133, 4138, 4141
\__enumext_starred_item_viii_aux_ii:w . 111,
    4133, 4139, 4153, 4155
\__enumext_starred_joined_item_vii:n 100, 105,
    3590, 3590, 3894
\__enumext_starred_joined_item_viii:n . 100,
    110, 3590, 3639, 4112
\__enumext_starred_ref: .... 38, 603, 641, 3136
\__enumext_starred_ref:n .... 38, 597, 603, 603
\g__enumext_starred_series_tl . 60, 1603, 1604,
    1773, 1776
\__enumext_start_from:NNn 39, 700, 700, 713, 735
\l__enumext_start_i_int ..... 1715, 1727, 1746
\__enumext_start_item_tmp_vii: 103, 3811, 3879,
    3879
\__enumext_start_item_tmp_viii: .. 108, 4054,
    4097, 4097
\__enumext_start_item_vii:w ... 106, 3907, 3912,
    3937, 3944, 3946, 3946
\__enumext_start_item_viii:w .. 110, 4125, 4130,
    4158, 4188, 4188
\g__enumext_start_line_tl 30, 34, 255, 270, 324,
    1942, 1947, 1952, 1966, 1971, 1976
\__enumext_start_list:nn 32, 87, 98, 342, 344, 3157,
    3325, 3478, 3803, 4046
\__enumext_start_mini_vii: 104, 3688, 3688, 3848
\__enumext_start_mini_viii: ... 109, 3743, 3743,
    4089
\__enumext_start_save_ans_msg: .. 61, 62, 1789,
    1789, 1814
\__enumext_start_store_level: . 91, 3156, 3198,
    3198
\__enumext_start_store_level_vii: 105, 3802,
    3859, 3859
\l__enumext_start_vii_int ... 1720, 1734, 1755
\l__enumext_start_X_int ..... 96, 730
\__enumext_stop_item_tmp_vii: .. 103, 105, 106,
    3810, 3814, 3881, 3948
\__enumext_stop_item_tmp_viii: 108, 110, 4053,
    4057, 4099, 4190
\__enumext_stop_item_vii: 106, 107, 3948, 4002,
    4002
\__enumext_stop_item_viii: 112, 4190, 4239, 4239
\__enumext_stop_list: .. 32, 342, 345, 3166, 3335,
    3491, 3816, 4060
\__enumext_stop_mini_vii: 102, 104, 3688, 3707,
    3852
\__enumext_stop_mini_viii: 109, 3743, 3762, 4093
\__enumext_stop_save_ans_msg: . 61, 1789, 1794,
    2001
\__enumext_stop_store_level: .. 91, 3167, 3198,
    3227
\__enumext_stop_store_level_vii: . 105, 3817,
    3859, 3869
\l__enumext_store_active_bool . 27, 62, 90, 104,
    108, 1744, 1753, 1821, 2266, 3202, 3215, 3340, 3348,
    3436, 3495, 3861, 3871, 4072
\__enumext_store_active_keys:n .. 67, 68, 2072,
    2072, 3195
\__enumext_store_active_keys_vii:n 67, 68, 104,
    2072, 2082, 3839
\__enumext_store_addto_prop:n 69, 80, 2147, 2147,
    2155, 2297, 2696, 4163
\__enumext_store_addto_seq:n 69, 81, 2156, 2156,
    2160, 2167, 2181, 2189, 2198, 2212, 2220, 2359, 2789
\l__enumext_store_anskey_arg_tl 27, 72, 73, 108,
    2307, 2312, 2314, 2319, 2326, 2329, 2339, 2344, 2347,
    2353, 2359
\__enumext_store_anskey_code:nn . 71, 72, 2260,
    2295, 2295
\l__enumext_store_anskey_env_tl 27, 78, 79, 108,
    2591, 2593, 2648, 2655, 2663
\l__enumext_store_anskey_opt_tl 27, 78, 79, 108,
    2592, 2609, 2615, 2622, 2628, 2638, 2650, 2659
\__enumext_store_anskey_safe_outer: .... 72
\g__enumext_store_columns_break_bool . 2531,
    2606, 2671
\l__enumext_store_columns_break_bool . 2239,
    2309
\l__enumext_store_current_label_tl 27, 80, 81,
    111, 108, 2679, 2682, 2685, 2692, 2694, 2696, 2753,
    2756, 2759, 2765, 2770, 2780, 2789, 4143, 4148, 4149,
    4162, 4163, 4165
\l__enumext_store_current_label_tmp_tl . 27,
    108, 2980, 2984
\l__enumext_store_current_opt_arg_tl 27, 111,
    108, 2799, 2812, 2818, 4151
\__enumext_store_internal_ref: .. 72, 73, 2300,
    2361, 2361
\g__enumext_store_item_join_int .. 2534, 2613,
    2617, 2672
\l__enumext_store_item_join_int .. 2242, 2317,
    2321
\g__enumext_store_item_star_bool . 2536, 2620,
    2673
\l__enumext_store_item_star_bool . 2244, 2324
\g__enumext_store_item_symbol_sep_dim 2541,
    2635, 2640, 2675
\l__enumext_store_item_symbol_sep_dim 2249,
    2336, 2341
\g__enumext_store_item_symbol_tl . 2539, 2626,
    2630, 2674
\l__enumext_store_item_symbol_tl . 2247, 2327,
    2331
\l__enumext_store_keyans_item_opt_sep_
    tl .... 2025, 2690, 2692, 2763, 2767, 4146, 4148
\__enumext_store_level_close: . 70, 2161, 2185,
    3231
\__enumext_store_level_close_vii: 2192, 2216,
    3875
\__enumext_store_level_open: .. 70, 2161, 2161,
    3210, 3223

```

`__enumext_store_level_open_vii:` . 2192, 2192, 3865
`\g__enumext_store_name_tl` . 27, 62, 93, 108, 323, 330, 331, 332, 333, 1797, 1823, 1941, 1946, 1951, 1965, 1970, 1975, 1999
`\l__enumext_store_name_tl` 27, 62, 63, 108, 1630, 1633, 1657, 1660, 1748, 1757, 1792, 1801, 1802, 1823, 1824, 1825, 1827, 1828, 1830, 1832, 1833, 1835, 1837, 1838, 1862, 2149, 2151, 2158, 2427, 2428, 2464, 2595, 2736, 2737, 2840, 2853, 4180
`\l__enumext_store_ref_key_bool` 72, 2048, 2298, 2350, 2700, 2777
`\l__enumext_store_save_key_vii_bool` . 2084, 2114
`\l__enumext_store_save_key_vii_tl` 2086, 2087, 2115, 2116, 2196, 2204, 2208, 2212
`\l__enumext_store_save_key_X_bool` . 67, 122
`\l__enumext_store_save_key_X_tl` . 67, 68, 122
`\l__enumext_store_upper_level_X_bool` . 122
`__enumext_storing_exec:` 62, 77, 1799, 1815, 1819
`__enumext_storing_set:n` 61, 62, 1784, 1799, 1799
`\l__enumext_the_counter_v_tl` 685
`\l__enumext_the_counter_vii_tl` 615
`\l__enumext_the_counter_viii_tl` 632
`\l__enumext_the_counter_X_tl` 49
`__enumext_tmp:n` 44, 48, 53, 59, 70, 77, 78, 83, 90, 95, 96, 107, 125, 132, 150, 154, 158, 177, 794, 803, 849, 853, 1533, 1544, 1780, 1788, 1841, 1859, 2015, 2053, 2054, 2071, 2090, 2103, 2363, 2370, 2371, 2392, 2406, 2409, 2421, 2702, 2709, 3078, 3117, 3118, 3150
`__enumext_tmp:nn` 468, 489, 490, 521, 522, 537, 730, 749, 830, 848, 906, 914, 915, 929, 994, 1010, 1011, 1024, 1422, 1438, 2865, 2880
`__enumext_tmp:nnn` 538, 554, 555, 556, 557, 585, 601, 602
`__enumext_tmp:nnnnn` 750, 775, 778, 781, 783, 785, 788, 791
`__enumext_tmp:w` 4291, 4294
`\l__enumext_tmpa_vii_int` 3562, 3565
`\l__enumext_tmpa_viii_int` 3581, 3584
`\l__enumext_tmpa_X_int` 158
`\l__enumext_topsep_v_skip` 1082, 1086, 1229, 1242, 1250, 1255, 1275, 1279, 3494, 3526
`\l__enumext_topsep_vii_skip` . 1306, 1315, 1319
`\l__enumext_topsep_viii_skip` . 1328, 1350, 1354
`__enumext_undefine_anskey_env:` . 66, 76, 2010, 2472, 2472
`\l__enumext_vspace_a_star_v_bool` 1471
`\l__enumext_vspace_a_star_vii_bool` . . 1493
`\l__enumext_vspace_a_star_viii_bool` . . 1504
`\l__enumext_vspace_a_star_X_bool` 96
`__enumext_vspace_above:` . 54, 1439, 1439, 3236
`__enumext_vspace_above_v:` . 55, 1467, 1467, 3364
`\l__enumext_vspace_above_v_skip` . 1469, 1473, 1475
`__enumext_vspace_above_vii:` . 55, 1489, 1489, 3845
`\l__enumext_vspace_above_vii_skip` 1491, 1495, 1497
`__enumext_vspace_above_viii:` . 55, 1489, 1500, 4087
`\l__enumext_vspace_above_viii_skip` 1502, 1506, 1508
`\l__enumext_vspace_b_star_v_bool` 1482
`\l__enumext_vspace_b_star_vii_bool` . . 1515
`\l__enumext_vspace_b_star_viii_bool` . . 1526
`\l__enumext_vspace_b_star_X_bool` 96
`__enumext_vspace_below:` . 55, 1453, 1453, 3315
`__enumext_vspace_below_v:` . 55, 1478, 1478, 3432
`\l__enumext_vspace_below_v_skip` . 1480, 1484, 1486
`__enumext_vspace_below_vii:` . 56, 1511, 1511, 3855
`\l__enumext_vspace_below_vii_skip` 1513, 1517, 1519
`__enumext_vspace_below_viii:` . 56, 1511, 1522, 4095
`\l__enumext_vspace_below_viii_skip` 1524, 1528, 1530
`__enumext_widest_from:nnn` . 40, 714, 714, 729, 741
`\g__enumext_widest_label_tl` 26, 35, 66, 456, 460, 464
`\l__enumext_wrap_label_opt_v_bool` 2974
`\l__enumext_wrap_label_opt_vii_bool` 106, 3906
`\l__enumext_wrap_label_opt_viii_bool` . 110, 4124
`\l__enumext_wrap_label_opt_X_bool` 96
`\l__enumext_wrap_label_v_bool` 2970, 2974, 2982, 3038
`\l__enumext_wrap_label_vii_bool` . 106, 3905, 3910, 3918, 3986
`\l__enumext_wrap_label_viii_bool` . 110, 4123, 4128, 4136, 4215
`\l__enumext_wrap_label_X_bool` 96
`__enumext_wrapper_label_v:n` 3040, 3473
`__enumext_wrapper_label_vii:n` 3989
`__enumext_wrapper_label_viii:n` 4218
`\l__enumext_write_aux_file_tl` . 28, 75, 81, 147, 2430, 2436, 2743, 2749
`__enumext_zero_parsep:` . . 50, 1126, 1181, 1181
enumext* 5, 3797
enumXi 428
enumXii 428
enumXiii 428
enumXiv 428
enumXv 428
enumXvi 428
enumXvii 428
enumXviii 428

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, 66–74, 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, 66–72, 74, 76, 78, 80, 83–88, 90, 91, 93, 94, 98, 99, 101, 105, 114, 115, 118, 119
keyans* 24, 25, 27–29, 31, 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, 31, 34, 35, 39, 53, 62, 63, 66, 69, 76, 80–82, 96–98, 120
keyans 24, 25, 27, 29, 31, 34, 35, 39, 40, 42–45, 47, 50, 51, 53–55, 62, 63, 66, 67, 69, 76, 80–82, 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	4294
\exp_args:Ne	2652, 2660, 3192, 4282
\exp_not:N .	57, 459, 573, 618, 635, 688, 863, 877, 878, 889, 890, 901, 902, 2355, 2461, 2462, 2782, 2837, 2838, 2850, 2851, 4177, 4178, 4291
\exp_not:n	257, 272, 284, 291, 298, 512, 532, 573, 574, 618, 619, 635, 636, 688, 689, 864, 1561, 1573, 2036, 2133, 2145, 2321, 2331, 2341, 2355, 2356, 2433, 2617, 2630, 2640, 2746, 2784, 2786
F	
\fbox	2020
file commands:	
\file_input_stop:	4703
first	915
font	468
\footnote	83
\footnote	83, 2889
\footnotemark	2899
\footnotesize	2462, 2838, 2851, 4178
\footnotetext	2883
G	
\getkeyans	15, 113, 4280
group commands:	
\group_begin: . .	2255, 2460, 2571, 2647, 2836, 2849, 3965, 3984, 4176, 4203, 4213, 4302, 4336
\group_end: .	2262, 2467, 2667, 2843, 2856, 3994, 4006, 4183, 4223, 4243, 4304, 4343
H	
\hbadness	4013, 4250
hbox commands:	
\hbox_set:Nn	448
\hfill	498, 502, 507, 508, 1393, 1411, 2355, 2782, 3712, 3767
hook commands:	
\hook_gput_code:nnn	9, 185, 189, 193, 366
\hook_gremove_code:nn	78, 2587
\hook_gset_rule:nnnn	367
\hook_if_empty:nTF	2585
\hspace	4024, 4262
\hyperlink	73, 81
\hyperlink	2355, 2782
\hypertarget	33
\hypertarget	396
I	
\IfHyperBoolean	374
\IfPackageLoadedTF	11, 19, 370, 384
\ignorespaces	866
\inputlineno	257, 272, 284, 291, 298
int commands:	
\int_add:Nn	3623, 3672
\int_case:nn . . .	1039, 1183, 1872, 1894, 1932, 1956
\int_compare:nNnTF . .	355, 606, 623, 643, 650, 1108, 1227, 1372, 1376, 1380, 1980, 1986, 1997, 2270, 2274, 2278, 2290, 2505, 2509, 2513, 2680, 2719, 2724, 2729, 2754, 2832, 3175, 3185, 3207, 3220, 3259, 3275, 3289, 3303, 3349, 3353, 3382, 3407, 3420, 3440, 3444, 3500, 3593, 3603, 3619, 3642, 3652, 3668, 3825, 3863, 3873, 4019, 4028, 4066, 4073, 4256, 4266, 4411
\int_compare_p:nNn . . .	227, 236, 248, 249, 263, 264, 1878, 1900, 2317, 2376, 2386, 2398, 2399, 2414, 2416, 2482, 2483, 2494, 2495, 2613

\int_decr:N	3622, 3671
\int_eval:n .	340, 2151, 2428, 2462, 2737, 2838, 2851, 3093, 3135, 3611, 3660, 4178
\int_from_alph:n	708, 722
\int_from_roman:n	710, 724
\int_gadd:Nn	3624, 3673
\int_gdecr:N	1881, 1885, 1888, 1891, 1903
\int_gincr:N	1714, 1719, 2258, 2792, 2920, 2950, 2988, 3249, 3373, 3462, 3883, 3961, 4101, 4167
\int_gset:Nn	1925, 2897
\int_gset_eq:NN	1613, 1620, 1626, 1632, 1640, 1647, 1653, 1659, 2894
\int_gzero:N .	311, 312, 313, 1401, 1418, 1992, 2672, 3308, 3425, 4037, 4277
\int_if_exist:NNTF	1588, 1624, 1630, 1651, 1657, 1835
\int_incr:N	2289, 3174, 3344, 3499, 3824, 3882, 4065, 4100
\int_mod:nn	4030, 4268
\int_new:N .	28, 29, 30, 31, 32, 33, 60, 61, 84, 100, 119, 135, 136, 141, 142, 143, 144, 155, 161, 162, 163, 164, 165, 1590, 1838
\int_set:Nn	704, 708, 710, 1727, 1734, 1746, 1755, 2572, 3539, 3540, 3562, 3581, 3592, 3598, 3614, 3641, 3647, 3663, 4013, 4250, 4407
\int_set_eq:NN	1715, 1720, 3621, 3670
\int_sign:n	1927
\int_step_function:nnN	2392, 2406, 2421
\int_step_inline:nnn	3541
\int_to_roman:n	197, 2372, 2410
\int_use:N	333, 338, 339, 1109, 1729, 1736, 1748, 1757, 3093, 3112, 3135, 3193, 3260, 3269, 3284, 3290, 3596, 3597, 3609, 3645, 3646, 3658
\int_zero:N	4022, 4260
\item	32, 44, 45, 70, 84, 96, 98, 99, 103, 108
\item .	84, 85, 105, 106, 110, 112, 346, 2169, 2175, 2200, 2206, 2314, 2756, 2759, 2957, 2992, 3809, 3811, 4052, 4054, 4165
\item*	5, 14, 66, 2990
item-pos*	2865
item-sym*	2865
\itemindent	88
\itemindent	87
itemindent	830
\itemsep	97, 98
\itemsep	3515, 3521
\itemwidth	3569, 3588, 3632, 3636, 3681, 3685

K	
keyans	13, 3320
keyans*	13, 4041
keyanspic	14, 3475
Keys for command provide by enumext:	
break-col	71, 72, 77, 78
item-join	71, 73, 77, 78
item-pos*	71, 73, 77, 79
item-star	71, 73, 77, 79
item-sym*	71, 73, 77, 79
Keys for environments provide by enumext:	
above*	26, 54, 55
above	26, 54, 55, 91, 94, 104, 109
after	43-45, 93, 96, 104, 109
align	26, 36, 86, 107, 117
base-fix	41
before*	43-45, 91, 104, 109
before	43-45, 94

below*	26, 54-56	4082, 4378, 4379, 4380, 4381, 4382, 4383, 4384, 4385, 4386, 4387, 4388, 4389, 4390, 4428
below	26, 54-56, 93, 96, 104, 109	
check-ans	27, 29, 30, 61-66, 69, 82, 84, 85, 91, 93, 108, 119	
columns-sep	45, 92, 95	
columns	26, 45, 48, 54, 92, 95	
first	43-45, 107	
font	35, 86, 107	
item-pos*	83	
item-sym*	27, 83, 85	
item*-sep	85	
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	
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, 73, 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	470, 492, 524, 540, 587, 658, 732, 752, 796, 832, 851, 908, 917, 996, 1013, 1424, 1535, 1782, 1843, 2017, 2056, 2092, 2097, 2237, 2529, 2550, 2867, 4307, 4376	
\l_keys_key_str	4517	
\keys_precompile:nnN	114, 4306, 4309, 4313, 4317, 4321, 4325, 4329	
\keys_set:nn	484, 812, 823, 1019, 1429, 1434, 1676, 1681, 1768, 1776, 2305, 2659, 3187, 3192, 3360, 3837,	
keyval commands:		
\keyval_parse:NNn	1549, 2122	
L		
label	538, 585, 658	
Labels provide by enumext:		
\Alph*	34, 35	
\Roman*	34, 35	
\alph*	34, 35	
\arabic*	29, 34, 35	
\roman*	34, 35	
\labelsep	98	
\labelsep	3516, 3519	
labelsep	468	
\labelwidth	35, 98	
\labelwidth	3516, 3517	
labelwidth	468	
\leftmargin	88	
\leftmargin	87, 3516	
legacy commands:		
\legacy_if:nTF	3949, 3952, 4191, 4194	
\legacy_if_gset_false:n	360	
\legacy_if_set_false:n	3951, 4193	
\legacy_if_set_true:n	3911, 3936, 3943, 3956, 4129, 4157, 4198	
\linewidth	92, 94	
\linewidth	3244, 3370, 3538, 3565, 3584, 3694, 3749	
\list	32	
\list	344	
list-indent	830	
list-offset	830	
\listparindent	3518	
listparindent	830	
\lrbbox	3966, 4204	
M		
\makebox	99	
\makebox	2227, 2229, 3012, 3980, 3988, 3992, 4217, 4221	
\makelabel	84, 86, 87, 99	
\makelabel	86, 87, 3018, 3034	
\makesavenoteenv	390	
mark-ans	2015	
mark-pos	2015, 2054	
mark-ref	2015	
mini-env	994	
mini-sep	994	
\minipage	32	
\minipage	350	
\miniright	10, 53, 1370, 3306, 3423	
\miniright*	10	
mode commands:		
\mode_if_vertical:TF	1064, 1092, 1208, 1287	
\mode_leave_vertical:	810, 821, 863, 877, 889, 901, 2225, 3010, 3978	
msg commands:		
\msg_error:nn	2287, 2292, 3351, 3355, 3442, 3502, 3827, 4068, 4075, 4391	
\msg_error:nnn	563, 610, 627, 680, 1374, 1378, 1403, 1420, 1688, 1692, 1807, 2487, 2499, 2507, 2511, 2515, 4296, 4301, 4373, 4444	
\msg_error:nnnn	2268, 2272, 2276, 2280, 3342, 3438, 3446, 4356	
\msg_error:nnnnn	511, 531, 2035	

<code>\msg_fatal:nn</code>	3176
<code>\msg_fatal:nnn</code>	422
<code>\msg_info:nnn</code>	13, 16, 21, 24, 372, 386
<code>\msg_line_context:</code>	4482, 4487, 4492, 4521, 4526, 4531, 4546, 4561, 4565, 4569, 4573, 4577, 4581, 4588, 4595, 4601, 4615, 4619, 4624, 4628, 4633, 4637, 4641, 4646, 4651, 4655, 4660, 4665, 4669, 4674, 4678, 4683, 4688, 4693, 4697, 4701
<code>\msg_log:nnn</code>	1827, 1832, 1837
<code>\msg_log:nnnnn</code>	337, 1965, 1970, 1975
<code>\msg_log:nnnnnn</code>	329
<code>\msg_new:nnn</code>	4445, 4449, 4453, 4457, 4462, 4479, 4484, 4489, 4494, 4503, 4511, 4515, 4519, 4524, 4529, 4544, 4559, 4563, 4567, 4571, 4575, 4579, 4583, 4592, 4598, 4604, 4608, 4612, 4617, 4622, 4626, 4631, 4635, 4639, 4644, 4649, 4653, 4658, 4663, 4667, 4672, 4676, 4681, 4686, 4691, 4695, 4699
<code>\msg_new:nnnn</code>	4470
<code>\msg_term:nnnn</code>	1791, 1796, 3102, 3112, 3141, 3146
<code>\msg_term:nnnnn</code>	1946
<code>\msg_warning:nn</code>	3305, 3422
<code>\msg_warning:nnnn</code>	1983, 1989, 3050, 3055, 3595, 3608, 3644, 3657
<code>\msg_warning:nnnnn</code>	1941, 1951
<code>\multicolsep</code>	92, 95
<code>\multicolsep</code>	3274, 3395
<code>\myenv</code>	4468
<code>\mypkg</code>	4468

N

<code>\NeedsTeXFormat</code>	3
<code>\newcounter</code>	425
<code>\NewDocumentCommand</code>	1370, 2252, 3434, 4280, 4334, 4398
<code>\NewDocumentEnvironment</code>	3151, 3320, 3475, 3797, 4041
<code>\newenvsc</code>	2522
<code>\newlabel</code>	34
<code>\newlabel</code>	408
<code>no-store</code>	1841
<code>\noindent</code>	103, 108
<code>\noindent</code>	3251, 3375, 3703, 3758, 3810, 4021, 4053, 4259
<code>\nointerlineskip</code>	3251, 3375, 3703, 3758
<code>noitemsep</code>	750
<code>\nopagebreak</code>	1075, 1103, 1219, 1298, 1361, 1367
<code>\normalfont</code>	2461, 2837, 2850, 4177
<code>nosep</code>	750

P

Packages:	
<code>caption</code>	102
<code>enumext</code>	24, 36, 61, 88, 96, 117
<code>enumitem</code>	34
<code>expl3</code>	99
<code>footnotehyper</code>	33
<code>hyperref</code>	28, 29, 33, 34, 73, 81, 106, 107, 117
<code>lua-visual-debug</code>	48
<code>multicol</code>	24, 117
<code>scontents</code>	24, 76, 77, 79
<code>shortlst</code>	99
<code>\par</code>	1075, 1103, 1219, 1298, 1361, 1367, 1396, 1413, 2440, 3295, 3310, 3412, 3427, 3550, 3721, 3735, 3776, 3790, 4021, 4035, 4259, 4275
<code>\parindent</code>	3998, 4227
<code>\parsep</code>	47, 50, 97, 98
<code>\parsep</code>	3132, 3515, 3522, 3527
<code>parsep</code>	750
<code>\parskip</code>	3999, 4228

<code>\partopsep</code>	98
<code>\partopsep</code>	3133, 3520
<code>partopsep</code>	750
peek commands:	
<code>\peek_meaning:NTF</code>	3888, 3902, 3919, 3930, 4106, 4120, 4137
<code>\peek_meaning_remove:NTF</code>	3895, 4113
<code>\peek_remove_spaces:n</code>	2996
<code>\phantomsection</code>	33
<code>\phantomsection</code>	397
prg commands:	
<code>\prg_do_nothing:</code>	401
<code>\prg_new_protected_conditional:Npnn</code>	199
<code>\prg_replicate:nn</code>	216
<code>\prg_return_false:</code>	203
<code>\prg_return_true:</code>	202
<code>\printkeyans</code>	16, 114, 4334
prop commands:	
<code>\prop_count:N</code>	331, 2151, 2428, 2464, 2737, 2840, 2853, 4180
<code>\prop_gput_if_not_in:Nnn</code>	2149
<code>\prop_if_exist:NTF</code>	1825, 4300
<code>\prop_item:Nn</code>	4303
<code>\prop_new:N</code>	1828
<code>\ProvidesExplPackage</code>	4

R

<code>\raggedcolumns</code>	3283, 3401
<code>\ref</code>	73, 80
<code>ref</code>	538, 585, 658
<code>\refstepcounter</code>	3958, 4200
regex commands:	
<code>\regex_match:nnTF</code>	201, 707, 709, 721, 723
<code>\regex_replace_once:nnN</code>	209
<code>\renewcommand</code>	573, 618, 635, 688
<code>\RenewDocumentCommand</code>	2889, 2957, 2992, 3018, 3034
<code>\RequirePackage</code>	17, 25
<code>resume</code>	1533
<code>resume*</code>	1533
<code>rightmargin</code>	830
<code>\Roman</code>	35, 39, 40
<code>\Roman</code>	444
<code>\roman</code>	35, 39, 40
<code>\roman</code>	445, 556, 4324

S

<code>save-ans</code>	1780
<code>save-key</code>	2090
<code>save-ref</code>	2015
<code>save-sep</code>	2015
scan commands:	
<code>\scan_stop:</code>	98, 3529, 3809, 4052, 4291, 4294
scontents internal commands:	
<code>\l_scontents_fname_out_tl</code>	2560
<code>__scontents_parse_environment_keys:n</code>	2566
<code>__scontents_rescan_tokens:n</code>	2573
<code>\l_scontents_storing_bool</code>	2558
<code>\l_scontents_writing_bool</code>	2559
seq commands:	
<code>\seq_clear:N</code>	4405
<code>\seq_const_from_clist:Nn</code>	4393
<code>\seq_count:N</code>	332, 3488, 4409
<code>\seq_gclear:N</code>	2887, 2888
<code>\seq_gput_right:Nn</code>	2158, 2900, 2901
<code>\seq_if_empty:NTF</code>	2906, 4349, 4423

©2024 by Pablo González L

<code>\tl_to_str:n</code>	1671, 1677, 1682, 4283
<code>\tl_trim_spaces:n</code>	455, 4421, 4433, 4439
<code>\tl_use:N</code>	461, 464, 582, 647, 654, 697, 932, 936, 940, 944, 948, 952, 956, 960, 964, 968, 972, 976, 980, 984, 988, 992, 2231, 2372, 2380, 2391, 2405, 2410, 2422, 2923, 2929, 2953, 2971, 2975, 2983, 3020, 3021, 3028, 3036, 3037, 3043, 3158, 3326, 3472, 3731, 3786, 3985, 3996, 4000, 4214, 4225, 4231, 4236, 4337, 4338, 4339, 4340, 4341, 4359, 4417
token commands:	
<code>\token_to_str:N</code>	408
topsep	<u>750</u>
<code>\typeout</code>	376, 379, 389, 390
U	
<code>\u</code>	210
use commands:	
<code>\use:N</code>	217, 3025, 3160
V	
<code>\use:n</code>	1547, 2120, 4289
<code>\use_none:nn</code>	400
<code>\usecounter</code>	3092, 3134
V	
<code>\value</code>	1614, 1620, 1627, 1633, 1641, 1647, 1654, 1660
vbox commands:	
<code>\vbox_set_top:Nn</code>	3729, 3784
<code>\vspace</code>	361, 811, 822, 1446, 1449, 1460, 1463, 1473, 1475, 1484, 1486, 1495, 1497, 1506, 1508, 1517, 1519, 1528, 1530, 3483, 3494, 4036, 4276
W	
widest	<u>730</u>
wrap-ans	<u>2015</u>
wrap-label	<u>468</u>
wrap-label*	<u>468</u>
wrap-opt	<u>2015</u>