

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-31^{*}

©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 the `multicol` package and the `l3seq` and `l3prop` modules.

Contents

1	Introduction	1	5	The storage system	10
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for label and ref	11
1.3	User interface	3	5.1.2	Keys for wrap and display	11
1.3.1	Internal counters	3	5.1.3	Keys for debug and checking	11
1.3.2	Support for multicol	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for minipage	4	5.2.1	Keys for command	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>keyans</code>	12
1.3.5	Support for <code>\footnote</code>	4	5.3.1	The <code>\item*</code> in <code>keyans</code>	13
2	The environments provided	4	5.4	The environment <code>keyanspic</code>	13
2.1	The environment <code>enumext</code>	4	5.4.1	The command <code>\anspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5	Printing stored content	14
2.3	The command <code>\item*</code>	5	5.5.1	The command <code>\getkeyans</code>	14
2.3.1	Keys for <code>\item*</code>	5	5.5.2	The command <code>\printkeyans</code>	14
2.4	The command <code>\item</code> in <code>enumext*</code>	5	6	Full examples	15
3	The command <code>\setenumext</code>	6	7	The way of non-enumerated lists	18
4	The <code>keyval</code> system	6	8	References	20
4.1	Keys for label and ref	6	9	Change history	20
4.2	Keys for spaces	7	10	Index of Documentation	21
4.2.1	Vertical spaces	7	11	Implementation	23
4.2.2	Horizontal spaces	8	12	Index of Implementation	116
4.3	Keys for add code	8			
4.4	Keys for start, series and resume	9			
4.5	Keys for multicol	9			
4.6	Keys for minipage	9			
4.6.1	The command <code>\miniright</code>	10			
4.6.2	The key <code>mini-right</code>	10			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-05-31.

[†]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] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TTeX. 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)



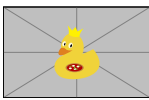
(B)



(C)



(D)



(E)

5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value


- 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 / 128

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`[5], 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

enumext.pdf

README.md

enumext.dtx

>>

>>

>>

>>

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

TDS:source/latex/enumext/

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[5] 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.

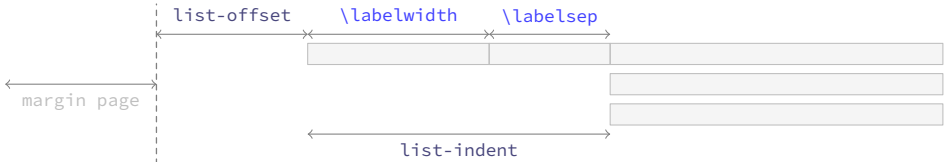


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

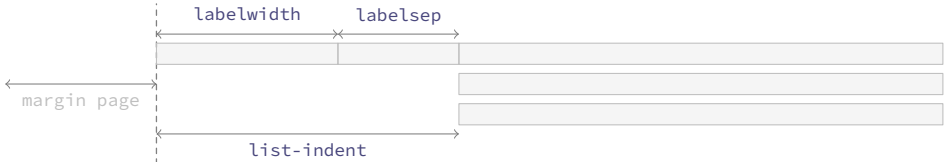


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

The user interface consists in `enumext`, `enumext*`, `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 missing 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.

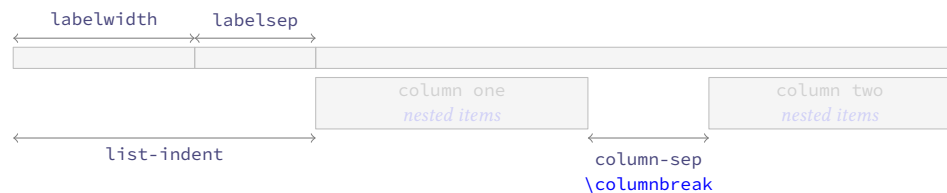


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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`[5] 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`[7] 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`[8] 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 \LaTeX , `\item` and `\item[\langle custom \rangle]` commands work in the usual way. The environment can be nested with at most “four levels” and the options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Example with columns=2

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

2.2 The environment enumext*

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

Some considerations to take into account for this environment:

- The environment cannot be nested within itself, but it can be nested within `enumext` and can contain it nested within it.
- Each “item” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that includes `labelwidth`, `labelsep` plus the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.

Example with columns=2

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

2.3 The command \item*

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

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

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

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

2.3.1 Keys for `\item*`

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

default: $\$ \star \$$

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

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

default: *by levels*

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

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

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

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

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

3 The command `\setenumext`

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

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

Displays the *current ⟨argument⟩* passed to the `\anskey` command, the current *⟨label⟩* for `\item*` and `\anspic*` commands 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 the commands `\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 commands `\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 by 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` command. 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 environment structure stored in the {⟨sequence⟩} at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested environments in which you want to use `enumext` or `enumext*` but without using the `\anskey` command, without interfering with the `check-ans` key and without storing an unwanted environment structure in the {⟨sequence⟩}.

5.2 The command `\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* 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`[7] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by \TeX will be used.

5.2.1 Keys for command

By default the {⟨content⟩} argument 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

- | | |
|---|---|
| <ul style="list-style-type: none"> ★ 1. Text containing our instructions or questions. <li style="margin-left: 20px;">★ first answer 2. Text containing our instructions or questions. <li style="margin-left: 20px;">(a) Question. <li style="margin-left: 40px;">★ second answer | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. <li style="margin-left: 20px;">★ third answer 4. Text containing our instructions or questions. <li style="margin-left: 20px;">★ fourth answer |
|---|---|

```
\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}
```

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

5.3.1 The \item* in keyans and keyans*

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

The `\item*` and `\item* [<content>]` command “*store*” the current `<label>` set by `label` key next to the `<content>` (if it is present) in *sequence* and *prop list* `{<store name>}` set by `save-ans` key in the “*first level*” of the `enumext` or `enumext*` environments.

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


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

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.
    \begin{keyans*}[nosep,columns=2]
      \item Choice
      \item* Correct choice
      \item Choice
      \item Choice
      \item Choice
    \end{keyans*}
  \item Text containing a question and image.
    \begin{keyans}[nosep,mini-env={0.4\linewidth}]
      \item Choice
      \item Choice
      \item Choice
      \item Choice
      \item*[<note>] Correct choice
      \miniright
      \includegraphics[scale=0.25]{example-image-a}
      Some text
    \end{keyans}
\end{enumext}
```

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


Some text

5.4 The environment keyanspic

keyanspic `\begin{keyanspic}[<number above, number 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.

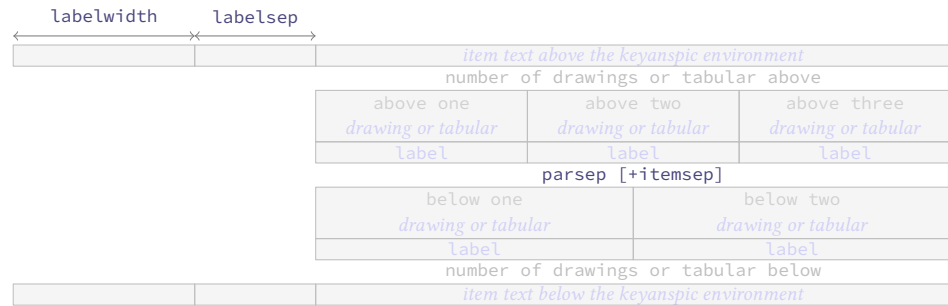


Figure 6: Representation of the **keyanspic** environment with optional argument [3,2] in **enumext**.

The optional argument determines the number drawings or tabular “above” and “below” within the environment. The vertical separation between “above” and “below” is controlled by the values set by `parsep` and `itemsep` keys passed to **keyans** environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

5.4.1 The command \anspic

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

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

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

Example

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

1. Question with images.



(A)



(B)



(C)



(D)



* (E)[note]

5.5 Printing stored content

5.5.1 The command \getkeyans

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

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

The form taken by the argument `{<store name> : <position>}` is the same as that used to generate the “internal label and ref” system when `save-ref` key are active, so to refer to a “stored content”. For example

`\getkeyans{test:4}` will return the “stored content” at position 4 of the environment in which the key `save-ans=test` was set.

5.5.2 The command `\printkeyans`

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

The command `\printkeyans` prints “all stored content” in sequence `{⟨store name⟩}` defined by `save-ans` key placing this inside the `enumext` environment or the `enumext*` environment if the *starred argument* ‘*’ is used. The “stored content” can only be accessed *after* it is stored in the sequence, if `{⟨store name⟩}` does not exist the command will return an error.

The optional argument allows managing the `⟨keys⟩` in the “first level” of the environment in which the “stored content” of the sequence `{⟨store name⟩}` will be printed, if the *starred argument* ‘*’ is used it will be `enumext*` otherwise `enumext`.

The default values for the “first level” are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep`, `first=\small`, `font=\small` and `columns=2`. For the inner levels of the environment `enumext` saved in the sequence `{⟨store name⟩}` the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the sequence `{⟨store name⟩}` it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

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

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

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[⟨print, 1⟩]{⟨keys⟩}` and `\setenumext[⟨print*⟩]{⟨keys⟩}`. If we need to set the `⟨keys⟩` for the environment `enumext` “saved” in the sequence `{⟨store name⟩}` we will use `\setenumext[⟨print, level⟩]{⟨keys⟩}` and if we need to set the `⟨keys⟩` for the environment `enumext*` “saved” in the sequence `{⟨store name⟩}` we will use `\setenumext[⟨print, *⟩]{⟨keys⟩}`.

Example

```
\begin{enumext}[save-ans=sample,columns=2,show-pos=true,nosep,save-ref=true]
  \item Factor  $3x+3y+3z$ . \anskey{$3(x+y+z)}$
  \item True False

  \begin{enumext}[nosep]
    \item \LaTeX2e\ is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

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

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)

TeX 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 ororo¹ tool to compile them.

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

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

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

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

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

2. A

3. B

4. A

Example 2

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

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

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

D $3,60 \times 10^8$ km/h.

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

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

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

B

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

C

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

D

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

2. A

3. B


4. A
- *

*

*

*

Example 3

A “simple multiple choice” test .

1. First type of questions
- A

value

B

correct

C

value

D

value
2. Second type of questions
- I.

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

II.

$\alpha = \delta$

III.

$\angle EDF = 45^\circ$
- A

I only

B

II only

C

I and II only

D

I and III only

E

I, II, and III
3. Third type of questions
- (1)

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

(2)

$\angle EDF = 45^\circ$
- A

value

B

value

C


value

D


value

E


value
4. Question with image and label below:




A




B




A




E



A



D



B

5. Question with image on left side:

A

value

B

value

C

value

D

correct

E

value

Test keys

1. B, $x = 5$

2. D

3. C, some note

4. E, A duck

5. D, other note


*


*

*


*

Example 4

A “simple worksheet” using ducks :) .


 1.

Factor $x^2 - 2x + 1$

 2.

Factor $3x + 3y + 3z$


The following questions need to be cuaqtified :)

 3.

True False

(a) $\alpha > \delta$

(b) ~~ETX~~ze is cool?

 4.

Related to Linux

(a) You use linux?

©2024 by Pablo González L


17 / 128

- (b) Usually uses the package manager?
- (c) Rate the following package and class
 - i. `xsim-exam`
 - ii. `xsim`
 - iii. `exsheets`

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

- | | | | |
|-------------------|---|---------------------------------|---|
| 1. $(x - 1)^2$ | * | (b) Yes, dnf | * |
| 2. $3(x + y + z)$ | * | (c) i. doesn't exist for now :(| * |
| 3. (a) False | * | ii. very good | * |
| (b) Very True! | * | iii. obsolete | * |
| 4. (a) Yes | * | | |

Example 5

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

1

Which choice best describes what happens in the passage?

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

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

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

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

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 \TeX 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 | * First level item |
| – Second level item | ◇ Second level item |
| • Third level item | ◦ Third level item |
| · Fourth level item | ★ Fourth level item |
| • First 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 A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
long Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

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

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`[4] 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`[10] and `l3seq`[10] modules together with the `hyperref`[7] and `enumitem`[5] packages, which did the job, but with some limitations.

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

Why list environments?

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

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

Why not random questions and other utilities

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

8 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [5] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [6] BERRY, KARL. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [7] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [8] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [9] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [10] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.

- [12] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [13] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [14] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

9 Change history

v1.0 2024-05-31 – First public release.

10 Index of Documentation

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

C

Document class:

article 2

book 2

exam 2

letter 2

report 2

\columnbreak 4, 12

\columnsep 10

Commands provide by enumext:

\anskey 4, 10-12

\anspic* 4, 10-12, 14

\anspic 14

\getkeyans 4, 12, 14

\item* 4-7, 10-13

\itemwidth 5

\item 5-7, 9, 10, 12, 13

\miniright 4, 10

\printkeyans 4, 6, 11, 15

\setenumext 4-7, 11, 13, 15

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

enumext* 4-13, 15

enumext 4-13, 15, 18

keyans* 4-11, 13

keyanspic 4, 6, 8, 10, 11, 14, 18

keyans 4-11, 13, 14, 18

Environments:

enumerate 1, 3, 5, 20

figure 5

list 3, 9, 20

minipage 3-5, 10, 20

multicols 3, 4, 10

table 5

task 5

F

\footnote 5

I

\item 3, 5

\itemsep 8

K

Keys for command provide by enumext:

break-col 12

item-join 12

item-pos* 12

item-star 12

item-sym* 12

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 19

before* 9

before 9

below* 8

below 8

check-ans 12

columns-sep 4, 10

columns 4, 8, 10

first 9

font 7

item-pos* 5, 6

item-sym* 5, 6

itemindent 8

itemsep 8, 14

labelsep 3, 5-10, 12, 19

labelwidth 3, 6, 7, 9, 10, 12, 19

labelwith 5

label 7, 9, 13, 18, 19

list-indent 3, 9

list-offset 3, 9, 19

listparindent 9

mark-ans 11, 12

mark-pos 12

mark-ref 11

mini-env 4, 8, 10

mini-right* 6, 10

mini-right 6, 10

mini-sep 4, 10

no-store 11, 12

noitemsep 8

nosep 8, 18

parsep 8, 14

partopsep 8

ref 4, 7

resume* 6, 9, 10

resume 6, 9, 10

rightmargin 8

save-ans 4, 6, 9-15

save-key 9, 11, 15

save-ref 4, 7, 11, 12, 14

save-sep 11

series 6, 9, 10

show-ans 11

show-length 7

show-pos 11, 12, 14

start 9

topsep 8

widest 7

wrap-ans 11

wrap-label* 7, 19

wrap-label 7

wrap-opt 11

L

\label 4

Labels provide by enumext:

\Alph* 7, 13

\Roman* 7

\alph* 7

<code>\arabic*</code>	7	<code>l3prop</code>	1, 20
<code>\roman*</code>	7	<code>l3seq</code>	1, 20
<code>\labelsep</code>	3, 7	<code>multicol</code>	1, 2, 4, 20
<code>\labelwidth</code>	3, 7	<code>task</code>	5, 6
<code>\linewidth</code>	10	<code>xsim</code>	2
<code>\listparindent</code>	9	<code>\parsep</code>	8
P		<code>\partopsep</code>	8
Packages:			
<code>enumerate</code>	20	R	
<code>enumext</code>	1–6, 14, 20	<code>\raggedcolumns</code>	4
<code>enumitem</code>	3–5, 9, 19, 20	<code>\ref</code>	4
<code>footnotehyper</code>	4, 5	<code>\rightmargin</code>	8
<code>hyperref</code>	4, 5, 11, 12, 20	T	
<code>l3keys</code>	6	<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 T_EXpert 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 (L^AT_EX3 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 L^AT_EX to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2023-11-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-31}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the `multicol` package is 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}[2023-03-30]
18   }
19 }
```

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.

```
20 \int_new:N \l__enumext_level_int
21 \int_new:N \l__enumext_level_h_int
22 \int_new:N \l__enumext_anskey_level_int
23 \int_new:N \l__enumext_keyans_level_int
24 \int_new:N \l__enumext_keyans_level_h_int
25 \int_new:N \l__enumext_keyans_pic_level_int
```

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

```

\l__enumext_starred_bool
\g__enumext_starred_bool
  \l_enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
  \l_enumext_standar_first_bool
\l__enumext_keyans_env_bool

```

The boolean variables `\g__enumext_starred_bool` and `\g__enumext_standar_bool` will be set to “true” when the `enumext` and `enumext*` environments are not nested with each other.

```

26 \bool_new:N \l__enumext_starred_bool
27 \bool_new:N \g__enumext_starred_bool
28 \bool_new:N \l__enumext_starred_first_bool
29 \bool_new:N \l__enumext_standar_bool
30 \bool_new:N \g__enumext_standar_bool
31 \bool_new:N \l__enumext_standar_first_bool
32 \bool_new:N \l__enumext_keyans_env_bool

```

(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` (§11.8) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.11).

```

33 \cs_set_protected:Npn \__enumext_tmp:n #1
34 {
35   \tl_new:c { \l__enumext_counter_#1_tl }
36 }
37 \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 (§11.11).

```

38 \tl_const:Nn \c__enumext_counter_style_tl
39 { { { arabic } { roman } { Roman } { alph } { Alph } } }
40 \tl_new:N \l__enumext_ref_key_arg_tl
41 \tl_new:N \l__enumext_ref_the_count_tl
42 \cs_set_protected:Npn \__enumext_tmp:n #1
43 {
44   \tl_new:c { \l__enumext_renew_the_count_#1_tl }
45   \tl_new:c { \l__enumext_the_counter_#1_tl }
46   \tl_set:ce { \l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
47 }
48 \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_item_symbol_tl
  \g__enumext_standar_series_tl
  \g__enumext_starred_series_tl

```

Internal variables used by `resume`, `resume*` and `series` keys. The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§11.27).

```

49 \int_new:N \g__enumext_resume_int
50 \int_new:N \g__enumext_resume_vii_int
51 \tl_new:N \l__enumext_resume_name_tl
52 \bool_new:N \l__enumext_resume_active_bool
53 \tl_new:N \g__enumext_item_symbol_tl
54 \tl_new:N \g__enumext_standar_series_tl
55 \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` (§11.12) and `label` (§11.10) keys.

```

56 \dim_new:N \l__enumext_current_widest_dim
57 \tl_new:N \g__enumext_counter_styles_tl
58 \tl_new:N \g__enumext_widest_label_tl
59 \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 (§11.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.31.1) which determines the internal values for `\leftmargin` and `\itemindent`.

```

60 \cs_set_protected:Npn \__enumext_tmp:n #1
61 {

```

```

62     \bool_new:c { \__enumext_leftmargin_tmp_#1_bool }
63     \dim_new:c { \__enumext_leftmargin_tmp_#1_dim }
64     \dim_new:c { \__enumext_leftmargin_#1_dim }
65     \dim_new:c { \__enumext_itemindent_#1_dim }
66 }
67 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `__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 (§11.18).

```

68 \cs_set_protected:Npn \__enumext_tmp:n #1
69 {
70     \skip_new:c { \__enumext_multicols_above_#1_skip }
71     \skip_new:c { \__enumext_multicols_below_#1_skip }
72 }
73 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for `__enumext_multicols_above_X_skip` and `__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 (§11.19.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.17, §11.19).

```

74 \int_new:N \g__enumext_minipage_stat_int
75 \skip_new:N \l__enumext_minipage_left_skip
76 \skip_new:N \l__enumext_minipage_right_skip
77 \skip_new:N \l__enumext_minipage_after_skip
78 \skip_new:N \g__enumext_minipage_right_skip
79 \skip_new:N \g__enumext_minipage_after_skip
80 \cs_set_protected:Npn \__enumext_tmp:n #1
81 {
82     \dim_new:c { \__enumext_minipage_left_#1_dim }
83     \bool_new:c { \__enumext_minipage_active_#1_bool }
84 }
85 \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 (§11.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.10). 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

```

86 \cs_set_protected:Npn \__enumext_tmp:n #1
87 {
88     \bool_new:c { \__enumext_wrap_label_#1_bool }
89     \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
90     \int_new:c { \__enumext_start_#1_int }
91     \tl_new:c { \__enumext_fake_item_indent_#1_tl }
92     \tl_new:c { \__enumext_label_fill_left_#1_tl }
93     \tl_new:c { \__enumext_label_fill_right_#1_tl }
94     \bool_new:c { \__enumext_vspace_a_star_#1_bool }
95     \bool_new:c { \__enumext_vspace_b_star_#1_bool }
96 }
97 \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_columns_join_int
\l__enumext_store_keyans_label_tl
\l__enumext_store_keyans_item_opt_tl
\l__enumext_keyans_item_opt_tl
\l__enumext_keyans_tmpa_tl

```

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

The variable `\l__enumext_store_name_tl` sets the name for the storage in *sequence* and *prop list*, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the `check-ans` key (§?).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§11.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§11.29.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§11.34.1) for the `keyanspic` environment.

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

```

98 \bool_new:N \l__enumext_store_active_bool
99 \tl_new:N \l__enumext_store_name_tl
100 \tl_new:N \g__enumext_store_name_tl

```



```

101 \tl_new:N \l__enumext_store_anskey_arg_tl
102 \int_new:N \l__enumext_store_columns_join_int
103 \tl_new:N \l__enumext_store_keyans_label_tl
104 \tl_new:N \l__enumext_store_keyans_item_opt_tl
105 \tl_new:N \l__enumext_keyans_item_opt_tl
106 \tl_new:N \l__enumext_keyans_tmpa_tl

```

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

Internal variables used by the command `\setenumext` (§11.39).

```

107 \tl_new:N \l__enumext_setkey_tmpa_tl
108 \tl_new:N \l__enumext_setkey_tmpp_tl
109 \int_new:N \l__enumext_setkey_tmpa_int
110 \seq_new:N \l__enumext_setkey_tmpa_seq
111 \seq_new:N \l__enumext_setkey_tmpp_seq

```

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

Internal variables used by `[<key = val>]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§11.38) and `save-key` key.

```

112 \tl_new:N \l__enumext_print_keyans_starred_tl
113 \cs_set_protected:Npn \__enumext_tmp:n #1
114 {
115   \tl_new:c { \l__enumext_store_save_key_#1_tl }
116   \bool_new:c { \l__enumext_store_save_key_#1_bool }
117   \tl_new:c { \l__enumext_store_active_keys_#1_tl }
118   \tl_new:c { \l__enumext_print_keyans_#1_tl }
119   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
120 }
121 \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.)

Internal variables for “*storage system*” mechanism used by `\anskey` (§11.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§11.23).

```

122 \bool_new:N \l__enumext_show_answer_bool
123 \bool_new:N \l__enumext_show_position_bool
124 \tl_new:N \l__enumext_mark_ref_sym_tl
125 \tl_new:N \l__enumext_mark_answer_sym_tl
126 \str_new:N \l__enumext_mark_position_str

```

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

Internal variables used by `keyanspic` environment (§11.34.2).

```

127 \seq_new:N \l__enumext_keyans_pic_body_seq
128 \dim_new:N \l__enumext_keyans_pic_width_dim
129 \int_new:N \l__enumext_keyans_pic_above_int
130 \int_new:N \l__enumext_keyans_pic_below_int
131 \skip_new:N \l__enumext_keyans_pic_above_skip

```

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

Internal variables used by “*check answer*” mechanism (§11.22.3) 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.

```

132 \bool_new:N \l__enumext_check_answers_bool
133 \bool_new:N \l__enumext_check_ans_key_bool
134 \bool_new:N \g__enumext_check_ans_key_bool
135 \tl_new:N \l__enumext_check_start_line_env_tl
136 \tl_new:N \g__enumext_start_line_tl
137 \tl_new:N \g__enumext_envir_name_tl
138 \int_new:N \g__enumext_check_starred_cmd_int
139 \int_new:N \g__enumext_item_anskey_int
140 \int_new:N \g__enumext_item_number_int
141 \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 (§11.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
142 \bool_new:N \l__enumext_hyperref_bool
143 \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_store_write_aux_file_tl
\l__enumext_label_copy_X_tl
```

Internal variables are used when executing the `save-ref` key. 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` and the variable `\l__enumext_store_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```
144 \tl_new:N \l__enumext_newlabel_arg_one_tl
145 \tl_new:N \l__enumext_newlabel_arg_two_tl
146 \tl_new:N \l__enumext_store_write_aux_file_tl
147 \cs_set_protected:Npn \__enumext_tmp:n #1
148 {
149   \tl_new:c { l__enumext_label_copy_#1_tl }
150 }
151 \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`.

```
152 \int_new:N \g__enumext_footnote_int
153 \seq_new:N \g__enumext_footnote_arg_seq
154 \seq_new:N \g__enumext_footnote_int_seq
```

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

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

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

```
155 \cs_set_protected:Npn \__enumext_tmp:n #1
156 {
157   \bool_new:c { l__enumext_item_starred_#1_bool }
158   \int_new:c { l__enumext_item_column_pos_#1_int }
159   \int_new:c { g__enumext_item_count_all_#1_int }
160   \int_new:c { l__enumext_joined_item_#1_int }
161   \int_new:c { l__enumext_joined_item_aux_#1_int }
162   \int_new:c { l__enumext_tmpa_#1_int }
163   \box_new:c { l__enumext_item_text_#1_box }
164   \dim_new:c { l__enumext_joined_width_#1_dim }
165   \dim_new:c { l__enumext_item_width_#1_dim }
166   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
167   \str_new:c { l__enumext_align_label_#1_str }
168   \bool_new:c { g__enumext_minipage_active_#1_bool }
169   \tl_new:c { g__enumext_miniright_code_#1_tl }
170   \bool_new:c { g__enumext_minipage_center_#1_bool }
171   \dim_new:c { g__enumext_minipage_right_#1_dim }
172   \skip_new:c { g__enumext_minipage_right_#1_skip }
173 }
174 \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`.

```
175 \clist_const:Nn \c__enumext_all_envs_clist
176 {
177   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
178   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
179 }
```

(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.

```
180 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
181 {
182   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
183 }
```

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

`__enumext_after_env:nn`

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

```
184 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
185 {
186   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
187 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
188 \cs_new:Nn \__enumext_level:
189 {
190   \int_to_roman:n { \__enumext_level_int }
191 }
```

(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?](#).

```
192 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
193 {
194   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
195   { \prg_return_true: }
196   { \prg_return_false: }
197 }
```

(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.

```
198 \cs_new_protected:Nn \__enumext_regex_counter_style:
199 {
200   \tl_map_inline:Nn \c__enumext_counter_style_tl
201   {
202     \regex_replace_once:nnN { \c{##1}\* }
203     { \c{##1}\cB{\u{\__enumext_ref_the_count_tl}\cE} } \__enumext_ref_key_arg_tl
204   }
205 }
```

(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.

```
206 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
207 {
208   * ~ #2
209   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
210   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
211 }
```

(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.

```

212 \cs_new_protected:Nn __enumext_is_not_nested:
213 {
214   \str_case:en { \@currentenv }
215   {
216     {enumext}
217     {
218       \bool_lazy_and:nnT
219       { \bool_not_p:n { \g__enumext_standar_bool } }
220       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
221       {
222         \bool_gset_true:N \g__enumext_standar_bool
223       }
224     }
225     {enumext*}
226     {
227       \bool_lazy_and:nnT
228       { \bool_not_p:n { \g__enumext_starred_bool } }
229       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
230       {
231         \bool_gset_true:N \g__enumext_starred_bool
232       }
233     }
234   }
235 }

```

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

```

236 \cs_new_protected:Nn __enumext_is_on_first_level:
237 {
238   \bool_lazy_all:nT
239   {
240     { \bool_if_p:N \g__enumext_standar_bool }
241     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
242     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
243   }
244   {
245     \bool_set_true:N \l__enumext_standar_first_bool
246     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
247     \tl_gset:Nn \g__enumext_start_line_tl
248     {
249       on ~ line ~ \exp_not:V \inputlineno
250     }
251   }
252   \bool_lazy_all:nT
253   {
254     { \bool_if_p:N \g__enumext_starred_bool }
255     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
256     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
257   }
258   {
259     \bool_set_true:N \l__enumext_starred_first_bool
260     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
261     \tl_gset:Nn \g__enumext_start_line_tl
262     {
263       on ~ line ~ \exp_not:V \inputlineno
264     }
265   }
266 }

```

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

`__enumext_keyans_save_start_line:`

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

```

267 \cs_new_protected:Nn \__enumext_keyans_save_start_line:
268 {
269   \str_case:en { \@currenvir }
270   {
271     {keyans}
272     {
273       \tl_set:Nc \l__enumext_check_start_line_env_tl
274       {
275         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
276       }
277     }
278     {keyans*}
279     {
280       \tl_set:Nc \l__enumext_check_start_line_env_tl
281       {
282         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
283       }
284     }
285     {keyanspic}
286     {
287       \tl_set:Nc \l__enumext_check_start_line_env_tl
288       {
289         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
290       }
291     }
292   }
293 }

```

(End of definition for `__enumext_keyans_save_start_line:`.)

11.5.2 Utilities for log and terminal

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

```

294 \cs_new_protected:Nn \__enumext_reset_global_vars:
295 {
296   \__enumext_reset_global_int:
297   \__enumext_reset_global_bool:
298   \__enumext_reset_global_tl:
299 }
300 \cs_new_protected:Nn \__enumext_reset_global_int:
301 {
302   \int_gzero:N \g__enumext_item_number_int
303   \int_gzero:N \g__enumext_item_anskey_int
304   \int_gzero:N \g__enumext_item_answer_diff_int
305 }
306 \cs_new_protected:Nn \__enumext_reset_global_bool:
307 {
308   \bool_gset_false:N \g__enumext_check_ans_key_bool
309   \bool_gset_false:N \g__enumext_standar_bool
310   \bool_gset_false:N \g__enumext_starred_bool
311 }
312 \cs_new_protected:Nn \__enumext_reset_global_tl:
313 {
314   \tl_gclear:N \g__enumext_store_name_tl
315   \tl_gclear:N \g__enumext_start_line_tl
316   \tl_gclear:N \g__enumext_envir_name_tl
317 }

```

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

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

```

318 \cs_new_protected:Nn \__enumext_log_global_vars:
319 {
320   \msg_log:nneeee { enumext } { prop-seq-int-hook }
321   { \g__enumext_store_name_tl }
322   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
323   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
324   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
325 }

```

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.

```

326 \cs_new_protected:Nn \__enumext_log_answer_vars:
327 {
328   \msg_log:nneee { enumext } { item-answer-hook }
329   { \int_use:N \g__enumext_item_number_int }
330   { \int_use:N \g__enumext_item_anskey_int }
331   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
332 }

```

(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.

```

333 \__enumext_at_begin_document:n
334 {
335   \cs_new_eq:NN \__enumext_start_list:nn \list
336   \cs_new_eq:NN \__enumext_stop_list: \endlist
337   \cs_new_eq:NN \__enumext_item_std:w \item
338 }

```

(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.

```

339 \__enumext_at_begin_document:n
340 {
341   \cs_new_eq:NN \__enumext_minipage:w \minipage
342   \cs_new_eq:NN \__enumext_endminipage: \endminipage
343 }

```

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

11.7 Compatibility with hyperref and footnotehyper

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

```

344 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
345 \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”.

```

346 \cs_new_protected:Nn \__enumext_after_hyperref:
347 {
348   \IfPackageLoadedTF { hyperref }
349   {
350     \msg_info:nnn { enumext } { package-load } { hyperref }
351     \bool_set_true:N \l__enumext_hyperref_bool
352     \IfHyperBoolean{hyperfootnotes}
353     {
354       \typeout{hyperfootnotes=true}

```



```

355         \bool_set_true:N \l__enumext_footnotes_key_bool
356     }
357     { \typeout{hyperfootnotes=false} }
358 }
359 { }

```

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`.

```

360 \bool_if:NT \l__enumext_footnotes_key_bool
361 {
362     \IfPackageLoadedTF { footnotehyper }
363     {
364         \msg_info:nnn { enumext } { package-load } { footnotehyper }
365     }
366     {
367         \typeout{No ~ footnotehyper ~ load}
368         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
369         \bool_set_false:N \l__enumext_footnotes_key_bool
370     }
371 }

```

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.

```

372 \bool_if:NTF \l__enumext_hyperref_bool
373 {
374     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
375     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
376 }
377 {
378     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
379     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
380 }
381 }

```

(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.

```

382 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
383 {
384     \protected@write \@auxout { }
385     {
386         \token_to_str:N \newlabel {#1}
387         {
388             {#2}
389             \bool_if:NT \l__enumext_hyperref_bool
390             { { \thepage } {#2} {#1} }
391             { }
392         }
393     }
394     \__enumext_hypertarget:nn {#1} { }
395     \__enumext_phantomsection:
396 }

```

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

11.8 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.

```

397 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
398 {
399   \cs_if_exist:cTF { c@ #2 }
400   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
401   {
402     \tl_set:Nn #1 { #2 }
403     \newcounter { #2 }
404   }
405 }

```

(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      406 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     407 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    408 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     409 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv      410 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvii    411 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii   412 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
            413 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

11.9 Definition of labels

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

__enumext_register_counter_style:Nn

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

```

414 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
415 {
416   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
417   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
418 }
419 \__enumext_register_counter_style:Nn \arabic { 0 }
420 \__enumext_register_counter_style:Nn \Alph { M }
421 \__enumext_register_counter_style:Nn \alph { m }
422 \__enumext_register_counter_style:Nn \Roman { VIII }
423 \__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.

```

424 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
425 {
426   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
427   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
428 }
429 \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.

```

430 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
431 {
432   \tl_clear_new:N #1
433   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
434   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
435   \tl_map_inline:Nn \g__enumext_counter_styles_tl
436   {
437     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }

```

```

438     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
439     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
440   }
441   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
442   { \tl_use:N \g__enumext_widest_label_tl }
443   \tl_set_eq:cN { the #2 } #1
444 }
445 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

(End of definition for __enumext_label_style:Nnn.)

11.10 Setting keys associated with label

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

```

446 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
447 {
448   \keys_define:nn { enumext / #1 }
449   {
450     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
451     font      .value_required:n = true,
452     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
453     labelsep  .initial:n   = { 0.3333em },
454     labelsep  .value_required:n = true,
455     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
456     labelwidth .value_required:n = true,
457     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
458     wrap-label .initial:n   = { ##1 },
459     wrap-label .value_required:n = true,
460     wrap-label* .code:n = {
461       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
462       \keys_set:nn { enumext / #1 } { wrap-label = { ##1 } }
463     },
464     wrap-label* .value_required:n = true,
465   }
466 }
467 \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.

```

468 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
469 {
470   \keys_define:nn { enumext / #1 }
471   {
472     align .choice:,
473     align / left .code:n =
474     {
475       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
476       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
477     },
478     align / right .code:n =
479     {
480       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
481       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
482     },
483     align / center .code:n =
484     {
485       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
486       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
487     },
488     align .initial:n = left,
489     align .value_required:n = true,
490   }
491 }
492 \clist_map_inline:nn
493 {
494   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}

```

```

495 }
496 { \__enumext_tmp:nn #1 }

497 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
498 {
499   \keys_define:nn { enumext / #1 }
500   {
501     align .choice:,
502     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
503     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
504     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
505     align .initial:n = left,
506     align .value_required:n = true,
507   }
508 }
509 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for *align*.)

11.11 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.

11.11.1 Define and set label and ref keys for enumext environment

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

```

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
510 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
511 {
512   \keys_define:nn { enumext / #1 }
513   {
514     label .code:n = {
515       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
516       { l__enumext_counter_#2_tl } {##1}
517       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
518       \l__enumext_current_widest_dim
519     },
520     label .initial:n = #3,
521     label .value_required:n = true,
522     ref .code:n = \__enumext_standar_ref:n {##1},
523     ref .value_required:n = true,
524   }
525 }
526 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
527 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
528 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
529 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

```

530 \cs_new_protected:Npn \__enumext_standar_ref:n #1
531 {
532   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
533   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
534   {
535     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
536   }
537   {
538     \tl_set_eq:Nc
539     \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
540     \__enumext_regex_counter_style:
541     \tl_set_eq:Nc
542     \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
543     \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }

```

```

544         {
545             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
546             { \exp_not:V \l__enumext_ref_key_arg_tl }
547         }
548     }
549 }

```

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

```

550 \cs_new_protected:Nn \__enumext_standar_ref:
551 {
552     \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
553     {
554         \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
555     }
556 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
557 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
558 {
559     \keys_define:nn { enumext / #1 }
560     {
561         label .code:n = {
562             \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
563             { \l__enumext_counter_#2_tl } {##1}
564             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
565             \l__enumext_current_widest_dim
566         },
567         label .initial:n = #3,
568         label .value_required:n = true,
569         ref .code:n = \__enumext_starred_ref:n {##1},
570         ref .value_required:n = true,
571     }
572 }
573 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
574 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

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

```

575 \cs_new_protected:Npn \__enumext_starred_ref:n #1
576 {
577     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
578     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
579     {
580         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
581         {
582             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
583         }
584         {
585             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
586             \__enumext_regex_counter_style:
587             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
588             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
589             {
590                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
591                 { \exp_not:V \l__enumext_ref_key_arg_tl }
592             }
593         }
594     }
595     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
596     {
597         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
598         {
599             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
600         }

```

```

601     {
602         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
603         \__enumext_regex_counter_style:
604         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
605         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
606         {
607             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
608             { \exp_not:V \l__enumext_ref_key_arg_tl }
609         }
610     }
611 }
612 }

```

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.

```

613 \cs_new_protected:Nn \__enumext_starred_ref:
614 {
615     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
616     {
617         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
618         {
619             \tl_use:N \l__enumext_renew_the_count_vii_tl
620         }
621     }
622     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
623     {
624         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
625         {
626             \tl_use:N \l__enumext_renew_the_count_viii_tl
627         }
628     }
629 }

```

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

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

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

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
630 \keys_define:nn { enumext / keyans }
631 {
632     label .code:n = {
633         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
634         { \l__enumext_counter_v_tl } {#1}
635         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
636         \l__enumext_current_widest_dim
637         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
638         { \l__enumext_counter_vi_tl } {#1}
639         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
640         \l__enumext_current_widest_dim
641     },
642     label .initial:n = (\Alph*),
643     label .value_required:n = true,
644     ref .code:n = \__enumext_keyans_ref:n {#1},
645     ref .value_required:n = true,
646 }

```

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

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

```

647 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
648 {
649     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
650     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
651     {
652         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
653     }
654     {
655         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
656         \__enumext_regex_counter_style:
657         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl

```

```

658     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
659     {
660         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
661         { \exp_not:V \l__enumext_ref_key_arg_tl }
662     }
663 }
664 }

```

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

```

665 \cs_new_protected:Nn \__enumext_keyans_ref:
666 {
667     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
668     {
669         \tl_use:N \l__enumext_renew_the_count_v_tl
670     }
671 }

```

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

11.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

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

```

#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3: <integer or string>

```

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

```

672 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
673 {
674     \__enumext_if_is_int:nTF { #3 }
675     {
676         \int_set:Nn #2 {#3}
677     }
678     {
679         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
680         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
681         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
682         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
683     }
684 }
685 \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.

```

686 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
687 {
688     \__enumext_if_is_int:nTF {#4}
689     {
690         \setcounter{enumX#1} { #4 }
691     }
692     {
693         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
694         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
695         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
696         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
697     }
698     \__enumext_label_width_by_box:cv
699     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
700 }
701 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```


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

start
widest
`\l__enumext_start_X_int`

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

```

702 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
703 {
704   \keys_define:nn { enumext / #1 }
705   {
706     start .code:n = {
707       \__enumext_start_from:ccn
708       { l__enumext_label_#2_tl }
709       { l__enumext_start_#2_int } {##1}
710     },
711     start .initial:n = 1,
712     widest .code:n = {
713       \__enumext_widest_from:nccn {#2}
714       { l__enumext_label_#2_tl }
715       { l__enumext_labelwidth_#2_dim } {##1}
716     },
717     widest .value_required:n = true,
718     start .value_required:n = true,
719   }
720 }
721 \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`.)

11.13 Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```

722 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
723 {
724   \keys_define:nn { enumext / #1 }
725   {
726     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
727     topsep .initial:n = {#3},
728     topsep .value_required:n = true,
729     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
730     partopsep .initial:n = {#4},
731     partopsep .value_required:n = true,
732     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
733     parsep .initial:n = {#5},
734     parsep .value_required:n = true,
735     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
736     itemsep .initial:n = {#6},
737     itemsep .value_required:n = true,
738     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
739     noitemsep .value_forbidden:n = true,
740     noset .meta:n = {
741       itemsep = 0pt, parsep = 0pt,
742       topsep = 0pt, partopsep = 0pt,
743     },
744     noset .value_forbidden:n = true,
745   }
746 }

```

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

```

747 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
748 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
749 { 4.0pt plus 2.0pt minus 1.0pt }
750 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
751 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
752 { 2.0pt plus 1.0pt minus 1.0pt }
753 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
754 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
755 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
756 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
757 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
758 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
759 { 2.0pt plus 1.0pt minus 1.0pt }
760 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }

```

```

761 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
762 { 4.0pt plus 2.0pt minus 1.0pt }
763 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
764 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
765 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

11.14 Setting keys for horizontal spaces

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

```

766 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
767 {
768   \keys_define:nn { enumext / #1 }
769   {
770     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
771     itemindent .value_required:n = true,
772     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
773     rightmargin .value_required:n = true,
774     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
775     listparindent .value_required:n = true,
776     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
777     list-offset .value_required:n = true,
778     list-indent .code:n =
779       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
780       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
781     list-indent .value_required:n = true,
782   }
783 }
784 \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.

```

785 \cs_set_protected:Npn \__enumext_tmp:n #1
786 {
787   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
788 }
789 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

11.14.1 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.

```

790 \cs_set_protected:Nn \__enumext_fake_item:
791 {
792   \dim_compare:nNnT
793     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
794     >
795     { \c_zero_dim }
796   {
797     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
798     {
799       \exp_not:N \mode_leave_vertical:
800       \exp_not:n { \skip_horizontal:n }
801       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } } }
802     \ignorespaces
803   }
804 }
805 }
806 \cs_set_protected:Nn \__enumext_keyans_fake_item:
807 {
808   \dim_compare:nNnT
809     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
810     {
811       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
812       {
813         \exp_not:N \mode_leave_vertical:
814         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim

```

```

815     }
816   }
817 }
818 \cs_set_protected:Nn \__enumext_fake_item_vii:
819 {
820   \dim_compare:nNnT
821   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
822   {
823     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
824     {
825       \exp_not:N \mode_leave_vertical:
826       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
827     }
828   }
829 }
830 \cs_set_protected:Nn \__enumext_fake_item_viii:
831 {
832   \dim_compare:nNnT
833   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
834   {
835     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
836     {
837       \exp_not:N \mode_leave_vertical:
838       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
839     }
840   }
841 }

```

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

11.15 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.

```

842 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
843 {
844   \keys_define:nn { enumext / #1 }
845   {
846     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
847     show-length .initial:n = false,
848   }
849 }
850 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.16 Setting before, after and first keys

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

```

before*
after
first
851 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
852 {
853   \keys_define:nn { enumext / #1 }
854   {
855     before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
856     before .value_required:n = true,
857     before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
858     before* .value_required:n = true,
859     after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
860     after .value_required:n = true,
861     first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
862     first .value_required:n = true,
863   }
864 }
865 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.16.1 Functions for before, after and first keys in enumext

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

```
866 \cs_new_protected:Nn \__enumext_before_args_exec:
867 {
868   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
869 }
```

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

```
870 \cs_new_protected:Nn \__enumext_before_keys_exec:
871 {
872   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
873 }
```

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

```
874 \cs_new_protected:Nn \__enumext_after_stop_list:
875 {
876   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
877 }
```

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

```
878 \cs_new_protected:Nn \__enumext_after_args_exec:
879 {
880   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
881 }
```

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

11.16.2 Functions for before, after and first keys in keyans

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

```
882 \cs_new_protected:Nn \__enumext_before_args_exec_v:
883 {
884   \tl_use:N \l__enumext_before_starred_key_v_tl
885 }
```

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

```
886 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
887 {
888   \tl_use:N \l__enumext_before_no_starred_key_v_tl
889 }
```

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

```
890 \cs_new_protected:Nn \__enumext_after_stop_list_v:
891 {
892   \tl_use:N \l__enumext_after_stop_list_v_tl
893 }
```

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.`

```
894 \cs_new_protected:Nn \__enumext_after_args_exec_v:
895 {
896   \tl_use:N \l__enumext_after_list_args_v_tl
897 }
```

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

11.16.3 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.

```
898 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
899 {
900   \tl_use:N \l__enumext_before_starred_key_vii_tl
901 }
902 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
903 {
904   \tl_use:N \l__enumext_before_starred_key_viii_tl
905 }
```

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⟩`.

```
906 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
907 {
908   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
909 }
910 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
911 {
912   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
913 }
```

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

```
914 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
915 {
916   \tl_use:N \l__enumext_after_stop_list_vii_tl
917 }
918 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
919 {
920   \tl_use:N \l__enumext_after_stop_list_viii_tl
921 }
```

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`.

```
922 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
923 {
924   \tl_use:N \l__enumext_after_list_args_vii_tl
925 }
926 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
927 {
928   \tl_use:N \l__enumext_after_list_args_viii_tl
929 }
```

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

11.17 Setting keys for multicol 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` and `keyans` environments.

```
930 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
931 {
932   \keys_define:nn { enumext / #1 }
933   {
934     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
935     mini-env .value_required:n = true,
936     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
937     mini-sep .initial:n = 0.3333em,
938     mini-sep .value_required:n = true,
939     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
940     columns-sep .value_required:n = true,
941     columns .int_set:c = { l__enumext_columns_#2_int },
942     columns .initial:n = 1,
943     columns .value_required:n = true,
944   }
```

```

945 }
946 \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 default value for `columns` key are `2` and the command `\miniright` is not available, so we will add the keys `mini-right` and `mini-right*` to implement support for `minipage`.

```

947 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
948 {
949   \keys_define:nn { enumext / #1 }
950   {
951     columns      .initial:n = 2,
952     mini-right   .tl_gset:c = { g__enumext_miniright_code_#2_tl },
953     mini-right   .value_required:n = true,
954     mini-right*  .code:n     = {
955       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
956       \keys_set:nn { enumext / #1 } { miniright = {##1} }
957     },
958     mini-right*  .value_required:n = true,
959   }
960 }
961 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.18 Adjustment of vertical spaces for multicols

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

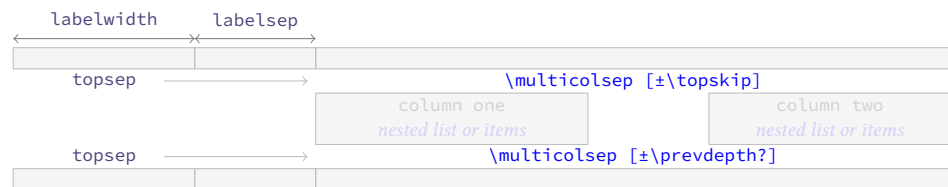


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

11.18.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

962 \cs_new_protected:Nn \__enumext_multi_set_vskip:
963 {
964   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
965   {
966     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
967   }
968   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
969   {
970     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
971   }
972   \__enumext_add_pre_parsep:
973 }

```

(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*.

```

974 \cs_new_protected:Nn \__enumext_add_pre_parsep:
975 {
976   \int_case:nn { \l__enumext_level_int }
977   {
978     { 2 }{
979       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
980       {
981         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
982       }
983     }
984     { 3 }{
985       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
986       {
987         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
988       }
989     }
990     { 4 }{
991       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
992       {
993         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
994       }
995     }
996   }
997 }

```

(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*.

```

998 \cs_new_protected:Nn \__enumext_multi_addvspace:
999 {
1000   \__enumext_multi_set_vskip:
1001   \mode_if_vertical:T
1002   {
1003     \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1004     {
1005       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1006     }
1007     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1008     {
1009       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1010     }
1011   }
1012   \par\nopagebreak
1013   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1014 }

```

(End of definition for `__enumext_multi_addvspace:`.)

11.18.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1015 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1016 {
1017   \skip_set:Nn \l__enumext_multicols_above_v_skip
1018   {
1019     \l__enumext_topsep_v_skip
1020   }
1021   \skip_set:Nn \l__enumext_multicols_below_v_skip
1022   {
1023     \l__enumext_topsep_v_skip
1024   }
1025 }
1026 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1027 {

```



```

1028 \__enumext_keyans_multi_set_vskip:
1029 \mode_if_vertical:T
1030 {
1031   \skip_add:Nn \l__enumext_multicols_above_v_skip
1032   {
1033     \skip_use:N \l__enumext_partopsep_v_skip
1034   }
1035   \skip_add:Nn \l__enumext_multicols_below_v_skip
1036   {
1037     \skip_use:N \l__enumext_partopsep_v_skip
1038   }
1039 }
1040 \par\nopagebreak
1041 \addvspace{ \l__enumext_multicols_above_v_skip }
1042 }

```

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

11.19 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.

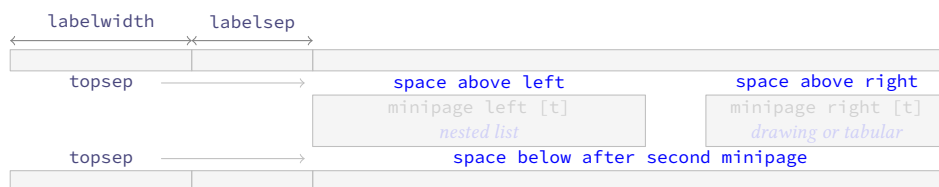


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

`__enumext_mini_env*` Creates a `__enumext_mini_env*` environment (custom version of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\vspace{0pt}` to maintain alignment on “top”. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only.

```

1043 \DeclareDocumentEnvironment{\__enumext_mini_env*}{ m }
1044 {
1045   \__enumext_minipage:w [ t ] { #1 }
1046   \legacy_if_gset_false:n { @minipage }
1047   \vspace { 0pt }
1048 }
1049 { \__enumext_endminipage: }

```

(End of definition for `__enumext_mini_env*`.)

11.19.1 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`.

```

1050 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1051 {

```

```

1052 \int_compare:nNnTF
1053 { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1054 {

```

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.

```

1055 \skip_if_eq:nnTF
1056 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1057 {
1058   \skip_set:Nn \l__enumext_minipage_left_skip
1059   {
1060     -0.150\box_dp:N \strutbox
1061   }
1062   \skip_set:Nn \l__enumext_minipage_right_skip
1063   {
1064     0.695\box_dp:N \strutbox
1065   }
1066   \skip_set:Nn \l__enumext_minipage_after_skip
1067   {
1068     \box_dp:N \strutbox
1069   }
1070   \__enumext_zero_parsep:
1071 }
1072 {
1073   \skip_set:Nn \l__enumext_minipage_left_skip
1074   {
1075     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1076   }
1077   \skip_set:Nn \l__enumext_minipage_right_skip
1078   {
1079     0.695\box_dp:N \strutbox
1080   }
1081   \skip_set:Nn \l__enumext_minipage_after_skip
1082   {
1083     1.85\box_dp:N \strutbox
1084     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1085   }
1086 }
1087 }
1088 {

```

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.

```

1089 \skip_if_eq:nnTF
1090 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1091 {
1092   \skip_set:Nn \l__enumext_minipage_left_skip
1093   {
1094     0.5\box_dp:N \strutbox
1095     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1096   }
1097   \skip_set:Nn \l__enumext_minipage_right_skip
1098   {
1099     \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1100   }
1101   \skip_set:Nn \l__enumext_minipage_after_skip
1102   {
1103     1.6\box_dp:N \strutbox
1104   }
1105 }
1106 {
1107   \skip_set:Nn \l__enumext_minipage_left_skip
1108   {
1109     0.5875\box_dp:N \strutbox
1110     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1111   }
1112   \skip_set:Nn \l__enumext_minipage_right_skip
1113   {
1114     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1115     + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }

```

```

1116         }
1117         \skip_set:Nn \l__enumext_minipage_after_skip
1118         {
1119             0.325\box_dp:N \strutbox
1120             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1121         }
1122     }
1123 }
1124 }

```

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

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

```

1125 \cs_new_protected:Nn \l__enumext_zero_parsep:
1126 {
1127     \int_case:nn { \l__enumext_level_int }
1128     {
1129         { 2 }{
1130             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1131             {
1132                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1133             }
1134         }
1135         { 3 }{
1136             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1137             {
1138                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1139             }
1140         }
1141         { 4 }{
1142             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1143             {
1144                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1145             }
1146         }
1147     }
1148 }

```

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

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

```

1149 \cs_new_protected:Nn \l__enumext_mini_addvspace:
1150 {
1151     \l__enumext_mini_set_vskip:
1152     \mode_if_vertical:T
1153     {
1154         \skip_add:Nn \l__enumext_minipage_left_skip
1155         {
1156             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1157         }
1158         \skip_add:Nn \l__enumext_minipage_after_skip
1159         {
1160             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1161         }
1162     }
1163     \par\nopagebreak
1164     \addvspace { \l__enumext_minipage_left_skip }
1165 }

```

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

11.19.2 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`.

```

1166 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1167 {
1168   \skip_zero_new:N \l__enumext_minipage_after_skip
1169   \skip_zero_new:N \l__enumext_minipage_left_skip
1170   \skip_zero_new:N \l__enumext_minipage_right_skip
1171   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1172   {
1173     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1174     {
1175       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1176       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1177       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1178       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1179       {
1180         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1181       }
1182     }
1183     {
1184       \skip_set:Nn \l__enumext_minipage_left_skip
1185       {
1186         \skip_use:N \l__enumext_topsep_v_skip
1187       }
1188       \skip_set:Nn \l__enumext_minipage_right_skip
1189       {
1190         0.705\box_dp:N \strutbox
1191       }
1192       \skip_set:Nn \l__enumext_minipage_after_skip
1193       {
1194         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1195       }
1196     }
1197   }
1198   {
1199     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1200     {
1201       \skip_set:Nn \l__enumext_minipage_left_skip
1202       {
1203         0.5\box_dp:N \strutbox
1204         + \l__enumext_partopsep_v_skip
1205       }
1206       \skip_set:Nn \l__enumext_minipage_right_skip
1207       {
1208         \l__enumext_partopsep_v_skip
1209       }
1210       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1211     }
1212     {
1213       \skip_set:Nn \l__enumext_minipage_left_skip
1214       {
1215         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1216       }
1217       \skip_set:Nn \l__enumext_minipage_right_skip
1218       {
1219         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1220       }
1221       \skip_set:Nn \l__enumext_minipage_after_skip
1222       {
1223         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1224       }
1225     }
1226   }
1227 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`.)

`__enumext_keyans_mini_addvspace:`

The function `__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `__enumext_mini_env*` environment in `keyans`, taking into account whether $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ is in

(*horizontal mode*) or (*vertical mode*). For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in `enumext`.

```

1228 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1229 {
1230   \__enumext_keyans_mini_set_vskip:
1231   \mode_if_vertical:T
1232   {
1233     \skip_add:Nn \l__enumext_minipage_left_skip
1234     {
1235       \l__enumext_partopsep_v_skip
1236     }
1237     \skip_add:Nn \l__enumext_minipage_after_skip
1238     {
1239       \l__enumext_partopsep_v_skip
1240     }
1241   }
1242   \par\nopagebreak
1243   \addvspace { \l__enumext_minipage_left_skip }
1244 }

```

(End of definition for `__enumext_keyans_mini_addvspace:.`)

11.19.3 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*`.

```

1245 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1246 {
1247   \skip_zero_new:N \l__enumext_minipage_left_skip
1248   \skip_gzero_new:N \g__enumext_minipage_right_skip
1249   \skip_gzero_new:N \g__enumext_minipage_after_skip
1250   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1251   {
1252     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1253     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1254   }
1255   {
1256     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1257     \skip_gset:Nn \g__enumext_minipage_right_skip
1258     {
1259       \l__enumext_topsep_vii_skip
1260     }
1261     \skip_gset:Nn \g__enumext_minipage_after_skip
1262     {
1263       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1264     }
1265   }
1266 }
1267 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1268 {
1269   \skip_zero_new:N \l__enumext_minipage_after_skip
1270   \skip_zero_new:N \l__enumext_minipage_left_skip
1271   \skip_zero_new:N \l__enumext_minipage_right_skip
1272   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1273   {
1274     \skip_set:Nn \l__enumext_minipage_left_skip
1275     {
1276       0.5\box_dp:N \strutbox
1277     }
1278     \skip_set:Nn \l__enumext_minipage_right_skip
1279     {
1280       \l__enumext_partopsep_viii_skip
1281     }
1282     \skip_set:Nn \l__enumext_minipage_after_skip
1283     {
1284       1.6\box_dp:N \strutbox
1285     }
1286   }
1287 }

```

```

1288     \skip_set:Nn \l__enumext_minipage_left_skip
1289     {
1290         0.5875\box_dp:N \strutbox
1291     }
1292     \skip_set:Nn \l__enumext_minipage_right_skip
1293     {
1294         \l__enumext_topsep_viii_skip
1295     }
1296     \skip_set:Nn \l__enumext_minipage_after_skip
1297     {
1298         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1299     }
1300 }
1301 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

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

```

1302 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1303 {
1304     \__enumext_mini_set_vskip_vii:
1305     \par\nopagebreak
1306     \addvspace { \l__enumext_minipage_left_skip }
1307 }
1308 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1309 {
1310     \__enumext_mini_set_vskip_viii:
1311     \par\nopagebreak
1312     \addvspace { \l__enumext_minipage_left_skip }
1313 }

```

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

11.19.4 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 \TeX justification is maintained in the `__enumext_mini_env*` on the “right side”.

`\miniright`

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

```

1314 \NewDocumentCommand \miniright { s }
1315 {
1316     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1317     {
1318         \msg_error:nnn { enumext } { wrong-miniright-place }
1319     }
1320     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1321     {
1322         \msg_error:nnn { enumext } { wrong-miniright-place }
1323     }
1324     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1325     {
1326         \__enumext_keyans_mini_right_cmd:n {#1}
1327     }
1328     { \__enumext_mini_right_cmd:n {#1} }
1329 }

```

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

`__enumext_mini_right_cmd:n`

The function `__enumext_mini_right_cmd:n` takes as argument the *starred* ‘*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\l__enumext_minipage_right_X_dim`, if so we close the `\multicols` environment with the `__enumext_mini_env*` environment on the “left side”, then we open the `__enumext_mini_env*` environment on

the “*right side*”, apply our adjusted “*vertical spaces*”, followed by adding the `\centering` command when the starred argument ‘***’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1330 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1331 {
1332   \dim_compare:nNnTF
1333   { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1334   {
1335     \__enumext_multicols_stop:
1336     \end{__enumext_mini_env*}
1337     \hfill
1338     \begin{__enumext_mini_env*}
1339     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1340     \par\addvspace { \l__enumext_minipage_right_skip }
1341     \bool_if:nF {#1}
1342     {
1343       \centering
1344     }
1345     \int_gzero:N \g__enumext_minipage_stat_int
1346   }
1347   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1348 }

```

(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.

```

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

```

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

11.20 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* 1366 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1367 {
below* 1368   \keys_define:nn { enumext / #1 }
1369   {
1370     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1371     above .value_required:n = true,
1372     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1373               \keys_set:nn { enumext / #1 } { above = {##1} },
1374     above* .value_required:n = true,
1375     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1376     below .value_required:n = true,
1377     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1378               \keys_set:nn { enumext / #1 } { below = {##1} },
1379     below* .value_required:n = true,

```



```

1380     }
1381 }
1382 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

11.20.1 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.

```

1383 \cs_new_protected:Nn \__enumext_vspace_above:
1384 {
1385     \skip_if_eq:nnF
1386     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1387     {
1388         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1389         {
1390             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1391         }
1392         {
1393             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1394         }
1395     }
1396 }

```

(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.

```

1397 \cs_new_protected:Nn \__enumext_vspace_below:
1398 {
1399     \skip_if_eq:nnF
1400     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1401     {
1402         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1403         {
1404             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1405         }
1406         {
1407             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1408         }
1409     }
1410 }

```

(End of definition for `__enumext_vspace_below:`.)

11.20.2 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.

```

1411 \cs_new_protected:Nn \__enumext_vspace_above_v:
1412 {
1413     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1414     {
1415         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1416         {
1417             \vspace*{ \l__enumext_vspace_above_v_skip }
1418         }
1419         { \vspace { \l__enumext_vspace_above_v_skip } }
1420     }
1421 }

```

(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.

```

1422 \cs_new_protected:Nn \__enumext_vspace_below_v:
1423 {
1424     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1425     {
1426         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1427         {
1428             \vspace*{ \l__enumext_vspace_below_v_skip }

```

```

1429     }
1430     { \vspace { \l__enumext_vspace_below_v_skip } }
1431   }
1432 }

```

(End of definition for `__enumext_vspace_below_v:`)

11.20.3 Functions for above and below keys in `enumext*` `keyans*`

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:`

```

1433 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1434 {
1435   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1436   {
1437     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1438     {
1439       \vspace*{ \l__enumext_vspace_above_vii_skip }
1440     }
1441     { \vspace { \l__enumext_vspace_above_vii_skip } }
1442   }
1443 }
1444 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1445 {
1446   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1447   {
1448     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1449     {
1450       \vspace*{ \l__enumext_vspace_above_viii_skip }
1451     }
1452     { \vspace { \l__enumext_vspace_above_viii_skip } }
1453   }
1454 }

```

(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.

`__enumext_vspace_below_viii:`

```

1455 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1456 {
1457   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1458   {
1459     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1460     {
1461       \vspace*{ \l__enumext_vspace_below_vii_skip }
1462     }
1463     { \vspace { \l__enumext_vspace_below_vii_skip } }
1464   }
1465 }
1466 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1467 {
1468   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1469   {
1470     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1471     {
1472       \vspace*{ \l__enumext_vspace_below_viii_skip }
1473     }
1474     { \vspace { \l__enumext_vspace_below_viii_skip } }
1475   }
1476 }

```

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

11.21 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* 1477 \cs_set_protected:Npn \__enumext_tmp:n #1
1478 {
1479   \keys_define:nn { enumext / #1 }
1480   {
1481     series .str_set:N = \__enumext_series_str,
1482     series .value_required:n = true,
1483     resume .code:n = \__enumext_resume_series:n {##1},
1484     resume* .code:n = \__enumext_resume_starred:,
1485     resume* .value_forbidden:n = true,
1486   }
1487 }
1488 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.21.1 Internal functions for series key

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

```

1489 \cs_new:Npn \__enumext_filter_series:n #1
1490 {
1491   \use:e
1492   {
1493     \keyval_parse:NNn
1494     \__enumext_filter_series_key:n
1495     \__enumext_filter_series_pair:nn {#1}
1496   }
1497 }

```

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.

```

1498 \cs_new:Npn \__enumext_filter_series_key:n #1
1499 {
1500   \str_case:nnF {#1}
1501   {
1502     { resume } {}
1503     { resume* } {}
1504   }
1505   { , { \exp_not:n {#1} } }
1506 }

```

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.

```

1507 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1508 {
1509   \str_case:nnF {#1}
1510   {
1511     { series } {}
1512     { resume } {}
1513     { start } {}
1514     { save-ans } {}
1515     { save-key } {}
1516   }
1517   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1518 }

```

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

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

```

1519 \cs_new_protected:Npn \__enumext_parse_series:n #1
1520 {

```

```

1521 \str_if_empty:NTF \l__enumext_series_str
1522 {
1523     \bool_if:NF \l__enumext_resume_active_bool
1524     {
1525         \__enumext_resume_last:n {#1}
1526     }
1527 }
1528 {
1529     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1530     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1531     { \__enumext_filter_series:n {#1} }
1532     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1533     {
1534         \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1535     }
1536 }
1537 }

```

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.

```

1538 \cs_new_protected:Npn \__enumext_resume_last:n #1
1539 {
1540     \bool_if:NT \l__enumext_standar_first_bool
1541     {
1542         \tl_gclear:N \g__enumext_standar_series_tl
1543         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1544     }
1545     \bool_if:NT \l__enumext_starred_first_bool
1546     {
1547         \tl_gclear:N \g__enumext_starred_series_tl
1548         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1549     }
1550 }

```

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

11.21.2 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 (§11.32) and the `enumext*` environment definition (§11.35).

```

1551 \cs_new_protected:Npn \__enumext_resume_save_counter:
1552 {
1553     \bool_if:NT \g__enumext_standar_bool
1554     {
1555         \tl_if_empty:NF \l__enumext_series_str
1556         {
1557             \int_gset_eq:cN
1558             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1559         }
1560         \tl_if_empty:NTF \l__enumext_resume_name_tl
1561         {
1562             \str_if_empty:NF \l__enumext_series_str
1563             {
1564                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1565             }
1566         }
1567         {
1568             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1569             {
1570                 \int_gset_eq:cN

```

```

1571         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1572     }
1573 }
1574 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1575 {
1576     \int_gset_eq:cN
1577     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1578 }
1579 }
1580 \bool_if:NT \g__enumext_starred_bool
1581 {
1582     \tl_if_empty:NF \l__enumext_series_str
1583     {
1584         \int_gset_eq:cN
1585         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1586     }
1587     \tl_if_empty:NTF \l__enumext_resume_name_tl
1588     {
1589         \str_if_empty:NT \l__enumext_series_str
1590         {
1591             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1592         }
1593     }
1594     {
1595         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1596         {
1597             \int_gset_eq:cN
1598             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1599         }
1600     }
1601     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1602     {
1603         \int_gset_eq:cN
1604         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1605     }
1606 }
1607 }

```

(End of definition for __enumext_resume_save_counter:.)

11.21.3 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.

```

1608 \cs_new_protected:Npn \__enumext_resume_series:n #1
1609 {
1610     \tl_if_empty:nTF {#1}
1611     {
1612         \__enumext_resume_counter:n { }
1613     }
1614     {
1615         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1616         {
1617             \__enumext_resume_counter:n {#1}
1618             \bool_if:NT \g__enumext_standar_bool
1619             {
1620                 \keys_set:nv { enumext / level-1 }
1621                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1622             }
1623             \bool_if:NT \g__enumext_starred_bool
1624             {
1625                 \keys_set:nv { enumext / enumext* }
1626                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1627             }
1628         }
1629     }
1630     \bool_if:NT \g__enumext_standar_bool

```

```

1631         {
1632             \msg_error:nnn { enumext } { unknown-series } {#1}
1633         }
1634         \bool_if:NT \g__enumext_starred_bool
1635         {
1636             \msg_error:nnn { enumext } { unknown-series } {#1}
1637         }
1638     }
1639 }
1640 }

```

(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 $\langle \text{series name} \rangle$. 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=\langle \text{series name} \rangle` 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`.

```

1641 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1642 {
1643     \bool_set_true:N \l__enumext_resume_active_bool
1644     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1645     \tl_if_empty:NTF \l__enumext_resume_name_tl
1646     {
1647         \__enumext_resume_counter:
1648     }
1649     {
1650         \__enumext_resume_counter_series:
1651     }
1652     \__enumext_resume_counter_save_ans:
1653 }

```

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.

```

1654 \cs_new_protected:Nn \__enumext_resume_counter:
1655 {
1656     \bool_if:NT \g__enumext_standar_bool
1657     {
1658         \int_gincr:N \g__enumext_resume_int
1659         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1660     }
1661     \bool_if:NT \g__enumext_starred_bool
1662     {
1663         \int_gincr:N \g__enumext_resume_vii_int
1664         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1665     }
1666 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume=\langle \text{series name} \rangle` 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.

```

1667 \cs_new_protected:Nn \__enumext_resume_counter_series:
1668 {
1669     \bool_if:NT \g__enumext_standar_bool
1670     {
1671         \int_set:Nn \l__enumext_start_i_int
1672         {
1673             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1674         }
1675     }
1676     \bool_if:NT \g__enumext_starred_bool
1677     {
1678         \int_set:Nn \l__enumext_start_vii_int
1679         {
1680             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1681         }
1682     }
1683 }

```

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.

```

1684 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1685 {
1686   \bool_lazy_and:nnT
1687     { \bool_if_p:N \__enumext_standar_first_bool }
1688     { \bool_if_p:N \__enumext_store_active_bool }
1689     {
1690       \int_set:Nn \__enumext_start_i_int
1691       {
1692         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1693       }
1694     }
1695   \bool_lazy_and:nnT
1696     { \bool_if_p:N \__enumext_starred_first_bool }
1697     { \bool_if_p:N \__enumext_store_active_bool }
1698     {
1699       \int_set:Nn \__enumext_start_vii_int
1700       {
1701         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1702       }
1703     }
1704 }

```

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

11.21.4 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.

```

1705 \cs_new_protected:Nn \__enumext_resume_starred:
1706 {
1707   \bool_if:NT \g__enumext_standar_bool
1708   {
1709     \tl_if_empty:NF \g__enumext_standar_series_tl
1710     {
1711       \__enumext_resume_counter:n { }
1712       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1713     }
1714   }
1715   \bool_if:NT \g__enumext_starred_bool
1716   {
1717     \tl_if_empty:NF \g__enumext_starred_series_tl
1718     {
1719       \__enumext_resume_counter:n { }
1720       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1721     }
1722   }
1723 }

```

(End of definition for `__enumext_resume_starred:`.)

11.22 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.

11.22.1 Setting `save-ans` key

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

```

1724 \cs_set_protected:Npn \__enumext_tmp:n #1
1725 {
1726   \keys_define:nn { enumext / #1 }
1727   {
1728     save-ans .code:n = \__enumext_storing_set:n {##1},
1729     save-ans .value_required:n = true,
1730   }
1731 }
1732 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

11.22.2 Internal functions for save-ans key

```

\__enumext_start_save_ans_msg:
\__enumext_stop_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:`.

```

1733 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1734 {
1735     \msg_term:nnVV { enumext } { save-ans-log }
1736     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1737 }
1738 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1739 {
1740     \msg_term:nnVV { enumext } { save-ans-log-hook }
1741     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1742 }

```

(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.

```

1743 \cs_new_protected:Npn \__enumext_storing_set:n #1
1744 {
1745     \tl_set:Nx \l__enumext_store_name_tl {#1}
1746     \tl_if_empty:NTF \l__enumext_store_name_tl
1747     {
1748         \bool_lazy_or:nnT
1749         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1750         {
1751             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1752         }
1753     }
1754     {
1755         \bool_lazy_or:nnT
1756         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1757         {
1758             \__enumext_start_save_ans_msg:
1759             \__enumext_storing_exec:
1760         }
1761     }
1762 }

```

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. 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*`.

```

1763 \cs_new_protected:Nn \__enumext_storing_exec:
1764 {
1765     \bool_set_true:N \l__enumext_store_active_bool
1766     \bool_set_true:N \l__enumext_check_answers_bool
1767     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1768     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1769     {
1770         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1771         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1772     }
1773     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1774     {
1775         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1776         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1777     }
1778     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1779     {
1780         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl

```

```

1781         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1782     }
1783 }

```

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

11.22.3 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.

11.22.4 Setting check-ans and no-store keys

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

```

check-ans no-store
1784 \cs_set_protected:Npn \__enumext_tmp:n #1
1785 {
1786     \keys_define:nn { enumext / #1 }
1787     {
1788         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1789         check-ans .initial:n = false,
1790         check-ans .value_required:n = true,
1791         no-store .code:n = {
1792             \bool_set_false:N \l__enumext_check_answers_bool
1793             \bool_set_false:N \l__enumext_check_ans_key_bool
1794         },
1795         no-store .value_forbidden:n = true,
1796     }
1797 }
1798 \clist_map_inline:nn
1799 {
1800     level-1, level-2, level-3, level-4, enumext*
1801 }
1802 { \__enumext_tmp:n {#1} }

```

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

11.22.5 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.

```

1803 \cs_new_protected:Nn \__enumext_check_ans_active:
1804 {
1805     \tl_if_empty:NF \l__enumext_store_name_tl

```

```

1806     {
1807         \bool_if:NT \l__enumext_check_answers_bool
1808         {
1809             \__enumext_check_ans_level:
1810         }
1811     }
1812 }

```

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.

```

1813 \cs_new_protected:Nn \__enumext_check_ans_level:
1814 {
1815     \int_case:nn { \l__enumext_level_int }
1816     {
1817         { 1 }{
1818             \bool_lazy_all:nT
1819             {
1820                 { \bool_if_p:N \g__enumext_starred_bool }
1821                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1822             }
1823             {
1824                 \int_gdecr:N \g__enumext_item_number_int
1825             }
1826         }
1827         { 2 }{
1828             \int_gdecr:N \g__enumext_item_number_int
1829         }
1830         { 3 }{
1831             \int_gdecr:N \g__enumext_item_number_int
1832         }
1833         { 4 }{
1834             \int_gdecr:N \g__enumext_item_number_int
1835         }
1836     }

```

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.

```

1837     \int_case:nn { \l__enumext_level_h_int }
1838     {
1839         { 1 }{
1840             \bool_lazy_all:nT
1841             {
1842                 { \bool_if_p:N \g__enumext_standar_bool }
1843                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1844             }
1845             {
1846                 \int_gdecr:N \g__enumext_item_number_int
1847             }
1848         }
1849     }
1850 }

```

(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.

```

1851 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1852 {
1853     \bool_lazy_and:nnT
1854     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1855     { \bool_if_p:N \g__enumext_standar_bool }
1856     {
1857         \bool_gset_true:N \g__enumext_check_ans_key_bool
1858     }
1859     \bool_lazy_and:nnT
1860     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1861     { \bool_if_p:N \g__enumext_starred_bool }
1862     {

```

```

1863         \bool_gset_true:N \g__enumext_check_ans_key_bool
1864     }
1865 }

```

(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:`.

```

1866 \cs_new_protected:Nn \__enumext_item_answer_diff:
1867 {
1868     \int_gset:Nn \g__enumext_item_answer_diff_int
1869     {
1870         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1871     }
1872 }

```

(End of definition for __enumext_item_answer_diff:.)

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

```

1873 \cs_new_protected:Nn \__enumext_check_ans_show:
1874 {
1875     \int_case:nn { \g__enumext_item_answer_diff_int }
1876     {
1877         { -1 } { \__enumext_check_ans_msg_less: }
1878         { 0 } { \__enumext_check_ans_msg_same_ok: }
1879         { 1 } { \__enumext_check_ans_msg_greater: }
1880     }
1881 }
1882 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1883 {
1884     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1885     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1886 }
1887 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1888 {
1889     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1890     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1891 }
1892 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1893 {
1894     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1895     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1896 }

```

(End of definition for __enumext_check_ans_show: and others.)

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

```

1897 \cs_new_protected:Nn \__enumext_check_ans_log:
1898 {
1899     \int_case:nn { \g__enumext_item_answer_diff_int }
1900     {
1901         { -1 } { \__enumext_check_ans_log_msg_less: }
1902         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1903         { 1 } { \__enumext_check_ans_log_msg_greater: }
1904     }
1905 }
1906 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1907 {
1908     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1909     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1910 }
1911 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:

```

```

1912 {
1913   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1914   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1915 }
1916 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1917 {
1918   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1919   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1920 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.22.6 Writing .log and executing the check-ans key

__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 will finally execute the function __enumext_reset_global_vars: returning the used variables to their original state. As this function is passed to the function __enumext_after_env:nn for the environments `enumext` and `enumext*` we must make sure that we are not nested at any level.

```

1921 \cs_new_protected:Nn \__enumext_execute_after_env:
1922 {
1923   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1924   {
1925     \tl_if_empty:NF \g__enumext_store_name_tl
1926     {
1927       \__enumext_stop_save_ans_msg:
1928       \__enumext_item_answer_diff:
1929       \__enumext_log_global_vars:
1930       \__enumext_log_answer_vars:
1931       \bool_if:NTF \g__enumext_check_ans_key_bool
1932       {
1933         \__enumext_check_ans_show:
1934       }
1935       { \__enumext_check_ans_log: }
1936     }
1937     \__enumext_reset_global_vars:
1938   }
1939 }

```

(End of definition for __enumext_execute_after_env:.)

11.22.7 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.

```

1940 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1941 {
1942   \int_compare:nNnT
1943   { \g__enumext_check_starred_cmd_int } = { 0 }
1944   {
1945     \msg_warning:nnnV
1946     { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1947   }
1948   \int_compare:nNnT
1949   { \g__enumext_check_starred_cmd_int } > { 1 }
1950   {
1951     \msg_warning:nnnV
1952     { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1953   }
1954   \int_gzero:N \g__enumext_check_starred_cmd_int
1955   \tl_clear:N \l__enumext_check_start_line_env_tl
1956 }

```

(End of definition for __enumext_check_starred_cmd:n.)

11.23 Keys and functions associated with storage

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “*storage system*” and internal mechanism of “*label and ref*” only at the *first level* of `enumext` and `enumext*`.

```

1957 \cs_set_protected:Npn \__enumext_tmp:n #1
1958 {
1959   \keys_define:nn { enumext / #1 }
1960   {
1961     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1962     wrap-ans .initial:n = \fbox{##1},
1963     wrap-ans .value_required:n = true,
1964     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1965     wrap-opt .initial:n = [{##1}],
1966     wrap-opt .value_required:n = true,
1967     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1968     save-sep .initial:n = {, ~ },
1969     save-sep .value_required:n = true,
1970     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
1971     mark-ans .initial:n = \textasteriskcentered,
1972     mark-ans .value_required:n = true,
1973     mark-pos .choice:,
1974     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1975     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1976     mark-pos .initial:n = right,
1977     mark-pos .value_required:n = true,
1978     show-ans .bool_set:N = \__enumext_show_answer_bool,
1979     show-ans .initial:n = false,
1980     show-ans .value_required:n = true,
1981     show-pos .bool_set:N = \__enumext_show_position_bool,
1982     show-pos .initial:n = false,
1983     show-pos .value_required:n = true,
1984     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
1985     mark-ref .initial:n = \textasteriskcentered,
1986     mark-ref .value_required:n = true,
1987     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
1988     save-ref .initial:n = false,
1989     save-ref .value_required:n = true,
1990   }
1991 }
1992 \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`.

```

1993 \cs_set_protected:Npn \__enumext_tmp:n #1
1994 {
1995   \keys_define:nn { enumext / #1 }
1996   {
1997     mark-pos .choice:,
1998     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1999     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2000     mark-pos .initial:n = right,
2001     mark-pos .value_required:n = true,
2002     show-ans .bool_set:N = \__enumext_show_answer_bool,
2003     show-ans .initial:n = false,
2004     show-ans .value_required:n = true,
2005     show-pos .bool_set:N = \__enumext_show_position_bool,
2006     show-pos .initial:n = false,
2007     show-pos .value_required:n = true,
2008   }
2009 }
2010 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

11.23.1 Store optional arguments of the environments

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

```

  \__enumext_store_active_keys:n
  \__enumext_store_active_keys_vii:n

```

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

```

2011 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2012 {
2013   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2014   {
2015     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2016     \tl_set:ce
2017       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2018       { \__enumext_filter_save_key:n {#1} }
2019   }
2020 }
2021 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2022 {
2023   \bool_if:NF \l__enumext_store_save_key_vii_bool
2024   {
2025     \tl_clear:N \l__enumext_store_save_key_vii_tl
2026     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2027   }
2028 }

```

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

11.23.2 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.

```

2029 \cs_set_protected:Npn \__enumext_tmp:n #1
2030 {
2031   \keys_define:nn { enumext / enumext* }
2032   {
2033     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2034     save-key .value_required:n = true,
2035   }
2036   \keys_define:nn { enumext / #1 }
2037   {
2038     save-key .code:n = \__enumext_parse_save_key:n {##1},
2039     save-key .value_required:n = true,
2040   }
2041 }
2042 \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*`.

```

2043 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2044 {
2045   \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2046   \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2047   \tl_set:ce
2048     { \l__enumext_store_save_key_ \__enumext_level: _tl }
2049     { \__enumext_filter_save_key:n {#1} }
2050 }
2051 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2052 {
2053   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2054   \tl_clear:N \l__enumext_store_save_key_vii_tl
2055   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2056 }

```

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

11.23.3 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.

```

2057 \cs_new:Npn \__enumext_filter_save_key:n #1
2058 {
2059   \use:e
2060   {
2061     \keyval_parse:NNn
2062     \__enumext_filter_save_key_key:n
2063     \__enumext_filter_save_key_pair:nn {#1}
2064   }
2065 }
```

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.

```

2066 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2067 {
2068   \str_case:nnF {#1}
2069   {
2070     { resume } {} { resume* } {} { no-store } {}
2071   }
2072   { , { \exp_not:n {#1} } }
2073 }
```

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.

```

2074 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2075 {
2076   \str_case:nnF {#1}
2077   {
2078     { series } {} { resume } {} { save-ans } {}
2079     { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2080     { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2081     { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2082     { mini-right } {} { mini-right* } {}
2083   }
2084   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2085 }
```

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

11.23.4 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.

```

2086 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2087 {
2088   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2089   {
2090     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2091   }
2092   { #1 }
2093 }
2094 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

11.23.5 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.

```

2095 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2096 {
```

```

2097 \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2098 }
2099 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

11.23.6 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.

```

2100 \cs_new_protected:Nn \__enumext_store_level_open:
2101 {
2102   \bool_if:NT \l__enumext_check_answers_bool
2103   {
2104     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2105     {
2106       \__enumext_store_addto_seq:n
2107       {
2108         \item \begin{enumext}
2109       }
2110     }
2111     {
2112       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2113       {
2114         \item \begin{enumext} [
2115         ]
2116       }
2117       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2118       {
2119         ]
2120       }
2121       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2122     }
2123   }
2124 \cs_new_protected:Nn \__enumext_store_level_close:
2125 {
2126   \bool_if:NT \l__enumext_check_answers_bool
2127   {
2128     \__enumext_store_addto_seq:n { \end{enumext} }
2129   }
2130 }

```

(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.

```

2131 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2132 {
2133   \bool_if:NT \l__enumext_check_answers_bool
2134   {
2135     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2136     {
2137       \__enumext_store_addto_seq:n
2138       {
2139         \item \mode_leave_vertical:
2140         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2141         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2142       }
2143     }
2144     {
2145       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2146       {
2147         \item \mode_leave_vertical:
2148         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2149         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2150       ]
2151       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2152       {
2153         ]

```

```

2154         }
2155         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2156     }
2157 }
2158 }
2159 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2160 {
2161     \bool_if:NT \l__enumext_check_answers_bool
2162     {
2163         \__enumext_store_addto_seq:n { \end{enumext*} }
2164     }
2165 }

```

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

11.23.7 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`

```

2166 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2167 {
2168     \mode_leave_vertical:
2169     \skip_horizontal:n { -\dim_use:N #2 }
2170     \makebox[0pt][ r ]
2171     {
2172         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2173         {
2174             \tl_use:N \l__enumext_mark_answer_sym_tl
2175         }
2176     }
2177     \skip_horizontal:n { \dim_use:N #2 }
2178 }
2179 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

11.24 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*`.

```

2180 \keys_define:nn { enumext / anskey }
2181 {
2182     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2183     break-col .default:n = true,
2184     break-col .value_forbidden:n = true,
2185     item-join .int_set:N = \l__enumext_store_item_join_int,
2186     item-join .value_required:n = true,
2187     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2188     item-star .default:n = true,
2189     item-star .value_forbidden:n = true,
2190     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2191     item-sym* .value_required:n = true,
2192     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2193     item-pos* .value_required:n = true,
2194 }

```

🟢 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`.

```

2195 \NewDocumentCommand \anskey { o +m }
2196 {
2197   \__enumext_anskey_safe_outer:
2198   \group_begin:
2199     \bool_if:NT \l__enumext_check_answers_bool
2200     {
2201       \int_gincr:N \g__enumext_item_anskey_int
2202       \__enumext_anskey_safe_inner:n {#2}
2203       \__enumext_store_anskey_code:nn {#1} {#2}
2204     }
2205   \group_end:
2206 }

```

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

11.24.1 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.

```

2207 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2208 {
2209   \bool_if:NF \l__enumext_store_active_bool
2210   {
2211     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2212   }
2213   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2214   {
2215     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2216   }
2217   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2218   {
2219     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2220   }
2221 }

```

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.

```

2222 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2223 {
2224   \tl_if_empty:nT {#1}
2225   {
2226     \msg_error:nn { enumext } { anskey-empty-arg }
2227   }
2228   \int_incr:N \l__enumext_anskey_level_int
2229   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2230   {
2231     \msg_error:nn { enumext } { anskey-nested }
2232   }
2233 }

```

(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.

```

2234 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2235 {
2236   \__enumext_store_addto_prop:n {#2}
2237   \bool_if:NT \l__enumext_store_ref_key_bool
2238   {
2239     \__enumext_store_internal_ref:
2240   }
2241   \__enumext_store_anskey_show_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`.

```

2242   \tl_if_novalue:nF {#1}
2243   {

```

```

2244     \keys_set:nn { enumext / anskey } {#1}
2245   }
2246   \tl_clear:N \l__enumext_store_anskey_arg_tl
2247   \bool_lazy_and:nnT
2248     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2249     { \bool_not_p:n { \l__enumext_starred_bool } }
2250   {
2251     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2252   }
2253   \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`.

```

2254   \bool_lazy_and:nnT
2255     { \bool_not_p:n { \l__enumext_starred_bool } }
2256     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2257   {
2258     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2259       {
2260         ( \exp_not:V \l__enumext_store_item_join_int )
2261       }
2262   }

```

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`).

```

2263   \bool_if:NTF \l__enumext_store_item_star_bool
2264   {
2265     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2266     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2267     {
2268       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2269         {
2270           [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2271         }
2272     }
2273   \dim_compare:nT
2274   {
2275     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2276   }
2277   {
2278     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2279       {
2280         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2281       }
2282   }
2283   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2284 }
2285 {
2286   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2287 }

```

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`).

```

2288   \bool_lazy_and:nnT
2289     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2290     { \bool_if_p:N \l__enumext_hyperref_bool }
2291   {
2292     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2293       {
2294         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2295           { \exp_not:V \l__enumext_mark_ref_sym_tl }
2296       }
2297   }
2298   \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2299 }

```

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

`__enumext_store_internal_ref:`

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

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

```

2300 \cs_new_protected:Nn \__enumext_store_internal_ref:
2301 {
2302   \cs_set_protected:Npn \__enumext_tmp:n ##1
2303   {
2304     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2305     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2306     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2307     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2308   }
2309   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2310   \cs_set:Npn \__enumext_tmp:n ##1
2311   { . \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.

```

2312 \bool_lazy_all:nT
2313 {
2314   { \bool_if_p:N \g__enumext_starred_bool }
2315   { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
2316 }
2317 {
2318   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2319   { \tl_use:N \__enumext_label_copy_vii_tl }
2320 }
2321 \bool_lazy_all:nT
2322 {
2323   { \bool_if_p:N \l__enumext_standar_bool }
2324   { \bool_if_p:N \g__enumext_starred_bool }
2325   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2326 }
2327 {
2328   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2329   {
2330     \tl_use:N \__enumext_label_copy_vii_tl
2331     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2332   }
2333 }

```

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.

```

2334 \bool_lazy_all:nT
2335 {
2336   { \bool_if_p:N \l__enumext_standar_bool }
2337   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2338   { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_zero_int } }
2339   { \bool_not_p:n { \__enumext_starred_bool } }
2340 }
2341 {
2342   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2343   {
2344     \tl_use:N \__enumext_label_copy_i_tl
2345     \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
2346   }
2347 }
2348 \cs_set:Npn \__enumext_tmp:n ##1
2349 { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2350 \bool_lazy_all:nT
2351 {
2352   { \bool_if_p:N \l__enumext_standar_bool }
2353   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2354   { \bool_not_p:n { \g__enumext_starred_bool } }
2355   { \int_compare_p:nNn { \__enumext_level_h_int } > { \c_zero_int } }
2356 }
2357 {
2358   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2359   {
2360     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2361     . \tl_use:N \__enumext_label_copy_vii_tl
2362   }
2363 }

```

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

```

2364 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2365 {
2366   \l__enumext_store_name_tl \c_colon_str
2367   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2368 }

```

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

```

2369 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2370 {
2371   \__enumext_newlabel:nn
2372   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2373   { \l__enumext_newlabel_arg_two_tl }
2374 }
2375 \l__enumext_store_write_aux_file_tl
2376 }

```

(End of definition for `__enumext_store_internal_ref:.`)

`__enumext_store_anskey_show_wrap:n`

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

```

2377 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2378 {
2379   \par
2380   \bool_if:NT \l__enumext_starred_bool
2381   {
2382     \cs_set:Nn \__enumext_level: { vii }
2383   }
2384   \__enumext_print_keyans_box:cc
2385   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2386   { \l__enumext_labelsep_ \__enumext_level: _dim }
2387   \__enumext_anskey_wrapper:n { #1 }
2388 }

```

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

`__enumext_store_anskey_show_left:n`

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

```

2389 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2390 {
2391   \bool_if:NT \l__enumext_show_answer_bool
2392   {
2393     \__enumext_store_anskey_show_wrap:n { #1 }
2394   }
2395   \bool_if:NT \l__enumext_show_position_bool
2396   {
2397     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2398     {
2399       \group_begin:
2400       \exp_not:N \normalfont
2401       \exp_not:N \footnotesize [ \int_eval:n
2402       {
2403         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2404       }
2405       ]
2406       \group_end:
2407     }
2408     \__enumext_store_anskey_show_wrap:n { #1 }
2409   }
2410 }

```

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

11.25 The environment anskey*

anskey*

Only as a complement and demarcation for very extensive content, we will provide the environment version of the command `\anskey`.

```
\__enumext_anskey_env_safe_outer:
\__enumext_anskey_env_safe_inner:n

2411 \NewDocumentEnvironment{anskey*}{ o +b }
2412 {
2413   \__enumext_anskey_env_safe_outer:
2414   \bool_if:NT \l__enumext_check_answers_bool
2415   {
2416     \int_gincr:N \g__enumext_item_anskey_int
2417     \__enumext_store_anskey_safe_inner:n {#2}
2418     \__enumext_store_anskey_code:nn {#1} {#2}
2419   }
2420 }
2421 {}
```

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

```
2422 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
2423 {
2424   \bool_if:NF \l__enumext_store_active_bool
2425   {
2426     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2427   }
2428   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2429   {
2430     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2431   }
2432   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2433   {
2434     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2435   }
2436 }
```

The `__enumext_anskey_env_safe_inner:n` function will first check to see if the body is empty and then check to see if the environment is nested by returning the appropriate messages.

```
2437 \cs_new_protected:Npn \__enumext_anskey_env_safe_inner:n #1
2438 {
2439   \tl_if_empty:nT {#1}
2440   {
2441     \msg_error:nn { enumext } { anskey-empty-arg }
2442   }
2443   \int_incr:N \l__enumext_anskey_level_int
2444   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2445   {
2446     \msg_error:nn { enumext } { anskey-nested-env }
2447   }
2448 }
```

(End of definition for `anskey*`, `__enumext_anskey_env_safe_outer:`, and `__enumext_anskey_env_safe_inner:n`. This function is documented on page ??.)

11.26 Common functions for keyans, keyans* and keyanspic

11.26.1 Storing content in prop list

__enumext_keyans_addto_prop:n

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

```
2449 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2450 {
2451   \tl_clear:N \l__enumext_store_keyans_label_tl
2452   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2453   {
2454     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2455   }
2456   {
2457     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2458   }
2459   \tl_if_novalue:nF { #1 }
```

```

2460     {
2461         % Set save-sep
2462         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2463         {
2464             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2465         }
2466         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2467     }
2468     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2469 }

```

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

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

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

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

```

2470 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2471 {
2472     \bool_if:NT \l__enumext_store_ref_key_bool
2473     {
2474         \cs_set_protected:Npn \__enumext_tmp:n ##1
2475         {
2476             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2477             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2478             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2479             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2480         }
2481         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2482         \__enumext_keyans_store_ref_aux_i:
2483     }
2484 }

```

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

```

2485 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2486 {
2487     \bool_if:NT \g__enumext_starred_bool
2488     {
2489         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2490     }
2491     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2492     {
2493         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2494         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2495     }
2496     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2497     {
2498         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2499         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2500     }
2501     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2502     {
2503         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2504         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2505     }
2506     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2507     {
2508         \l__enumext_store_name_tl \c_colon_str
2509         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2510     }
2511     \__enumext_keyans_store_ref_aux_ii:
2512 }

```

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

```

2513 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2514 {
2515   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2516   {
2517     \__enumext_newlabel:nn
2518     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2519     { \l__enumext_newlabel_arg_two_tl }
2520   }
2521   \l__enumext_store_write_aux_file_tl
2522 }

```

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

11.26.3 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_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

2523 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2524 {
2525   \tl_clear:N \l__enumext_store_keyans_label_tl
2526   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2527   {
2528     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2529   }
2530   {
2531     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2532   }
2533   \tl_if_novalue:nF { #1 }
2534   {
2535     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2536     {
2537       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2538       {
2539         \l__enumext_store_keyans_item_opt_sep_tl
2540       }
2541     }
2542     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2543   }
2544   \__enumext_keyans_addto_seq_link:
2545 }

```

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_keyans_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.

```

2546 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2547 {
2548   \bool_lazy_and:nnT
2549   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2550   { \bool_if_p:N \l__enumext_hyperref_bool }
2551   {
2552     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2553     {
2554       \hfill \exp_not:N \hyperlink
2555       {
2556         \exp_not:V \l__enumext_newlabel_arg_one_tl
2557       }
2558       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2559     }
2560   }
2561   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2562   \bool_if:NT \l__enumext_check_answers_bool
2563   {
2564     \int_gincr:N \g__enumext_item_anskey_int

```

```

2565     }
2566 }

```

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

11.26.4 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:
2567 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2568 {
2569   \tl_if_novalue:nF { #1 }
2570   {
2571     \tl_set:Nx \__enumext_keyans_item_opt_tl { #1 }
2572   }
2573   \bool_if:NT \l__enumext_show_answer_bool
2574   {
2575     \__enumext_keyans_show_ans:
2576   }
2577   \bool_if:NT \l__enumext_show_position_bool
2578   {
2579     \__enumext_keyans_show_pos:
2580   }
2581 }
2582 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2583 {
2584   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2585   {
2586     \bool_lazy_or:nnT
2587     { \bool_if_p:N \l__enumext_show_answer_bool }
2588     { \bool_if_p:N \l__enumext_show_position_bool }
2589     {
2590       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2591     }
2592   }
2593 }
2594 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2595 {
2596   \tl_put_left:Nn \l__enumext_label_v_tl
2597   {
2598     \__enumext_print_keyans_box:NN
2599     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2600   }
2601 }
2602 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2603 {
2604   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2605   {
2606     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2607     {
2608       \group_begin:
2609       \exp_not:N \normalfont
2610       \exp_not:N \footnotesize [ \int_eval:n
2611       {
2612         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2613       }
2614       ]
2615       \group_end:
2616     }
2617   }
2618   {
2619     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2620     {
2621       \group_begin:
2622       \exp_not:N \normalfont
2623       \exp_not:N \footnotesize [ \int_eval:n
2624       {
2625         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2626       }

```

```

2627         ]
2628     \group_end:
2629 }
2630 }
2631 \tl_put_left:Nn \l__enumext_label_v_tl
2632 {
2633     \__enumext_print_keyans_box:NN
2634     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2635 }
2636 }

```

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

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

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

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

```

item-pos*
2637 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2638 {
2639     \keys_define:nn { enumext / #1 }
2640     {
2641         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2642         item-sym* .value_required:n = true,
2643         item-sym* .initial:n = {$\star$},
2644         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2645         item-pos* .value_required:n = true,
2646     }
2647 }
2648 \clist_map_inline:nn
2649 {
2650     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2651 }
2652 { \__enumext_tmp:nn #1 }

```

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

11.28 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](#).

`__enumext_renew_footnote:`

`__enumext_print_footnote:`

```

2653 \cs_new_protected:Nn \__enumext_footnotetext:nn
2654 {
2655     \footnotetext[#1]{#2}
2656 }
2657 \cs_new_protected:Nn \__enumext_renew_footnote:
2658 {
2659     \seq_gclear:N \g__enumext_footnote_arg_seq
2660     \seq_gclear:N \g__enumext_footnote_int_seq
2661     \RenewDocumentCommand \footnote { o +m }
2662     {
2663         \tl_if_novalue:nTF {##1}
2664         {
2665             \stepcounter{footnote}
2666             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2667         }
2668         {
2669             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2670         }
2671         \footnotemark [ \g__enumext_footnote_int ]
2672         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2673         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2674     }
2675 }
2676 \cs_new_protected:Nn \__enumext_print_footnote:
2677 {
2678     \seq_if_empty:NF \g__enumext_footnote_int_seq
2679     {
2680         \seq_map_pairwise_function:NNN
2681         \g__enumext_footnote_int_seq

```

```

2682         \g__enumext_footnote_arg_seq
2683         \__enumext_footnotetext:nn
2684     }
2685 }

```

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

11.29 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.

11.29.1 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:` (§11.30).

```

2686 \cs_new_protected:Npn \__enumext_default_item:n #1
2687 {
2688     \tl_if_novalue:nTF {#1}
2689     {
2690         \bool_if:NT \l__enumext_check_answers_bool
2691         {
2692             \int_gincr:N \g__enumext_item_number_int
2693         }
2694         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2695         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2696     }
2697     {
2698         \bool_set_eq:cc
2699         { \l__enumext_wrap_label_ \__enumext_level: _bool }
2700         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2701         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2702     }
2703 }

```

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

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

```

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

```

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label:` (§11.30).

```

2704 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2705 {
2706     \tl_if_novalue:nF {#1}
2707     {
2708         \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2709     }
2710     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2711     \tl_if_novalue:nTF {#2}

```

```

2712     {
2713       \dim_set_eq:cc
2714       { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2715       { \__enumext_labelsep_ \__enumext_level: _dim }
2716     }
2717     {
2718       \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2719     }
2720     \bool_if:NT \__enumext_check_answers_bool
2721     {
2722       \int_gincr:N \g__enumext_item_number_int
2723     }
2724     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2725     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2726   }

```

(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 (§11.31.2).

```

2727 \cs_new_protected:Nn \__enumext_redefine_item:
2728 {
2729   \RenewDocumentCommand \item { s o o }
2730   {
2731     \bool_if:nTF {##1}
2732     {
2733       \__enumext_starred_item:nn {##2} {##3}
2734     }
2735     { \__enumext_default_item:n {##2} }
2736   }
2737 }

```

(End of definition for __enumext_redefine_item:.)

11.29.2 The `\item` command in keyans

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

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

```

2738 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2739 {
2740   \tl_if_novalue:nTF { #1 }
2741   {
2742     \bool_set_true:N \__enumext_wrap_label_v_bool
2743     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2744   }
2745   {
2746     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2747     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2748   }
2749 }

```

(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 $\langle label \rangle$.

```

2750 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2751 {
2752   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2753   \__enumext_keyans_show_left:n { #1 }
2754   \bool_set_true:N \__enumext_wrap_label_v_bool
2755   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item

```


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

```

2756 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2757 \__enumext_keyans_addto_prop:n { #1 }
2758 \__enumext_keyans_store_ref:
2759 \__enumext_keyans_addto_seq:n { #1 }
2760 \int_gincr:N \g__enumext_check_starred_cmd_int
2761 }

```

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

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.31.2).

```

2762 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2763 {
2764   \RenewDocumentCommand \item { s o }
2765   {
2766     \bool_if:nTF {##1}
2767     {
2768       \peek_remove_spaces:n
2769       {
2770         \__enumext_keyans_starred_item:n {##2}
2771       }
2772     }
2773     {
2774       \__enumext_keyans_default_item:n {##2}
2775     }
2776   }
2777 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 13.)

11.30 Redefining `\makeLabel` command

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

11.30.1 Redefining `\makeLabel` for `enumext`

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

```

2778 \cs_new_protected:Nn \__enumext_item_starred:
2779 {
2780   \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2781   {
2782     \mode_leave_vertical:
2783     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2784     \makebox[ 0pt ]{ r }{ \g__enumext_item_symbol_tl }
2785     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2786   }
2787 }

```

(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 (§11.31.2).

```

2788 \cs_new_protected:Nn \__enumext_make_label:
2789 {
2790   \RenewDocumentCommand \makeLabel { m }
2791   {
2792     \tl_use:c { \l__enumext_label_fill_left_ \l__enumext_level: _tl }
2793     \tl_use:c { \l__enumext_label_font_style_ \l__enumext_level: _tl }
2794     \bool_if:cTF { \l__enumext_wrap_label_ \l__enumext_level: _bool }
2795     {
2796       \__enumext_item_starred:

```

```

2797         \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2798     }
2799     { ##1 }
2800     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2801     \tl_gclear:N \g__enumext_item_symbol_tl
2802 }
2803 }

```

(End of definition for `__enumext_make_label:`)

11.30.2 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 (§11.31.2).

```

2804 \cs_new_protected:Nn \__enumext_keyans_make_label:
2805 {
2806     \RenewDocumentCommand \makelabel { m }
2807     {
2808         \tl_use:N \l__enumext_label_fill_left_v_tl
2809         \tl_use:N \l__enumext_label_font_style_v_tl
2810         \bool_if:NTF \l__enumext_wrap_label_v_bool
2811         {
2812             \__enumext_wrapper_label_v:n { ##1 }
2813         }
2814         { ##1 }
2815         \tl_use:N \l__enumext_label_fill_right_v_tl
2816     }
2817 }

```

(End of definition for `__enumext_keyans_make_label:`)

11.31 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`.

11.31.1 Calculation of `\leftmargin` and `\itemindent`

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

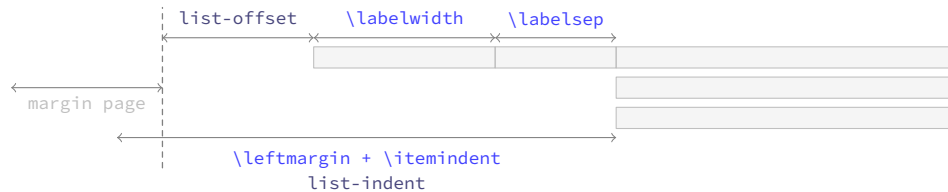


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

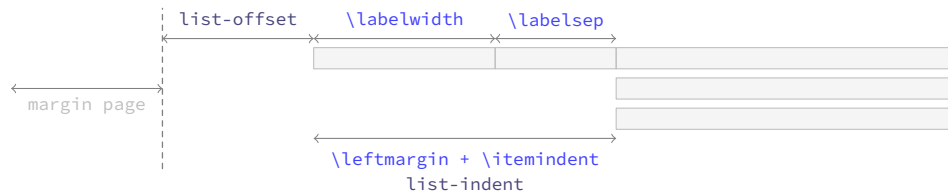


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

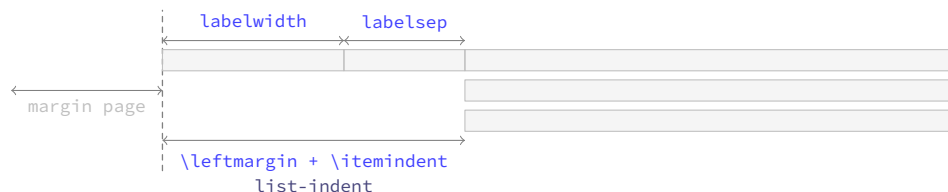


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

```

#1: \__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 (§11.31.2).

```

2818 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2819 {
2820   \dim_compare:nNt { #1 } < { \c_zero_dim }
2821   {
2822     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2823     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2824   }
2825   \dim_compare:nNt { #2 } < { \c_zero_dim }
2826   {
2827     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2828     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2829   }

```

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

```

2830   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2831   \dim_compare:nNtF { #4 } < { \c_zero_dim }
2832   {
2833     \dim_set:Nn #6 { #1 + #2 - #4 }
2834     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2835   }
2836   {
2837     \dim_compare:nNt { #4 } = { #1 + #2 }
2838     { \dim_set:Nn #6 { \c_zero_dim } }
2839     \dim_compare:nNt { #4 } < { #1 + #2 }
2840     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2841     \dim_compare:nNt { #4 } > { #1 + #2 }
2842     {
2843       \dim_set:Nn #6 { -#1 - #2 + #4 }
2844       \dim_set:Nn #6 { #6*-1 }
2845     }
2846     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2847   }
2848 }
2849 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

11.31.2 Setting second argument of the lists

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

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:

```

```

2850 \cs_set_protected:Npn \__enumext_tmp:n #1
2851 {
2852   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2853   {
2854     \__enumext_calc_hspace:ccccccc
2855     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2856     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2857     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2858     { \__enumext_leftmargin_tmp_#1_bool }
2859     \clist_map_inline:nn
2860     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2861     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2862     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2863     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2864     \usecounter { enumX#1 }
2865     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```

```

2866 \str_if_eq:nnTF {#1} { v }
2867 {
2868   \__enumext_keyans_redefine_item:
2869   \__enumext_keyans_make_label:
2870   \__enumext_keyans_ref:
2871   \__enumext_keyans_fake_item:
2872   \bool_if:cT { l__enumext_show_length_#1_bool }
2873   {
2874     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2875   }
2876 }
2877 {
2878   \__enumext_redefine_item:
2879   \__enumext_make_label:
2880   \__enumext_standar_ref:
2881   \__enumext_fake_item:
2882   \bool_if:cT { l__enumext_show_length_#1_bool }
2883   {
2884     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_#1 }
2885   }
2886 }
2887 }
2888 }
2889 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `0pt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `parindent` to make it set the value of `listparindent` and `parsep` to set the value of `\parskip` locally.

```

2890 \cs_set_protected:Npn \__enumext_tmp:n #1
2891 {
2892   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2893   {
2894     \__enumext_calc_hspace:ccccc
2895     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2896     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2897     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2898     { l__enumext_leftmargin_tmp_#1_bool }
2899     \clist_map_inline:nn
2900     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2901     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2902     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2903     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2904     \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
2905     \skip_zero:N \partopsep
2906     \usecounter { enumX#1 }
2907     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2908     \__enumext_starred_ref:
2909     \str_if_eq:nnTF {#1} { vii }
2910     {
2911       \__enumext_fake_item_vii:
2912       \bool_if:cT { l__enumext_show_length_vii_bool }
2913       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2914     }
2915     {
2916       \__enumext_fake_item_viii:
2917       \bool_if:cT { l__enumext_show_length_#1_bool }
2918       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2919     }
2920   }
2921 }
2922 \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:`.)

11.32 The environment `enumext`

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

```

2923 \NewDocumentEnvironment{enumext}{0}{}
2924 {

```

```

2925     \__enumext_safe_exec:
2926     \__enumext_parse_keys:n {#1}
2927     \__enumext_before_list:
2928     \__enumext_start_store_level:
2929     \__enumext_start_list:nn
2930     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2931     {
2932         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2933         \__enumext_before_keys_exec:
2934     }
2935     \__enumext_after_args_exec:
2936 }
2937 {
2938     \__enumext_stop_list:
2939     \__enumext_stop_store_level:
2940     \__enumext_after_list:
2941 }

```

(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:`.

```

2942 \cs_new_protected:Nn \__enumext_safe_exec:
2943 {
2944     \__enumext_is_not_nested:
2945     \int_incr:N \l__enumext_level_int
2946     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2947     { \msg_fatal:nn { enumext } { list-too-deep } }
2948     \bool_set_true:N \l__enumext_standar_bool
2949     \__enumext_is_on_first_level:
2950 }

```

(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`.

```

2951 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2952 {
2953     \tl_if_novalue:nF {#1}
2954     {
2955         \str_clear:N \l__enumext_series_str
2956         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2957         {
2958             \keys_set:nn { enumext / level-1 } {#1}
2959             \__enumext_parse_series:n {#1}
2960         }
2961         {
2962             \exp_args:Ne \keys_set:nn
2963             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2964         }
2965         \__enumext_store_active_keys:n {#1}
2966     }
2967 }

```

(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.

```

2968 \cs_new_protected:Nn \__enumext_start_store_level:
2969 {

```

```

2970 \bool_lazy_all:nT
2971 {
2972   { \bool_if_p:N \l__enumext_store_active_bool }
2973   { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2974   { \bool_not_p:n { \g__enumext_starred_bool } }
2975 }
2976 {
2977   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2978   {
2979     \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2980     \__enumext_store_level_open:
2981   }
2982 }
2983 \bool_lazy_all:nT
2984 {
2985   { \bool_if_p:N \l__enumext_store_active_bool }
2986   { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2987   { \bool_if_p:N \g__enumext_starred_bool }
2988 }
2989 {
2990   \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2991   {
2992     \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2993     \__enumext_store_level_open:
2994   }
2995 }
2996 }
2997 \cs_new_protected:Nn \__enumext_stop_store_level:
2998 {
2999   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3000   {
3001     \__enumext_store_level_close:
3002   }
3003 }

```

(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.

```

3004 \cs_new_protected:Nn \__enumext_before_list:
3005 {
3006   \__enumext_vspace_above:
3007   \__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.

```

3008 \__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.

```

3009 \dim_compare:nNnT
3010 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3011 {
3012   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3013   {
3014     \linewidth
3015     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3016     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3017   }

```

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

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

```

3018 \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }

```

```

3019         \int_gincr:N \g__enumext_minipage_stat_int
3020         \__enumext_mini_addvspace:
3021         \nointerlineskip\noindent
3022         \begin{__enumext_mini_env*}
3023         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3024     }
3025     \__enumext_multicols_start:
3026 }

```

(End of definition for __enumext_before_list:.)

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

```

3027 \cs_new_protected:Nn \__enumext_multicols_start:
3028 {
3029     \int_compare:nNt
3030     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3031     {
3032         \dim_compare:nNt
3033         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3034         {
3035             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3036             {
3037                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3038                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3039                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3040                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3041             }
3042         }
3043         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3044         \skip_zero:N \multicolsep
3045         \int_compare:nNt { \l__enumext_level_int } > { 1 }
3046         {
3047             \dim_zero:N \columnseprule
3048         }
3049     }

```

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.

```

3049         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3050         {
3051             \__enumext_multi_addvspace:
3052         }
3053         \raggedcolumns
3054         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3055     }
3056 }

```

(End of definition for __enumext_multicols_start:.)

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

```

3057 \cs_new_protected:Nn \__enumext_multicols_stop:
3058 {
3059     \int_compare:nNt
3060     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3061     {
3062         \end{multicols}
3063         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3064         {
3065             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3066         }
3067     }
3068 }

```

(End of definition for __enumext_multicols_stop:.)

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

```

3069 \cs_new_protected:Nn \__enumext_after_list:
3070 {
3071   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3072   {
3073     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3074     {
3075       \msg_warning:nn { enumext } { missing-miniright }
3076       \miniright
3077     }
3078     \int_gzero:N \g__enumext_minipage_stat_int
3079     \end{\__enumext_mini_env*}
3080     \par\addvspace { \l__enumext_minipage_after_skip }
3081   }
3082   { \__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`.

```

3083   \__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.

```

3084   \__enumext_after_stop_list:
3085   \__enumext_vspace_below:
3086   \bool_set_false:N \l__enumext_standar_bool
3087   \__enumext_resume_save_counter:
3088 }

```

(End of definition for `__enumext_after_list:`)

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

```

3089 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

11.33 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.

```

3090 \NewDocumentEnvironment{keyans}{0}{}
3091 {
3092   \__enumext_keyans_safe_exec:
3093   \__enumext_keyans_parse_keys:n {#1}
3094   \__enumext_before_list_v:
3095   \__enumext_start_list:nn
3096   { \tl_use:N \l__enumext_label_v_tl }
3097   {
3098     \__enumext_list_arg_two_v:
3099     \__enumext_before_keys_exec_v:
3100   }
3101   \__enumext_after_args_exec_v:
3102 }
3103 {
3104   \__enumext_check_starred_cmd:n { item }
3105   \__enumext_stop_list:
3106   \__enumext_after_list_v:
3107 }

```

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

`__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.

```

3108 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3109 {

```

```

3110 \bool_if:NF \l__enumext_store_active_bool
3111 {
3112   \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3113 }
3114 \int_incr:N \l__enumext_keyans_level_int
3115 \bool_set_true:N \l__enumext_keyans_env_bool
3116 \__enumext_keyans_save_start_line:
3117 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3118 \bool_set_false:N \l__enumext_store_active_bool
3119 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3120 {
3121   \msg_error:nn { enumext } { keyans-nested }
3122 }
3123 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3124 {
3125   \msg_error:nn { enumext } { keyans-wrong-level }
3126 }
3127 }

```

(End of definition for __enumext_keyans_safe_exec:.)

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

```

3128 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3129 {
3130   \keys_set:nn { enumext / keyans } { #1 }
3131 }

```

(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.

```

3132 \cs_new_protected:Nn \__enumext_before_list_v:
3133 {
3134   \__enumext_vspace_above_v:
3135   \__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.

```

3136 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3137 {
3138   \dim_set:Nn \l__enumext_minipage_left_v_dim
3139   {
3140     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3141   }

```

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

```

3142 \bool_set_true:N \l__enumext_minipage_active_v_bool
3143 \int_gincr:N \g__enumext_minipage_stat_int
3144 \__enumext_keyans_mini_addvspace:
3145 \nointerlineskip\noindent
3146 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3147 }

```

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

```

3148 \__enumext_keyans_multicols_start:
3149 }

```

(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.

```

3150 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3151 {
3152   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3153   {

```

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

```

3154     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3155     {
3156       \dim_set:Nn \__enumext_columns_sep_v_dim
3157       {
3158         (
3159           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3160         ) / \__enumext_columns_v_int
3161         - \__enumext_listoffset_v_dim
3162       }
3163     }
3164     \dim_set_eq:NN \columnsep \__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).

```

3165     \skip_zero:N \multicolsep
3166     \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.

```

3167     \bool_if:NF \__enumext_minipage_active_v_bool
3168     {
3169       \__enumext_keyans_multi_addvspace:
3170     }
3171     \raggedcolumns
3172     \begin{multicols}{\__enumext_columns_v_int }
3173   }
3174 }

```

(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 `__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

3175 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3176 {
3177   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3178   {
3179     \end{multicols}
3180     \bool_if:NF \__enumext_minipage_active_v_bool
3181     {
3182       \par\addvspace{ \__enumext_multicols_below_v_skip }
3183     }
3184   }
3185 }

```

(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 `__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 `__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3186 \cs_new_protected:Nn \__enumext_after_list_v:
3187 {
3188   \bool_if:NtF \__enumext_minipage_active_v_bool
3189   {
3190     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3191     {
3192       \msg_warning:nn { enumext } { missing-miniright }
3193       \miniright
3194     }
3195     \int_gzero:N \g__enumext_minipage_stat_int

```

```

3196         \end{__enumext_mini_env*}
3197         \par\addvspace{ \l__enumext_minipage_after_skip }
3198     }
3199     { __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.

```

3200     \bool_set_false:N \l__enumext_keyans_env_bool
3201     \__enumext_after_stop_list_v:
3202     \__enumext_vspace_below_v:
3203 }

```

(End of definition for `__enumext_after_list_v:`)

11.34 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.

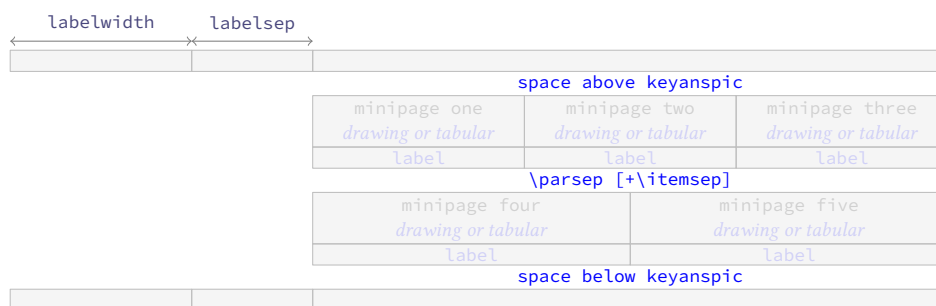


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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?](#).

11.34.1 The command `\anspic`

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

```

3204 \NewDocumentCommand \anspic { s o +m }
3205 {

```

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

```

3206     \bool_if:NF \l__enumext_store_active_bool
3207     {
3208         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3209     }
3210     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3211     {
3212         \msg_error:nn { enumext } { keyanspic-wrong-level }
3213     }
3214     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3215     {
3216         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3217     }

```

The three arguments are handled by the function `__enumext_keyans_anspic_code:nnn` and stored in the sequence `\l__enumext_keyans_pic_body_seq` which is processed by the `keyanspic` environment.

```

3218     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3219     {
3220         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3221     }
3222 }

```

(End of definition for `\anspic`. This function is documented on page 14.)

`__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.

```

3223 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3224 {
3225   \stepcounter { enumXvi }
3226   #3 \\\
3227   \bool_if:nT { #1 }
3228   {
3229     \__enumext_keyans_addto_prop:n { #2 }
3230     \__enumext_keyans_store_ref:
3231     \__enumext_keyans_addto_seq:n { #2 }
3232     \int_gincr:N \g__enumext_check_starred_cmd_int
3233     \bool_lazy_or:nnT
3234     { \bool_if_p:N \l__enumext_show_answer_bool }
3235     { \bool_if_p:N \l__enumext_show_position_bool }
3236     {
3237       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3238       \__enumext_keyans_show_left:n { #2 }
3239       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3240     }
3241   }
3242   \tl_use:N \l__enumext_label_font_style_v_tl
3243   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3244 }

```

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

11.34.2 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.

```

3245 \NewDocumentEnvironment{keyanspic}{o}
3246 {
3247   \__enumext_keyans_pic_safe_exec:
3248   \__enumext_start_list:nn
3249   { }
3250   {
3251     \__enumext_keyans_pic_arg_two:
3252   }

```

We apply the “adjusted” vertical spacing above the environment

```

3253   \vspace { \l__enumext_keyans_pic_above_skip }
3254 }

```

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.

```

3255 {
3256   \tl_if_novalue:nTF { #1 }
3257   {
3258     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3259   }
3260   { \__enumext_keyans_pic_do:n { #1 } }
3261   \__enumext_stop_list:
3262   \__enumext_check_starred_cmd:n { anspic }
3263   \setcounter { enumXvi } { 0 }
3264   \vspace { \l__enumext_topsep_v_skip }
3265   %\bool_set_false:N \l__enumext_store_active_bool
3266 }

```

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

`__enumext_keyans_pic_safe_exec:`

The function `__enumext_keyans_pic_safe_exec:` check nested and level position inside the `enumext` environment.

```

3267 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3268 {
3269   \int_incr:N \l__enumext_keyans_pic_level_int
3270   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3271   {
3272     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3273     }
3274     \__enumext_keyans_save_start_line:
3275 }

```

(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.

```

3276 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3277 {
3278     \dim_compare:nNnT { #1 } < { 0pt }
3279     { \skip_set:Nn #1 { -#1 } }
3280 }

```

(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.

```

3281 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3282 {

```

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.

```

3283     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3284     \__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.

```

3285     \skip_add:Nn \parsep { \itemsep }
3286     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3287     \dim_zero:N \labelwidth
3288     \dim_zero:N \listparindent
3289     \dim_zero:N \labelsep
3290     \skip_zero:N \partopsep
3291     \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.

```

3292     \__enumext_keyans_pic_skip_abs:N \parsep
3293     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3294     {
3295         \box_dp:N \strutbox
3296         + \l__enumext_topsep_v_skip
3297         - \parsep
3298     }
3299     \__enumext_item_std:w \scan_stop:
3300 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

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

```

3301 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3302 {
3303     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3304 }
3305 \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.

```

3306 \cs_new_protected:Npn \__enumext_keyans_pic_row:n
3307 {
3308     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3309     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3310     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }

```

```

3311 \int_step_inline:nnn
3312 { \l__enumext_keyans_pic_above_int + 1 }
3313 { \l__enumext_keyans_pic_below_int }
3314 {
3315   \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3316   \centering
3317   \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3318   \__enumext_endminipage:
3319 }
3320 \par
3321 }

```

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

11.35 The environment `enumext*`

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`).

11.35.1 Functions for item box width

`__enumext_starred_columns_set_vii:`

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

```

3322 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3323 {
3324   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3325   {
3326     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3327     {
3328       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3329       / \l__enumext_columns_vii_int
3330     }
3331   }
3332   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3333   \dim_set:Nn \l__enumext_item_width_vii_dim
3334   {
3335     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3336     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3337     - \l__enumext_labelsep_vii_dim
3338   }
3339   \dim_zero_new:N \itemwidth
3340 }

```

(End of definition for `__enumext_starred_columns_set_vii:`.)

`__enumext_starred_joined_item_vii:n`

The function `__enumext_starred_joined_item_vii:n` will set the *width* of the box in which the content passed to `\item(\langle number \rangle)` will be stored together with the value of `\itemwidth`.

```

3341 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3342 {
3343   \int_set:Nn \l__enumext_joined_item_vii_int { #1 }
3344   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3345   {
3346     \msg_warning:nnee { enumext } { item-joined }
3347     { \int_use:N \l__enumext_joined_item_vii_int }
3348     { \int_use:N \l__enumext_columns_vii_int }
3349     \int_set:Nn \l__enumext_joined_item_vii_int
3350     {
3351       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3352     }
3353   }
3354   \int_compare:nNnT
3355   { \l__enumext_joined_item_vii_int }
3356   >
3357   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```



```

3358     {
3359         \msg_warning:nnee { enumext } { item-joined-columns }
3360         { \int_use:N \l__enumext_joined_item_vii_int }
3361         {
3362             \int_eval:n
3363             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3364         }
3365         \int_set:Nn \l__enumext_joined_item_vii_int
3366         {
3367             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3368         }
3369     }

```

Only need if #1 » 1 (default are set before).

```

3370     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3371     {
3372         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3373         \int_decr:N \l__enumext_joined_item_aux_vii_int
3374         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3375         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3376         \dim_set:Nn \l__enumext_joined_width_vii_dim
3377         {
3378             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3379             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3380                 + \l__enumext_columns_sep_vii_dim
3381                 ) * \l__enumext_joined_item_aux_vii_int
3382         }
3383         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3384     }
3385     {
3386         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3387         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3388     }
3389 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n.)

`__enumext_start_mini_vii:` The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

3390 \cs_new_protected:Nn \__enumext_start_mini_vii:
3391 {
3392     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3393     {
3394         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3395         {
3396             \linewidth
3397             - \l__enumext_minipage_right_vii_dim
3398             - \l__enumext_minipage_hsep_vii_dim
3399         }
3400         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3401         \dim_gset_eq:NN
3402             \g__enumext_minipage_right_vii_dim
3403             \l__enumext_minipage_right_vii_dim
3404         \__enumext_mini_addvspace_vii:
3405         \nointerlineskip\noindent
3406         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3407     }
3408 }

```

(End of definition for __enumext_start_mini_vii:.)

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

```

3409 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3410 {
3411     \bool_if:NT \l__enumext_minipage_active_vii_bool

```

```

3412     {
3413         \end{__enumext_mini_env*}
3414         \hfill
3415         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3416     }
3417 }

```

Finally we execute code passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “right side”.

```

3418 \__enumext_after_env:nn {enumext*}
3419 {
3420     \bool_if:NT \g__enumext_minipage_active_vii_bool
3421     {
3422         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3423         \par\addvspace { \g__enumext_minipage_right_skip }
3424         \bool_if:NF \g__enumext_minipage_center_vii_bool
3425         {
3426             \centering
3427         }
3428         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3429         \end{__enumext_mini_env*}
3430         \par\addvspace{ \g__enumext_minipage_after_skip }
3431     }
3432     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3433     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3434     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3435     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3436     \bool_gset_false:N \g__enumext_starred_bool
3437 }

```

(End of definition for `__enumext_stop_mini_vii:.`)

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

```

3438 \NewDocumentEnvironment{enumext*}{ o }
3439 {
3440     \__enumext_safe_exec_vii:
3441     \__enumext_parse_keys_vii:n {#1}
3442     \__enumext_before_list_vii:
3443     \__enumext_start_store_level_vii:
3444     \__enumext_start_list:nn { }
3445     {
3446         \__enumext_list_arg_two_vii:
3447         \__enumext_before_keys_exec_vii:
3448     }
3449     \__enumext_starred_columns_set_vii:
3450     \item[] \scan_stop:
3451     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3452     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3453 }
3454 {
3455     \__enumext_stop_item_tmp_vii:
3456     \__enumext_remove_extra_parsep_vii:
3457     \__enumext_stop_list:
3458     \__enumext_stop_store_level_vii:
3459     \__enumext_after_list_vii:
3460 }

```

(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`.

```

3461 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3462 {
3463     \__enumext_is_not_nested:
3464     \int_incr:N \l__enumext_level_h_int
3465     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3466     {
3467         \msg_error:nn { enumext } { nested }
3468     }

```

```

3469     \bool_set_true:N \l__enumext_starred_bool
3470     \__enumext_is_on_first_level:
3471 }

```

(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*.

```

3472 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3473 {
3474     \tl_if_novalue:nF {#1}
3475     {
3476         \str_clear:N \l__enumext_series_str
3477         \keys_set:nn { enumext / enumext* } {#1}
3478         \__enumext_parse_series:n {#1}
3479         \__enumext_store_active_keys_vii:n {#1}
3480     }
3481 }

```

(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*.

```

3482 \cs_new_protected:Nn \__enumext_before_list_vii:
3483 {
3484     \__enumext_vspace_above_vii:
3485     \__enumext_check_ans_active:
3486     \__enumext_before_args_exec_vii:
3487     \__enumext_start_mini_vii:
3488 }

```

(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.

```

3489 \cs_new_protected:Nn \__enumext_after_list_vii:
3490 {
3491     \__enumext_stop_mini_vii:
3492     \__enumext_after_stop_list_vii:
3493     \__enumext_check_ans_key_hook:
3494     \__enumext_vspace_below_vii:
3495     \bool_set_false:N \l__enumext_starred_bool
3496     \__enumext_resume_save_counter:
3497 }

```

(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**.

```

3498 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3499 {
3500     \bool_if:NT \l__enumext_store_active_bool
3501     {
3502         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3503         {
3504             \__enumext_store_level_open_vii:
3505         }
3506     }
3507 }
3508 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3509 {
3510     \bool_if:NT \l__enumext_store_active_bool
3511     {

```

```

3512         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3513         {
3514             \__enumext_store_level_close_vii:
3515         }
3516     }
3517 }

```

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

11.35.2 The command \item in enumext*

__enumext_start_item_tmp_vii:

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

```

3518 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3519 {
3520     \__enumext_stop_item_tmp_vii:
3521     \int_incr:N \l__enumext_item_column_pos_vii_int
3522     \int_gincr:N \g__enumext_item_count_all_vii_int
3523     \__enumext_item_peek_args_vii:
3524 }

```

(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).

```

3525 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3526 {
3527     \peek_meaning:NTF (
3528         { \__enumext_joined_item_vii:w }
3529         { \__enumext_joined_item_vii:w (1) }
3530     }

```

(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.

```

3531 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3532 {
3533     \__enumext_starred_joined_item_vii:n {#1}
3534     \peek_meaning_remove:NTF *
3535     { \__enumext_starred_item_vii:w }
3536     { \__enumext_standar_item_vii:w }
3537 }

```

(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].

```

3538 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3539 {
3540     \bool_set_false:N \l__enumext_item_starred_vii_bool
3541     \peek_meaning:NTF [
3542         {
3543             \bool_set_eq:NN
3544             \l__enumext_wrap_label_vii_bool
3545             \l__enumext_wrap_label_opt_vii_bool
3546             \__enumext_start_item_vii:w
3547         }

```

```

3548     {
3549         \bool_set_true:N \__enumext_wrap_label_vii_bool
3550         \legacy_if_set_true:n { @noitemarg }
3551         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
3552     }
3553 }

```

(End of definition for __enumext_standar_item_vii:w)

```

\__enumext_starred_item_vii:w
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii: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*[\langle symbol \rangle] and \item*[\langle symbol \rangle][\langle offset \rangle].

```

3554 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3555 {
3556     \bool_set_true:N \__enumext_item_starred_vii_bool
3557     \bool_set_true:N \__enumext_wrap_label_vii_bool
3558     \peek_meaning:NTF [
3559         { \__enumext_starred_item_vii_aux_i:w }
3560         { \__enumext_starred_item_vii_aux_ii:w }
3561     }
3562     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3563     {
3564         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3565         \__enumext_starred_item_vii_aux_ii:w
3566     }
3567     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3568     {
3569         \peek_meaning:NTF [
3570             { \__enumext_starred_item_vii_aux_iii:w }
3571             {
3572                 \dim_set_eq:NN
3573                 \l__enumext_item_symbol_sep_vii_dim
3574                 \l__enumext_labelsep_vii_dim
3575                 \legacy_if_set_true:n { @noitemarg }
3576                 \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
3577             }
3578         }
3579     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3580     {
3581         \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3582         \legacy_if_set_true:n { @noitemarg }
3583         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
3584     }

```

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

11.35.3 Real definition of \item in enumext*

```
\__enumext_start_item_vii:w
```

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

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

```

3585 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3586 {
3587     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3588     \legacy_if:nT { @noitemarg }
3589     {
3590         \legacy_if_set_false:n { @noitemarg }
3591         \legacy_if:nT { @nmbrlist }
3592         {
3593             \bool_if:NT \l__enumext_hyperref_bool
3594             {
3595                 \legacy_if_set_true:n { @hyper@item }
3596             }
3597             \refstepcounter{enumXvii}
3598             \bool_if:NT \l__enumext_check_answers_bool
3599             {
3600                 \int_gincr:N \g__enumext_item_number_int
3601             }

```

```

3602     }
3603 }

```

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.

```

3604 \group_begin:
3605   \lrbox{ \l__enumext_item_text_vii_box }
3606   \bool_if:NF \l__enumext_footnotes_key_bool
3607   {
3608     \__enumext_renew_footnote:
3609   }
3610   \bool_if:NT \l__enumext_item_starred_vii_bool
3611   {
3612     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3613     {
3614       \tl_gset_eq:NN
3615         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3616     }
3617     \mode_leave_vertical:
3618     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3619     \makebox[ \opt ]{ r }{ \g__enumext_item_symbol_aux_vii_tl }
3620     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3621     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3622   }
3623   \group_begin:
3624     \tl_use:N \l__enumext_label_font_style_vii_tl
3625     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3626     {
3627       \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }
3628       { \__enumext_wrapper_label_vii:n {#1} }
3629     }
3630     {
3631       \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }{ #1 }
3632     }
3633   \group_end:
3634   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3635   \tl_use:N \l__enumext_after_list_args_vii_tl
3636   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3637   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3638   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3639   \tl_use:N \l__enumext_fake_item_indent_vii_tl
3640 }

```

(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.

```

3641 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3642 {
3643   \__enumext_endminipage:
3644   \endlrbox
3645   \group_end:
3646   \box_set_wd:Nn \l__enumext_item_text_vii_box
3647   {
3648     \l__enumext_joined_width_vii_dim
3649     + \l__enumext_labelwidth_vii_dim
3650     + \l__enumext_labelsep_vii_dim
3651   }
3652   \int_set:Nn \hbadness { 10000 }
3653   \box_use:N \l__enumext_item_text_vii_box
3654   \bool_if:NF \l__enumext_footnotes_key_bool
3655   {
3656     \__enumext_print_footnote:
3657   }
3658   \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }

```

```

3659     {
3660         \par\noindent
3661         \int_zero:N \l__enumext_item_column_pos_vii_int
3662     }
3663     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3664 }

```

(End of definition for \l__enumext_stop_item_vii:.)

\l__enumext_remove_extra_parsep_vii:

Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

3665 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
3666 {
3667     \int_compare:nNnT
3668     {
3669         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3670     }
3671     =
3672     { \c_zero_int }
3673     {
3674         \par
3675         \vspace{ -\l__enumext_itemsep_vii_skip }
3676         \int_gzero:N \g__enumext_item_count_all_vii_int
3677     }
3678 }

```

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

```

3679 \l__enumext_after_env:nn {enumext*} { \l__enumext_execute_after_env: }

```

11.36 The environment keyans*

11.36.1 Functions for item box width

\l__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.

```

3680 \cs_new_protected:Nn \l__enumext_starred_columns_set_viii:
3681 {
3682     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3683     {
3684         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3685         {
3686             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3687             / \l__enumext_columns_viii_int
3688         }
3689     }
3690     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3691     \dim_set:Nn \l__enumext_item_width_viii_dim
3692     {
3693         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3694         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3695         - \l__enumext_labelsep_viii_dim
3696     }
3697     \dim_zero_new:N \itemwidth
3698 }

```

(End of definition for \l__enumext_starred_columns_set_viii:.)

\l__enumext_starred_joined_item_viii:n

The function `\l__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the content passed to `\item<number>` will be stored together with the value of `\itemwidth`.

```

3699 \cs_new_protected:Npn \l__enumext_starred_joined_item_viii:n #1
3700 {
3701     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3702     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3703     {
3704         \msg_warning:nnee { enumext } { item-joined }
3705         { \int_use:N \l__enumext_joined_item_viii_int }
3706         { \int_use:N \l__enumext_columns_viii_int }
3707         \int_set:Nn \l__enumext_joined_item_viii_int
3708         {

```



```

3709         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3710     }
3711 }
3712 \int_compare:nNnT
3713 { \l__enumext_joined_item_viii_int }
3714 >
3715 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3716 {
3717     \msg_warning:nnee { enumext } { item-joined-columns }
3718     { \int_use:N \l__enumext_joined_item_viii_int }
3719     {
3720         \int_eval:n
3721         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3722     }
3723     \int_set:Nn \l__enumext_joined_item_viii_int
3724     {
3725         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3726     }
3727 }
3728 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3729 {
3730     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3731     \int_decr:N \l__enumext_joined_item_aux_viii_int
3732     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3733     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3734     \dim_set:Nn \l__enumext_joined_width_viii_dim
3735     {
3736         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3737         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3738             + \l__enumext_columns_sep_viii_dim
3739             ) * \l__enumext_joined_item_aux_viii_int
3740     }
3741     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3742 }
3743 {
3744     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3745     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3746 }
3747 }

```

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

`__enumext_start_mini_viii:`
`__enumext_stop_mini_viii:`

The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.

```

3748 \cs_new_protected:Nn \__enumext_start_mini_viii:
3749 {
3750     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3751     {
3752         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3753         {
3754             \linewidth
3755             - \l__enumext_minipage_right_viii_dim
3756             - \l__enumext_minipage_hsep_viii_dim
3757         }
3758         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3759         \dim_gset_eq:NN
3760         \g__enumext_minipage_right_viii_dim
3761         \l__enumext_minipage_right_viii_dim
3762         \__enumext_mini_addvspace_viii:
3763         \nointerlineskip\noindent
3764         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3765     }
3766 }
3767 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3768 {
3769     \bool_if:NT \l__enumext_minipage_active_viii_bool
3770     {
3771         \end{__enumext_mini_env*}
3772         \hfill
3773         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3774     }
3775 }

```

```

3776 \__enumext_after_env:nn {keyans*}
3777 {
3778   \bool_if:NT \g__enumext_minipage_active_viii_bool
3779   {
3780     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3781     \par\addvspace { \g__enumext_minipage_right_skip }
3782     \bool_if:NF \g__enumext_minipage_center_viii_bool
3783     {
3784       \centering
3785     }
3786     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3787     \end{__enumext_mini_env*}
3788     \par\addvspace{ \g__enumext_minipage_after_skip }
3789   }
3790   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3791   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3792   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3793   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3794 }

```

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

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

```

3795 \NewDocumentEnvironment{keyans*}{ o }
3796 {
3797   \__enumext_safe_exec_viii:
3798   \__enumext_parse_keys_viii:n {#1}
3799   \__enumext_before_list_viii:
3800   \__enumext_start_list:nn { }
3801   {
3802     \__enumext_list_arg_two_viii:
3803     \__enumext_before_keys_exec_viii:
3804   }
3805   \__enumext_starred_columns_set_viii:
3806   \item[] \scan_stop:
3807   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3808   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3809 }
3810 {
3811   \__enumext_stop_item_tmp_viii:
3812   \__enumext_remove_extra_parsep_viii:
3813   \__enumext_check_starred_cmd:n { item }
3814   \__enumext_stop_list:
3815   \__enumext_after_list_viii:
3816 }

```

(End of definition for keyans*. This function is documented on page 12.)

__enumext_safe_exec_viii: First check the maximum nesting level for the **keyans*** environment.

```

3817 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3818 {
3819   \int_incr:N \l__enumext_keyans_level_h_int
3820   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3821   {
3822     \msg_error:nn { enumext } { nested }
3823   }
3824   \__enumext_keyans_save_start_line:
3825   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3826   \bool_set_false:N \l__enumext_store_active_bool
3827   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3828   {
3829     \msg_error:nn { enumext } { keyans-wrong-level }
3830   }
3831 }

```

(End of definition for __enumext_safe_exec_viii:.)

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

```

3832 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3833 {
3834   \tl_if_novalue:nF {#1}
3835   {
3836     \keys_set:nn { enumext / keyans* } {#1}
3837   }
3838 }

```

(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*.

```

3839 \cs_new_protected:Nn \__enumext_before_list_viii:
3840 {
3841   \__enumext_vspace_above_viii:
3842   \__enumext_before_args_exec_viii:
3843   \__enumext_start_mini_viii:
3844 }

```

(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.

```

3845 \cs_new_protected:Nn \__enumext_after_list_viii:
3846 {
3847   \__enumext_stop_mini_viii:
3848   \__enumext_after_stop_list_viii:
3849   \__enumext_vspace_below_viii:
3850 }

```

(End of definition for `__enumext_after_list_viii:`.)

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

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

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

```

3851 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3852 {
3853   \__enumext_stop_item_tmp_viii:
3854   \int_incr:N \l__enumext_item_column_pos_viii_int
3855   \int_gincr:N \g__enumext_item_count_all_viii_int
3856   \__enumext_item_peek_args_viii:
3857 }

```

(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).

```

3858 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3859 {
3860   \peek_meaning:NTF (
3861     { \__enumext_joined_item_viii:w }
3862     { \__enumext_joined_item_viii:w (1) }
3863   )

```

(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`.

```

3864 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3865 {
3866   \__enumext_starred_joined_item_viii:n {#1}
3867   \peek_meaning_remove:NTF *
3868   { \__enumext_starred_item_viii:w }
3869   { \__enumext_standar_item_viii:w }
3870 }

```

(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 [\l__enumext_label_viii_tl]`.

```

3871 \cs_new_protected:Npn \__enumext_standar_item_viii:w
3872 {
3873   \bool_set_false:N \l__enumext_item_starred_viii_bool
3874   \peek_meaning:NTF [
3875   {
3876     \bool_set_eq:NN
3877     \l__enumext_wrap_label_viii_bool
3878     \l__enumext_wrap_label_opt_viii_bool
3879     \__enumext_start_item_viii:w
3880   }
3881   {
3882     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3883     \legacy_if_set_true:n { @noitemarg }
3884     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3885   }
3886 }

```

(End of definition for `__enumext_standar_item_viii:w`.)

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

```

3887 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3888 {
3889   \bool_set_true:N \l__enumext_item_starred_viii_bool
3890   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3891   \peek_meaning:NTF [
3892   { \__enumext_starred_item_viii_aux_i:w }
3893   { \__enumext_starred_item_viii_aux_ii:w }
3894 }

```

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

```

3895 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3896 {
3897   \tl_clear:N \l__enumext_store_keyans_label_tl
3898   \tl_if_no_value:nF { #1 }
3899   {
3900     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3901     {
3902       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3903       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3904     }
3905     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3906   }
3907   \__enumext_starred_item_viii_aux_ii:w
3908 }

```

```

3909 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3910 {
3911   \legacy_if_set_true:n { @noitemarg }
3912   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3913 }

```

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

`__enumext_starred_item_exec:`

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

```

3914 \cs_new_protected:Nn \__enumext_starred_item_exec:
3915 {
3916   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3917   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3918   \__enumext_keyans_store_ref:
3919   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3920   \__enumext_keyans_addto_seq_link:
3921   \int_gincr:N \g__enumext_check_starred_cmd_int
3922   \bool_if:NT \__enumext_show_answer_bool
3923   {
3924     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3925   }
3926   \bool_if:NT \__enumext_show_position_bool
3927   {
3928     \tl_set:Ne \__enumext_mark_answer_sym_tl
3929     {
3930       \group_begin:
3931       \exp_not:N \normalfont
3932       \exp_not:N \footnotesize [ \int_eval:n
3933       {
3934         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3935       }
3936       ]
3937       \group_end:
3938     }
3939     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3940   }
3941 }

```

(End of definition for `__enumext_starred_item_exec:`.)

Real definition of `\item in keyans*`

`__enumext_start_item_viii:w`

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

```

3942 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3943 {
3944   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3945   \legacy_if:nT { @noitemarg }
3946   {
3947     \legacy_if_set_false:n { @noitemarg }
3948     \legacy_if:nT { @nmbrlist }
3949     {
3950       \bool_if:NT \__enumext_hyperref_bool
3951       {
3952         \legacy_if_set_true:n { @hyper@item }
3953       }
3954       \refstepcounter{enumXviii}
3955     }
3956   }

```

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

```

3957   \group_begin:
3958   \lrbox{ \__enumext_item_text_viii_box }
3959   \bool_if:NF \__enumext_footnotes_key_bool
3960   {
3961     \__enumext_renew_footnote:
3962   }
3963   \bool_if:NT \__enumext_item_starred_viii_bool
3964   {

```

```

3965         \__enumext_starred_item_exec:
3966     }
3967     \group_begin:
3968         \tl_use:N \__enumext_label_font_style_viii_tl
3969         \bool_if:NTF \__enumext_wrap_label_viii_bool
3970         {
3971             \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]
3972                 { \__enumext_wrapper_label_viii:n {#1} }
3973         }
3974         {
3975             \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]{ #1
3976         }
3977     \group_end:
3978     \skip_horizontal:N \__enumext_labelsep_viii_dim
3979     \tl_use:N \__enumext_after_list_args_viii_tl
3980     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_viii_dim }
3981     \skip_set_eq:NN \parindent \__enumext_listparindent_viii_dim
3982     \skip_set_eq:NN \parskip \__enumext_parsep_viii_skip
3983     \bool_if:NT \__enumext_item_starred_viii_bool
3984     {
3985         \tl_use:N \__enumext_fake_item_indent_viii_tl
3986         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\__enumext_fake_item_indent
3987     }
3988     {
3989         \tl_use:N \__enumext_fake_item_indent_viii_tl
3990     }
3991 }

```

(End of definition for __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The function __enumext_stop_item_viii: shall terminate with the capture of \item and its *contents*. Close the environments minipage, lrbox and the group. Then we only have to set the width of the box and print it next to \footnote, and add the horizontal and vertical separation between the boxes.

```

3992 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3993 {
3994     \__enumext_endminipage:
3995     \endlrbox
3996     \group_end:
3997     \box_set_wd:Nn \__enumext_item_text_viii_box
3998     {
3999         \__enumext_joined_width_viii_dim
4000         + \__enumext_labelwidth_viii_dim
4001         + \__enumext_labelsep_viii_dim
4002     }
4003     \int_set:Nn \hbadness { 10000 }
4004     \box_use:N \__enumext_item_text_viii_box
4005     \bool_if:NF \__enumext_footnotes_key_bool
4006     {
4007         \__enumext_print_footnote:
4008     }
4009     \int_compare:nNnTF
4010     { \__enumext_item_column_pos_viii_int } = { \__enumext_columns_viii_int }
4011     {
4012         \par\noindent
4013         \int_zero:N \__enumext_item_column_pos_viii_int
4014     }
4015     { \hspace{ \__enumext_columns_sep_viii_dim } }
4016 }

```

(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.

```

4017 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4018 {
4019     \int_compare:nNnT
4020     {
4021         \int_mod:nn
4022         { \__enumext_item_count_all_viii_int }
4023         { \__enumext_columns_viii_int }
4024     }

```

```

4025     =
4026     { \c_zero_int }
4027     {
4028         \par
4029         \vspace{ -\l__enumext_itemsep_viii_skip }
4030         \int_gzero:N \g__enumext_item_count_all_viii_int
4031     }
4032 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:.`)

11.37 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.

```

4033 \NewDocumentCommand \getkeyans { m }
4034 {
4035     \exp_args:Ne \__enumext_getkeyans_aux:n
4036     { \tl_to_str:e { \text_expand:n {#1} } }
4037 }

```

(End of definition for `\getkeyans`. This function is documented on page 14.)

`__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.

```

4038 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4039 {
4040     \str_if_in:nnTF {#1} { : }
4041     {
4042         \use:e
4043         {
4044             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4045             { {##1} {##2} }
4046         }
4047         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4048     }
4049     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4050 }

```

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

`__enumext_getkeyans:nn`

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

```

4051 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4052 {
4053     \prop_if_exist:cF { g__enumext_#1_prop }
4054     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4055     \group_begin:
4056     \prop_item:cn { g__enumext_#1_prop }{#2}
4057     \group_end:
4058 }

```

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

11.38 The command `\printkeyans`

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

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

```

4059 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4060 \keys_define:nn { enumext / print }
4061 {
4062     print* .code:n = \keys_precompile:neN { enumext / enumext* }
4063     { \__enumext_filter_save_key:n {#1} }

```



```

4064 \l__enumext_print_keyans_starred_tl, % starred cmd
4065 print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4066 print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4067 { \l__enumext_filter_save_key:n {#1} }
4068 \l__enumext_print_keyans_i_tl,
4069 print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4070 print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4071 { \l__enumext_filter_save_key:n {#1} }
4072 \l__enumext_print_keyans_ii_tl,
4073 print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4074 print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4075 { \l__enumext_filter_save_key:n {#1} }
4076 \l__enumext_print_keyans_iii_tl,
4077 print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4078 print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4079 { \l__enumext_filter_save_key:n {#1} }
4080 \l__enumext_print_keyans_iv_tl,
4081 print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4082 print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4083 { \l__enumext_filter_save_key:n {#1} }
4084 \l__enumext_print_keyans_vii_tl, % starred nested
4085 print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4086 }

```

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

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

```

4087 \NewDocumentCommand \printkeyans { s O{} m }
4088 {
4089   \group_begin:
4090     \tl_use:N \l__enumext_print_keyans_i_tl
4091     \tl_use:N \l__enumext_print_keyans_ii_tl
4092     \tl_use:N \l__enumext_print_keyans_iii_tl
4093     \tl_use:N \l__enumext_print_keyans_iv_tl
4094     \tl_use:N \l__enumext_print_keyans_vii_tl
4095     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4096   \group_end:
4097 }

```

(End of definition for `\printkeyans`. This function is documented on page 14.)

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

```

4098 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4099 {
4100   \seq_if_exist:cTF { g__enumext_#3_seq }
4101   {
4102     \seq_if_empty:cF { g__enumext_#3_seq }
4103     {
4104       %%\seq_show:c { g__enumext_#3_seq }

```

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

```

4105   \bool_if:nTF {#1}
4106   {
4107     \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4108     {
4109       \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4110     }
4111     {
4112       \tl_use:N \l__enumext_print_keyans_starred_tl
4113       \begin{enumext*}[#2]
4114         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4115       \end{enumext*}
4116     }
4117   }

```

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

```

4118         {
4119             \begin{enumext}[#2]
4120             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4121             \end{enumext}
4122         }
4123     }
4124 }
4125 {
4126     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4127 }
4128 }

```

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

11.39 The command `\setenumext`

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

```

4129 \keys_define:nn { enumext / meta-families }
4130 {
4131     enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4132     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4133     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4134     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4135     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4136     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4137     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4138     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4139     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4140     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4141     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4142     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4143     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4144     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4145 }

```

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

```

4146 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4147 {
4148     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4149     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4150 }

```

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

```

4151 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4152 {
4153     \tl_if_novalue:nTF {#1}
4154     {
4155         \seq_map_inline:Nn \c__enumext_all_families_seq
4156     }
4157     {
4158         \seq_clear:N \l__enumext_setkey_tmpa_seq
4159         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4160         \int_set:Nn \l__enumext_setkey_tmpa_int
4161         {
4162             \seq_count:N \l__enumext_setkey_tmpb_seq
4163         }
4164         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4165         {
4166             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4167             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4168             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4169             {
4170                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4171             }
4172         }
4173         {
4174             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4175         }
4176         \seq_if_empty:NNTF \l__enumext_setkey_tmpa_seq
4177         { \seq_map_inline:Nn \c__enumext_all_families_seq }

```

```

4178         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4179     }
4180     {
4181         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4182     }
4183 }

```

(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.

```

4184 \cs_new_protected:Npn \__enumext_set_parse:n #1
4185 {
4186     \tl_set:Nc \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4187     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4188     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4189     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4190     {
4191         \seq_put_right:Nc \l__enumext_setkey_tmpa_seq
4192         { \tl_trim_spaces:n {#1} }
4193     }
4194     { \__enumext_set_error:nn {#1} { } }
4195 }
4196 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4197 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.40 Messages

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

```

4198 \msg_new:nnn { enumext } { package-load }
4199 {
4200     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4201 }
4202 \msg_new:nnn { enumext } { package-not-load }
4203 {
4204     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4205 }
4206 \msg_new:nnn { enumext } { package-load-foot }
4207 {
4208     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4209 }

```

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

```

4210 \msg_new:nnn { enumext } { counters }
4211 {
4212     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4213     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4214 }

```

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

```

4215 \msg_new:nnn { enumext } { store-prop }
4216 {
4217     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4218 }
4219 \msg_new:nnn { enumext } { store-seq }
4220 {
4221     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4222 }
4223 \msg_new:nnn { enumext } { store-int }
4224 {
4225     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4226 }
4227 \msg_new:nnn { enumext } { prop-seq-int-hook }
4228 {
4229     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4230     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4231     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4232 }
4233 \msg_new:nnn { enumext } { item-answer-hook }
4234 {
4235     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4236     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\

```

```

4237 * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4238 }

```

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

```

4239 \msg_new:nnn { enumext } { invalid-key }
4240 {
4241   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4242 }
4243 \msg_new:nnn { enumext } { unknown-key-family }
4244 {
4245   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4246 }

```

Messages used in length calculation.

```

4247 \msg_new:nnn { enumext } { width-negative }
4248 {
4249   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4250   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4251 }
4252 \msg_new:nnn { enumext } { width-zero }
4253 {
4254   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4255   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4256 }

```

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

```

4257 \msg_new:nnn { enumext } { list-lengths }
4258 {
4259   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4260   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4261   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4262   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4263   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4264   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4265   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4266   \__enumext_show_length:nnn { skip } { topsep } {#1}
4267   \__enumext_show_length:nnn { skip } { parsep } {#1}
4268   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4269   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4270   ****~
4271 }

```

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

```

4272 \msg_new:nnn { enumext } { list-lengths-not-nested }
4273 {
4274   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4275   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4276   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4277   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4278   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4279   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4280   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4281   \__enumext_show_length:nnn { skip } { topsep } {#1}
4282   \__enumext_show_length:nnn { skip } { parsep } {#1}
4283   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4284   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4285   ****~
4286 }

```

Messages used by `ref` key.

```

4287 \msg_new:nnn { enumext } { key-ref-empty }
4288 {
4289   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4290 }

```

Messages used by `save-ans` key.

```

4291 \msg_new:nnn { enumext } { save-ans-empty }
4292 {
4293   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4294 }
4295 \msg_new:nnn { enumext } { save-ans-log }
4296 {
4297   * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save
      ans=#2 ~ \msg_line_context:.

```

```

4298     }
4299     \msg_new:nnn { enumext } { save-ans-log-hook }
4300     {
4301         * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save
         ans=#2 ~ \msg_line_context:.
4302     }
4303     \msg_new:nnn { enumext } { save-ans-hook }
4304     {
4305         Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4306     }

```

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

```

4307 \msg_new:nnn { enumext } { need-save-ans }
4308 {
4309     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4310 }
4311 \msg_new:nnn { enumext } { items-same-answer }
4312 {
4313     *****\
4314     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right
4315     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\
4316     *****
4317 }
4318 \msg_new:nnn { enumext } { item-greater-answer }
4319 {
4320     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4321     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4322     Items ~ > ~ Answers.
4323 }
4324 \msg_new:nnn { enumext } { item-less-answer }
4325 {
4326     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4327     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4328     Items ~ < ~ Answers.
4329 }

```

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

```

4330 \msg_new:nnn { enumext } { missing-starred }
4331 {
4332     Missing ~ '\c_backslash_str #1*' ~ #2.
4333 }
4334 \msg_new:nnn { enumext } { many-starred }
4335 {
4336     Many ~ '\c_backslash_str #1*' ~ #2.
4337 }

```

Messages used by `\printkeyans*` command.

```

4338 \msg_new:nnn { enumext } { print-starred }
4339 {
4340     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4341     #2 ~ environment ~ \msg_line_context:.
4342 }

```

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

```

4343 \msg_new:nnn { enumext } { list-too-deep }
4344 {
4345     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
4346     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4347 }

```

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

```

4348 \msg_new:nnn { enumext } { anskey-empty-arg }
4349 {
4350     Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4351 }
4352 \msg_new:nnn { enumext } { anskey-wrong-place }
4353 {
4354     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
4355     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4356 }
4357 \msg_new:nnn { enumext } { anskey-nested }
4358 {
4359     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.

```

```

4360 }
4361 \msg_new:nnn { enumext } { anskey-nested-env }
4362 {
4363   The ~ environment ~ anskey* ~ can't ~ be ~ nested ~ \msg_line_context:.
4364 }
4365 \msg_new:nnn { enumext } { anspic-wrong-place }
4366 {
4367   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4368   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4369 }
4370 \msg_new:nnn { enumext } { command-wrong-place }
4371 {
4372   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4373   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4374 }

```

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

```

4375 \msg_new:nnn { enumext } { keyans-nested }
4376 {
4377   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4378 }
4379 \msg_new:nnn { enumext } { keyans-wrong-level }
4380 {
4381   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4382   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4383 }
4384 \msg_new:nnn { enumext } { wrong-place }
4385 {
4386   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4387   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4388 }
4389 \msg_new:nnn { enumext } { keyanspic-nested }
4390 {
4391   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4392 }
4393 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4394 {
4395   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4396   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4397 }

```

Messages used by **\getkeyans** command.

```

4398 \msg_new:nnn { enumext } { undefined-storage-anskey }
4399 {
4400   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4401 }

```

Messages used by **\miniright** command.

```

4402 \msg_new:nnn { enumext } { missing-miniright }
4403 {
4404   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4405   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4406 }
4407 \msg_new:nnn { enumext } { wrong-miniright-place }
4408 {
4409   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4410   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4411 }
4412 \msg_new:nnn { enumext } { wrong-miniright-use }
4413 {
4414   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4415   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4416 }

```

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

```

4417 \msg_new:nnn { enumext } { nested }
4418 {
4419   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4420 }
4421 \msg_new:nnn { enumext } { item-joined }
4422 {
4423   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4424 }

```

```
4425 \msg_new:nnn { enumext } { item-joined-columns }
4426 {
4427   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4428 }
```

11.41 Finish package

Finish package implementation.

```
4429 \file_input_stop:
4430 </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>	202
<code>\+</code>	194
<code>\-</code>	194
<code>\\</code> 210, 3226, 4212, 4229, 4230, 4235, 4236, 4249, 4254, 4259, 4274, 4313, 4314, 4315, 4320, 4321, 4326, 4327, 4345, 4354, 4367, 4372, 4381, 4386, 4395, 4404, 4409, 4414	
A	
<code>above</code>	<u>1366</u>
<code>above*</code>	<u>1366</u>
<code>\addvspace</code> 1013, 1041, 1164, 1243, 1306, 1312, 1340, 1357, 3065, 3080, 3182, 3197, 3423, 3430, 3781, 3788	
<code>after</code>	<u>851</u>
<code>align</code>	<u>468</u>
<code>\Alph</code>	34, 39
<code>\Alph</code>	420, 529, 574, 642, 4081
<code>\alph</code>	34, 39
<code>\alph</code>	421, 527, 4073
<code>\anskey</code>	12, 70, <u>2195</u>
<code>anskey*</code>	<u>2411</u>
<code>\anspic</code>	14, 92, <u>3204</u>
<code>\anspic*</code>	65
<code>\arabic</code>	29, 34
<code>\arabic</code>	419, 526, 573, 4065, 4069, 4085
B	
<code>\baselineskip</code>	47
<code>\baselineskip</code>	2140, 2148
<code>before</code>	<u>851</u>
<code>before*</code>	<u>851</u>
<code>below</code>	<u>1366</u>
<code>below*</code>	<u>1366</u>
bool commands:	
<code>\bool_gset_false:N</code> 308, 309, 310, 3432, 3436, 3790	
<code>\bool_gset_true:N</code> 222, 231, 955, 1857, 1863, 3415, 3433, 3773, 3791	
<code>\bool_if:NTF</code> 360, 372, 389, 1388, 1402, 1415, 1426, 1437, 1448, 1459, 1470, 1523, 1540, 1545, 1553, 1580, 1618, 1623, 1630, 1634, 1656, 1661, 1669, 1676, 1707, 1715, 1807, 1931, 2013, 2023, 2102, 2126, 2133, 2161, 2199, 2209, 2237, 2263, 2380, 2391, 2395, 2414, 2424, 2472, 2487, 2562, 2573, 2577, 2690, 2720, 2794, 2810, 2872, 2882, 2912, 2917, 2999, 3049, 3063, 3071, 3110, 3167, 3180, 3188, 3206, 3411, 3420, 3424, 3500, 3510, 3593, 3598, 3606, 3610, 3625, 3654, 3769, 3778, 3782, 3922, 3926, 3950, 3959, 3963, 3969, 3983, 4005	
<code>\bool_if:nTF</code> 1341, 1358, 2731, 2766, 2830, 3227, 4105	
<code>\bool_if_p:N</code> 240, 254, 1687, 1688, 1696, 1697, 1820, 1842, 1854, 1855, 1860, 1861, 2248, 2289, 2290, 2314, 2323, 2324, 2336, 2352, 2549, 2550, 2587, 2588, 2972, 2985, 2987, 3234, 3235	
<code>\bool_lazy_all:nTF</code> 238, 252, 1818, 1840, 2312, 2321, 2334, 2350, 2970, 2983	
<code>\bool_lazy_and:nnTF</code> 218, 227, 1686, 1695, 1853, 1859, 2247, 2254, 2288, 2548	
<code>\bool_lazy_or:nnTF</code> 1748, 1755, 2586, 3233	
<code>\bool_new:N</code> 26, 27, 28, 29, 30, 31, 32, 52, 62, 83, 88, 89, 94, 95, 98, 116, 119, 122, 123, 132, 133, 134, 142, 143, 157, 168, 170	
<code>\bool_not_p:n</code> 219, 228, 2249, 2255, 2339, 2354, 2973, 2974, 2986	
<code>\bool_set_eq:NN</code> 2698, 2746, 3543, 3876	
<code>\bool_set_false:N</code> 369, 1792, 1793, 3086, 3118, 3200, 3265, 3283, 3495, 3540, 3826, 3873	
<code>\bool_set_true:N</code> 245, 259, 351, 355, 461, 779, 1372, 1377, 1643, 1765, 1766, 2045, 2053, 2694, 2724, 2742, 2754, 2948, 2979, 2992, 3018, 3115, 3142, 3400, 3469, 3549, 3556, 3557, 3758, 3882, 3889, 3890	
box commands:	
<code>\box_dp:N</code> 1060, 1064, 1068, 1079, 1083, 1094, 1103, 1109, 1119, 1132, 1138, 1144, 1175, 1176, 1177, 1180, 1190, 1194, 1203, 1210, 1215, 1223, 1252, 1253, 1256, 1263, 1276, 1284, 1290, 1298, 3295	
<code>\box_new:N</code> 59, 163	
<code>\box_set_wd:Nn</code> 3646, 3997	
<code>\box_use:N</code> 3653, 4004	
<code>\box_wd:N</code> 427	
C	
<code>\c</code> 202, 203, 679, 681, 693, 695	
<code>\cB</code> 203	
<code>\cE</code> 203	
<code>\centering</code> 1343, 1360, 3316, 3426, 3784	
<code>check-ans</code> 1784	
Document class:	
<code>article</code> 40	
clist commands:	
<code>\clist_const:Nn</code> 175	
<code>\clist_map_function:nN</code> 3303	
<code>\clist_map_inline:Nn</code> 467, 721, 784, 850, 865, 946, 1382	
<code>\clist_map_inline:nn</code> 37, 48, 67, 73, 85, 97, 121, 151, 174, 492, 509, 789, 961, 1488, 1732, 1798, 1992, 2010, 2042, 2309, 2481, 2648, 2859, 2862, 2889, 2899, 2902, 2922, 4187	
<code>\columnbreak</code> 71	
<code>\columnbreak</code> 2251	
<code>columns</code> 930	
<code>columns-sep</code> 930	
<code>\columnsep</code> 88, 91	
<code>\columnsep</code> 3043, 3164	
<code>\columnseprule</code> 88, 91	
<code>\columnseprule</code> 3047, 3166	
Commands provide by enumext :	
<code>\anskey</code> 26, 27, 61, 62, 67, 68, 70, 72–74, 76, 78, 86, 98, 109, 110, 114	
<code>\anspic*</code> 26, 27, 65, 75–77, 92–94, 109, 110	
<code>\anspic</code> 68, 92–94, 114	
<code>\getkeyans</code> 68, 109, 115	
<code>\item*</code> 26, 27, 65, 68, 75–77, 80, 81, 100, 106, 107, 109, 110	
<code>\itemwidth</code> 95, 102	
<code>\item</code> 80, 81, 95, 99, 100, 102, 105, 106	
<code>\miniright</code> 26, 45, 52, 53, 87, 89–91, 115	
<code>\printkeyans*</code> 109	
<code>\printkeyans</code> 27, 68, 109, 110	
<code>\setenumext</code> 27, 110–113	
Counters defined by enumext :	
<code>enumXiii</code> 25, 34	
<code>enumXii</code> 25, 34	

enumXiv	25, 34	3156, 3308, 3326, 3333, 3376, 3394, 3581, 3684, 3691, 3734, 3752
enumXi	25, 34	\dim_set_eq:NN 517, 564, 635, 639, 2713, 2861, 2901, 3043, 3164, 3383, 3386, 3387, 3572, 3741, 3744, 3745
enumXviii	25, 34	\dim_use:N 793, 801, 1333, 1339, 2169, 2172, 2177, 2783, 2785, 3010, 3015, 3016, 3023, 3033, 3037, 3038, 3040
enumXvii	25, 34, 100	\dim_zero:N 3047, 3166, 3287, 3288, 3289
enumXvi	25, 34	\dim_zero_new:N 3339, 3697
enumXv	25, 34	\c_zero_dim 795, 809, 821, 833, 1333, 1351, 2275, 2820, 2825, 2831, 2838, 3010, 3033, 3136, 3154, 3324, 3392, 3682, 3750
cs commands:		E
\cs_generate_variant:Nn	429, 445, 685, 701, 2094, 2099, 2179, 2849, 3305, 4059	\end .. 1336, 1354, 2128, 2163, 3062, 3079, 3179, 3196, 3413, 3429, 3771, 3787, 4107, 4115, 4121
\cs_if_exist:NTF	399	\endlist 32
\cs_new:Nn	188	\endlist 336
\cs_new:Npn	206, 1489, 1498, 1507, 2057, 2066, 2074	\endlrbox 3644, 3995
\cs_new_eq:NN	335, 336, 337, 341, 342, 374, 375, 378, 379	\endminipage 32
\cs_new_protected:Nn	198, 212, 236, 267, 294, 300, 306, 312, 318, 326, 346, 550, 613, 665, 866, 870, 874, 878, 882, 886, 890, 894, 898, 902, 906, 910, 914, 918, 922, 926, 962, 974, 998, 1015, 1026, 1050, 1125, 1149, 1166, 1228, 1245, 1267, 1302, 1308, 1383, 1397, 1411, 1422, 1433, 1444, 1455, 1466, 1551, 1654, 1667, 1684, 1705, 1733, 1738, 1763, 1803, 1813, 1851, 1866, 1873, 1882, 1887, 1892, 1897, 1906, 1911, 1916, 1921, 2100, 2124, 2131, 2159, 2166, 2207, 2300, 2422, 2470, 2485, 2513, 2546, 2582, 2594, 2602, 2653, 2657, 2676, 2727, 2762, 2778, 2788, 2804, 2942, 2968, 2997, 3004, 3027, 3057, 3069, 3108, 3132, 3150, 3175, 3186, 3223, 3267, 3281, 3301, 3306, 3322, 3390, 3409, 3461, 3482, 3489, 3498, 3508, 3525, 3665, 3680, 3748, 3767, 3817, 3839, 3845, 3858, 3914, 4017	\endminipage 342
\cs_new_protected:Npn	180, 184, 382, 397, 414, 424, 430, 530, 575, 647, 672, 686, 1330, 1349, 1519, 1538, 1608, 1641, 1743, 1940, 2011, 2021, 2043, 2051, 2086, 2095, 2222, 2234, 2377, 2389, 2437, 2449, 2523, 2567, 2686, 2704, 2738, 2750, 2818, 2852, 2892, 2951, 3128, 3276, 3341, 3472, 3531, 3538, 3554, 3562, 3567, 3579, 3699, 3832, 3864, 3871, 3887, 3895, 3909, 4038, 4051, 4098, 4184, 4196	enumext 5, 2923
\cs_new_protected_nopar:Nn	3518, 3641, 3851, 3992	enumext internal commands:
\cs_new_protected_nopar:Npn	3585, 3942	\l__enumext__check_start_line_env_tl ... 30
\cs_set:Nn	2382	\l__enumext__ref_the_count_tl 36
\cs_set:Npn	2310, 2348, 4044	\l__enumext__resume_name_tl 57
\cs_set_eq:NN	3451, 3452, 3587, 3807, 3808, 3944	__enumext_add_pre_parsep: ... 46, 972, 974, 974
\cs_set_protected:Nn	790, 806, 818, 830	__enumext_after_args_exec: . 43, 866, 878, 2935
\cs_set_protected:Npn	33, 42, 60, 68, 80, 86, 113, 147, 155, 446, 468, 497, 510, 557, 702, 722, 766, 785, 842, 851, 930, 947, 1366, 1477, 1724, 1784, 1957, 1993, 2029, 2302, 2474, 2637, 2850, 2890	__enumext_after_args_exec_v: . 43, 44, 882, 894, 3101
\cs_to_str:N	416, 439	__enumext_after_args_exec_vii: ... 898, 922
D		__enumext_after_args_exec_viii: 926
\d	194	__enumext_after_env:nn 65, 89, 102, 184, 184, 3089, 3418, 3679, 3776
\DeclareDocumentEnvironment	1043	__enumext_after_hyperref: ... 32, 344, 346, 346
dim commands:		__enumext_after_list: .. 89, 98, 105, 2940, 3069, 3069
\dim_abs:n	2823, 2828	\l__enumext_after_list_args_v_tl 896
\dim_add:Nn	3286	\l__enumext_after_list_args_vii_tl 924, 3635
\dim_compare:nNnTF	792, 808, 820, 832, 1332, 1351, 2820, 2825, 2831, 2837, 2839, 2841, 3009, 3032, 3136, 3154, 3278, 3324, 3392, 3682, 3750	\l__enumext_after_list_args_viii_tl 928, 3979
\dim_compare:nTF	2273	__enumext_after_list_v: .. 91, 3106, 3186, 3186
\dim_gset_eq:NN	3401, 3759	__enumext_after_list_vii: ... 3459, 3489, 3489
\dim_gzero:N	3435, 3793	__enumext_after_list_viii: .. 3815, 3845, 3845
\dim_new:N	56, 63, 64, 65, 82, 128, 164, 165, 171	__enumext_after_star_env:nn 96
\dim_set:Nn	427, 780, 2718, 2823, 2828, 2830, 2833, 2834, 2838, 2840, 2843, 2844, 2846, 3012, 3035, 3138,	__enumext_after_stop_list: ... 43, 44, 866, 874, 3084
		__enumext_after_stop_list_v: 43, 882, 890, 3201
		\l__enumext_after_stop_list_v_tl 892
		__enumext_after_stop_list_vii: 898, 914, 3492
		\l__enumext_after_stop_list_vii_tl ... 916
		__enumext_after_stop_list_viii: . 918, 3848
		\l__enumext_after_stop_list_viii_tl ... 920
		\l__enumext_align_label_vii_str .. 3627, 3631
		\l__enumext_align_label_viii_str . 3971, 3975
		\l__enumext_align_label_X_str 155
		\c__enumext_all_envs_clist .. 175, 467, 721, 784, 850, 865, 946, 1382
		\c__enumext_all_families_seq .. 111, 4146, 4155, 4177
		__enumext_anskey_env_safe_inner:n 75, 2411, 2437
		__enumext_anskey_env_safe_outer: . 75, 2411, 2413, 2422
		\l__enumext_anskey_level_int .. 20, 2228, 2229, 2443, 2444

```

\__enumext_anskey_safe_inner:n .. 70, 71, 2202,
    2207, 2222
\__enumext_anskey_safe_outer: . 70, 2197, 2207,
    2207
\__enumext_anskey_wrapper:n ..... 1961, 2387
\__enumext_at_begin_document:n .. 32, 180, 180,
    333, 339
\__enumext_before_args_exec: 43, 866, 866, 3007
\__enumext_before_args_exec_v: 43, 44, 882, 882,
    3135
\__enumext_before_args_exec_vii: .. 898, 898,
    3486
\__enumext_before_args_exec_viii: 902, 3842
\__enumext_before_keys_exec: 43, 866, 870, 2933
\__enumext_before_keys_exec_v: .. 43, 882, 886,
    3099
\__enumext_before_keys_exec_vii ..... 898
\__enumext_before_keys_exec_vii: 44, 906, 3447
\__enumext_before_keys_exec_viii: .. 44, 910,
    3803
\__enumext_before_list: ... 87, 2927, 3004, 3004
\__enumext_before_list_v: . 90, 3094, 3132, 3132
\__enumext_before_list_vii: 98, 3442, 3482, 3482
\__enumext_before_list_viii: .. 105, 3799, 3839,
    3839
\l__enumext_before_no_starred_key_v_tl 888
\l__enumext_before_no_starred_key_vii-
    tl ..... 908
\l__enumext_before_no_starred_key_viii-
    tl ..... 912
\l__enumext_before_starred_key_v_tl ... 884
\l__enumext_before_starred_key_vii_tl . 900
\l__enumext_before_starred_key_viii_tl 904
\__enumext_calc_hspace:NNNNNNN 84, 2818, 2818,
    2849, 2854, 2894
\__enumext_check_ans_active: 62, 87, 1803, 1803,
    3008, 3485
\g__enumext_check_ans_item_tl ..... 77
\g__enumext_check_ans_key_bool 63, 64, 132, 308,
    1857, 1863, 1931
\l__enumext_check_ans_key_bool ... 63, 80, 132,
    1788, 1793, 1854, 1860
\__enumext_check_ans_key_hook: 63, 1851, 1851,
    3083, 3493
\__enumext_check_ans_level: . 62, 63, 1803, 1809,
    1813
\__enumext_check_ans_log: 64, 65, 1897, 1897, 1935
\__enumext_check_ans_log_msg_greater: 1897,
    1903, 1916
\__enumext_check_ans_log_msg_less: 1897, 1901,
    1906
\__enumext_check_ans_log_msg_same_ok: 1897,
    1902, 1911
\__enumext_check_ans_msg_greater: 1873, 1879,
    1892
\__enumext_check_ans_msg_less: 1873, 1877, 1882
\__enumext_check_ans_msg_same_ok: 1873, 1878,
    1887
\__enumext_check_ans_show: .. 64, 65, 1873, 1873,
    1933
\g__enumext_check_ans_show_bool ..... 89
\l__enumext_check_answers_bool 61, 62, 70, 132,
    1766, 1792, 1807, 2102, 2126, 2133, 2161, 2199, 2414,
    2562, 2690, 2720, 3598
\__enumext_check_starred_cmd:n 30, 65, 77, 1940,
    1940, 3104, 3262, 3813
\g__enumext_check_starred_cmd_int 132, 1943,
    1949, 1954, 2760, 3232, 3921
\l__enumext_check_start_line_env_tl 132, 273,
    280, 287, 1946, 1952, 1955
\l__enumext_columns_sep_v_dim 3154, 3156, 3164
\l__enumext_columns_sep_vii_dim .. 3324, 3326,
    3335, 3380, 3663
\l__enumext_columns_sep_viii_dim . 3682, 3684,
    3693, 3738, 4015
\l__enumext_columns_v_int 1171, 3152, 3160, 3172,
    3177
\l__enumext_columns_vii_int .. 3329, 3332, 3336,
    3344, 3348, 3351, 3357, 3363, 3367, 3658, 3669
\l__enumext_columns_viii_int . 3687, 3690, 3694,
    3702, 3706, 3709, 3715, 3721, 3725, 4010, 4023
\l__enumext_counter_i_tl ..... 33, 406
\l__enumext_counter_ii_tl ..... 33, 407
\l__enumext_counter_iii_tl ..... 33, 408
\l__enumext_counter_iv_tl ..... 33, 409
\c__enumext_counter_style_tl ..... 29, 38, 200
\g__enumext_counter_styles_tl . 25, 34, 56, 417,
    435
\l__enumext_counter_v_tl ..... 33, 410, 655
\l__enumext_counter_vi_tl ..... 33, 411
\l__enumext_counter_vii_tl ..... 33, 412, 585
\l__enumext_counter_viii_tl ..... 33, 413, 602
\l__enumext_current_widest_dim 25, 56, 441, 518,
    565, 636, 640
\__enumext_default_item:n ... 2686, 2686, 2735
\__enumext_define_counters:Nn 25, 397, 397, 406,
    407, 408, 409, 410, 411, 412, 413
\__enumext_endminipage: 32, 339, 342, 1049, 3318,
    3643, 3994
\g__enumext_envir_name_tl 30, 137, 246, 260, 316,
    1736, 1741, 1751, 1885, 1890, 1895, 1909, 1914, 1919
\__enumext_execute_after_env: 31, 32, 61, 64, 65,
    1921, 1921, 3089, 3679
\__enumext_fake_item: ..... 790, 790, 2881
\l__enumext_fake_item_indent_v_dim 809, 814
\l__enumext_fake_item_indent_v_tl 811, 2743,
    2747, 2755
\l__enumext_fake_item_indent_vii_dim 821, 826
\l__enumext_fake_item_indent_vii_tl 823, 3639
\l__enumext_fake_item_indent_viii_dim . 833,
    838, 3986
\l__enumext_fake_item_indent_viii_tl .. 835,
    3985, 3989
\l__enumext_fake_item_indent_X_tl ..... 86
\__enumext_fake_item_vii: .... 790, 818, 2911
\__enumext_fake_item_viii: .... 790, 830, 2916
\__enumext_filter_save_key:n .. 68, 2018, 2026,
    2049, 2055, 2057, 2057, 4063, 4067, 4071, 4075, 4079,
    4083
\__enumext_filter_save_key_key:n .. 68, 2057,
    2062, 2066
\__enumext_filter_save_key_pair:nn 68, 2057,
    2063, 2074
\__enumext_filter_series:n 56, 1489, 1489, 1531,
    1543, 1548
\__enumext_filter_series_key:n 56, 1489, 1494,
    1498
\__enumext_filter_series_pair:nn .. 56, 1489,

```

1495, 1507
 \g__enumext_footnote_arg_seq . 152, 2659, 2672, 2682
 \g__enumext_footnote_int . 152, 2666, 2669, 2671, 2673
 \g__enumext_footnote_int_seq . 152, 2660, 2673, 2678, 2681
 __enumext_footnotes_key_bool 32
 \l__enumext_footnotes_key_bool 28, 33, 101, 142, 355, 360, 369, 3606, 3654, 3959, 4005
 __enumext_footnotetext:nn . . . 2653, 2653, 2683
 __enumext_getkeyans:nn . . 109, 4047, 4051, 4051
 __enumext_getkeyans_aux:n 109, 4035, 4038, 4038
 \l__enumext_hyperref_bool . 28, 32, 33, 142, 351, 372, 389, 2290, 2550, 3593, 3950
 __enumext_hypertarget:nn 33, 346, 374, 378, 394
 __enumext_if_is_int:n 192
 __enumext_if_is_int:nTF 192, 674, 688
 __enumext_is_not_nested: 30, 86, 212, 212, 2944, 3463
 __enumext_is_on_first_level: . 30, 86, 212, 236, 2949, 3470
 \g__enumext_item_anskey_int 70, 77, 132, 303, 330, 331, 1870, 2201, 2416, 2564
 __enumext_item_answer_diff: 64, 65, 1866, 1866, 1928
 \g__enumext_item_answer_diff_int 64, 141, 304, 1868, 1875, 1899
 \l__enumext_item_column_pos_vii_int 99, 3351, 3357, 3363, 3367, 3374, 3521, 3658, 3661
 \l__enumext_item_column_pos_viii_int . . 105, 3709, 3715, 3721, 3725, 3732, 3854, 4010, 4013
 \l__enumext_item_column_pos_X_int 155
 \g__enumext_item_count_all_vii_int 99, 3375, 3522, 3669, 3676
 \g__enumext_item_count_all_viii_int 105, 3733, 3855, 4022, 4030
 \g__enumext_item_count_all_X_int 155
 \g__enumext_item_number_int . . 63, 132, 302, 329, 331, 1824, 1828, 1831, 1834, 1846, 1870, 2692, 2722, 3600
 __enumext_item_peek_args_vii: 99, 3523, 3525, 3525
 __enumext_item_peek_args_viii: . . 105, 3856, 3858, 3858
 __enumext_item_starred: . . 82, 2778, 2778, 2796
 \l__enumext_item_starred_vii_bool 3540, 3556, 3610
 \l__enumext_item_starred_viii_bool 3873, 3889, 3963, 3983
 \l__enumext_item_starred_X_bool 155
 __enumext_item_std:w 32, 80, 81, 94, 333, 337, 2695, 2701, 2725, 2743, 2747, 2755, 3299
 \g__enumext_item_symbol_aux_vii_tl 3564, 3612, 3615, 3619, 3621
 \g__enumext_item_symbol_aux_X_tl 155
 \l__enumext_item_symbol_sep_vii_dim . . 3573, 3581, 3618, 3620
 \g__enumext_item_symbol_tl 25, 80, 49, 2710, 2784, 2801
 \l__enumext_item_symbol_vii_tl 3615
 \l__enumext_item_text_vii_box 3605, 3646, 3653
 \l__enumext_item_text_viii_box 3958, 3997, 4004
 \l__enumext_item_text_X_box 155
 \l__enumext_item_width_vii_dim . . . 3333, 3378, 3386, 3387
 \l__enumext_item_width_viii_dim . . 3691, 3736, 3744, 3745
 \l__enumext_item_width_X_dim 155
 \l__enumext_itemindent_X_dim 60
 \l__enumext_itemsep_vii_skip 3675
 \l__enumext_itemsep_viii_skip 4029
 \l__enumext_joined_item_aux_vii_int . . 3372, 3373, 3374, 3375, 3381
 \l__enumext_joined_item_aux_viii_int . 3730, 3731, 3732, 3733, 3739
 \l__enumext_joined_item_aux_X_int 155
 __enumext_joined_item_vii:w . . 99, 3528, 3529, 3531, 3531
 \l__enumext_joined_item_vii_int . . 3343, 3344, 3347, 3349, 3355, 3360, 3365, 3370, 3372, 3378
 __enumext_joined_item_viii:w . 105, 106, 3861, 3862, 3864, 3864
 \l__enumext_joined_item_viii_int . 3701, 3702, 3705, 3707, 3713, 3718, 3723, 3728, 3730, 3736
 \l__enumext_joined_item_X_int 155
 \l__enumext_joined_width_vii_dim . 3376, 3383, 3386, 3636, 3648
 \l__enumext_joined_width_viii_dim 3734, 3741, 3744, 3980, 3999
 \l__enumext_joined_width_X_dim 155
 __enumext_keyans_addto_prop:n 75, 2449, 2449, 2757, 3229
 __enumext_keyans_addto_seq:n . 77, 2523, 2523, 2759, 3231
 __enumext_keyans_addto_seq_link: 2523, 2544, 2546, 3920
 __enumext_keyans_anspic_code:nnn 92, 93, 3220, 3223, 3223
 __enumext_keyans_default_item:n . . 81, 2738, 2738, 2774
 \l__enumext_keyans_env_bool 26, 2973, 2986, 3115, 3200
 __enumext_keyans_fake_item: . . 790, 806, 2871
 \l__enumext_keyans_item_opt_tl . 106, 98, 2571, 2584, 2590, 3905
 \l__enumext_keyans_level_h_int . . 20, 595, 622, 2501, 3819, 3820
 \l__enumext_keyans_level_int . . 20, 1324, 2213, 2428, 2496, 3114, 3119, 3214
 __enumext_keyans_make_label: 35, 83, 2804, 2804, 2869
 __enumext_keyans_mini_addvspace: 50, 90, 1228, 1228, 3144
 __enumext_keyans_mini_right_cmd:n 53, 1326, 1349, 1349
 __enumext_keyans_mini_set_vskip: . 50, 1166, 1166, 1230
 __enumext_keyans_multi_addvspace: 91, 1015, 1026, 3169
 __enumext_keyans_multi_set_vskip: 46, 1015, 1015, 1028
 __enumext_keyans_multicols_start: . . 90, 91, 3148, 3150, 3150
 __enumext_keyans_multicols_stop: . 91, 1353, 3175, 3175, 3199
 __enumext_keyans_parse_keys:n 3093, 3128, 3128
 \l__enumext_keyans_pic_above_int . 127, 3309,

[3310](#), [3312](#)
`\l__enumext_keyans_pic_above_skip` .. [94](#), [127](#),
[3253](#), [3293](#)
`__enumext_keyans_pic_arg_two:` [94](#), [3251](#), [3281](#),
[3281](#)
`\l__enumext_keyans_pic_below_int` . [127](#), [3309](#),
[3310](#), [3313](#)
`\l__enumext_keyans_pic_body_seq` .. [92-94](#), [127](#),
[3218](#), [3258](#), [3317](#)
`__enumext_keyans_pic_do:n` [94](#), [3258](#), [3260](#), [3301](#),
[3301](#), [3305](#)
`\l__enumext_keyans_pic_level_int` .. [20](#), [1316](#),
[2217](#), [2432](#), [2452](#), [2491](#), [2526](#), [2604](#), [3269](#), [3270](#)
`__enumext_keyans_pic_row:n` [94](#), [3303](#), [3306](#), [3306](#)
`__enumext_keyans_pic_safe_exec:` .. [93](#), [3247](#),
[3267](#), [3267](#)
`__enumext_keyans_pic_skip_abs:N` .. [94](#), [3276](#),
[3276](#), [3292](#)
`\l__enumext_keyans_pic_width_dim` . [127](#), [3308](#),
[3315](#)
`__enumext_keyans_redefine_item:` .. [82](#), [2762](#),
[2762](#), [2868](#)
`__enumext_keyans_ref:` [39](#), [647](#), [665](#), [2870](#)
`__enumext_keyans_ref:n` [38](#), [644](#), [647](#), [647](#)
`__enumext_keyans_safe_exec:` . [3092](#), [3108](#), [3108](#)
`__enumext_keyans_save_start_line:` . [30](#), [267](#),
[267](#), [3116](#), [3274](#), [3824](#)
`__enumext_keyans_show_ans:` .. [2567](#), [2575](#), [2594](#)
`__enumext_keyans_show_item_opt:` . [2567](#), [2582](#),
[2755](#), [3243](#), [3986](#)
`__enumext_keyans_show_left:n` . [81](#), [2567](#), [2567](#),
[2753](#), [3238](#)
`__enumext_keyans_show_pos:` .. [2567](#), [2579](#), [2602](#)
`__enumext_keyans_starred_item:n` .. [81](#), [2750](#),
[2750](#), [2770](#)
`__enumext_keyans_store_ref:` .. [76](#), [2470](#), [2470](#),
[2758](#), [3230](#), [3918](#)
`__enumext_keyans_store_ref_aux:i:` [76](#), [2470](#),
[2482](#), [2485](#)
`__enumext_keyans_store_ref_aux:ii:` [77](#), [2470](#),
[2511](#), [2513](#)
`\l__enumext_keyans_tmpa_tl` .. [26](#), [98](#), [2752](#), [2756](#)
`__enumext_keyans_wrapper_opt:n` .. [1964](#), [2590](#)
`\l__enumext_label_copy_i_tl` .. [2344](#), [2489](#), [2494](#),
[2499](#), [2504](#)
`\l__enumext_label_copy_v_tl` [2499](#)
`\l__enumext_label_copy_vi_tl` [2494](#)
`\l__enumext_label_copy_vii_tl` [2319](#), [2330](#), [2361](#),
[2489](#)
`\l__enumext_label_copy_viii_tl` [2504](#)
`\l__enumext_label_copy_X_tl` [144](#)
`\l__enumext_label_fill_left_v_tl` [2808](#)
`\l__enumext_label_fill_left_X_tl` [86](#)
`\l__enumext_label_fill_right_v_tl` [2815](#)
`\l__enumext_label_fill_right_X_tl` [86](#)
`\l__enumext_label_font_style_v_tl` [2809](#), [3242](#)
`\l__enumext_label_font_style_vii_tl` ... [3624](#)
`\l__enumext_label_font_style_viii_tl` .. [3968](#)
`\l__enumext_label_i_tl` [510](#)
`\l__enumext_label_ii_tl` [510](#)
`\l__enumext_label_iii_tl` [510](#)
`\l__enumext_label_iv_tl` [510](#)
`__enumext_label_style:Nnn` [25](#), [34](#), [430](#), [430](#), [445](#),
[515](#), [562](#), [633](#), [637](#)
`\l__enumext_label_v_tl` .. [75](#), [77](#), [630](#), [2457](#), [2531](#),
[2596](#), [2631](#), [2752](#), [2756](#), [3096](#), [3237](#), [3239](#)
`\l__enumext_label_vi_tl` . [75](#), [77](#), [630](#), [2454](#), [2528](#),
[3237](#), [3239](#), [3243](#)
`\l__enumext_label_vii_tl` . [557](#), [3551](#), [3576](#), [3583](#)
`\l__enumext_label_viii_tl` [557](#), [3884](#), [3912](#), [3916](#)
`\l__enumext_label_width_by_box` .. [56](#), [426](#), [427](#)
`__enumext_label_width_by_box:Nn` [34](#), [424](#), [424](#),
[429](#), [441](#), [698](#)
`\l__enumext_labelsep_i_dim` ... [2599](#), [2634](#), [3924](#),
[3939](#)
`\l__enumext_labelsep_v_dim` [3159](#)
`\l__enumext_labelsep_vii_dim` . [3328](#), [3337](#), [3379](#),
[3574](#), [3634](#), [3650](#)
`\l__enumext_labelsep_viii_dim` [3686](#), [3695](#), [3737](#),
[3978](#), [4001](#)
`\l__enumext_labelwidth_i_dim` . [2599](#), [2634](#), [3924](#),
[3939](#)
`\l__enumext_labelwidth_v_dim` [3159](#)
`\l__enumext_labelwidth_vii_dim` ... [3328](#), [3336](#),
[3379](#), [3627](#), [3631](#), [3649](#)
`\l__enumext_labelwidth_viii_dim` .. [3686](#), [3694](#),
[3737](#), [3971](#), [3975](#), [4000](#)
`\l__enumext_leftmargin_tmp_v_bool` . [94](#), [3283](#)
`\l__enumext_leftmargin_tmp_X_bool` [60](#)
`\l__enumext_leftmargin_tmp_X_dim` [60](#)
`\l__enumext_leftmargin_X_dim` [60](#)
`__enumext_level:` [188](#), [188](#), [539](#), [542](#), [543](#), [552](#), [554](#),
[793](#), [797](#), [801](#), [868](#), [872](#), [876](#), [880](#), [964](#), [966](#), [968](#), [970](#),
[1003](#), [1005](#), [1007](#), [1009](#), [1013](#), [1053](#), [1056](#), [1075](#), [1084](#),
[1090](#), [1095](#), [1099](#), [1110](#), [1114](#), [1115](#), [1120](#), [1156](#), [1160](#),
[1333](#), [1339](#), [1386](#), [1388](#), [1390](#), [1393](#), [1400](#), [1402](#), [1404](#),
[1407](#), [2013](#), [2015](#), [2017](#), [2045](#), [2046](#), [2048](#), [2104](#), [2112](#),
[2116](#), [2120](#), [2382](#), [2385](#), [2386](#), [2694](#), [2695](#), [2699](#), [2700](#),
[2701](#), [2708](#), [2710](#), [2714](#), [2715](#), [2718](#), [2724](#), [2725](#), [2780](#),
[2783](#), [2785](#), [2792](#), [2793](#), [2794](#), [2797](#), [2800](#), [2930](#), [2932](#),
[2979](#), [2992](#), [2999](#), [3010](#), [3012](#), [3015](#), [3016](#), [3018](#), [3023](#),
[3030](#), [3033](#), [3035](#), [3037](#), [3038](#), [3039](#), [3040](#), [3043](#), [3049](#),
[3054](#), [3060](#), [3063](#), [3065](#), [3071](#)
`\l__enumext_level_h_int` .. [20](#), [220](#), [242](#), [255](#), [578](#),
[615](#), [1821](#), [1837](#), [2338](#), [2355](#), [3464](#), [3465](#)
`\l__enumext_level_int` . [86](#), [20](#), [190](#), [229](#), [241](#), [256](#),
[976](#), [1127](#), [1320](#), [1815](#), [1843](#), [1923](#), [2315](#), [2325](#), [2331](#),
[2337](#), [2345](#), [2353](#), [2360](#), [2884](#), [2945](#), [2946](#), [2956](#), [2963](#),
[2977](#), [2990](#), [3045](#), [3123](#), [3210](#), [3502](#), [3512](#), [3827](#)
`__enumext_list_arg_two_i:` [2850](#)
`__enumext_list_arg_two_ii:` [2850](#)
`__enumext_list_arg_two_iii:` [2850](#)
`__enumext_list_arg_two_iv:` [2850](#)
`__enumext_list_arg_two_v:` . [82](#), [2850](#), [3098](#), [3284](#)
`__enumext_list_arg_two_vii:` [2890](#), [3446](#)
`__enumext_list_arg_two_viii:` [2890](#), [3802](#)
`\l__enumext_listoffset_v_dim` [3161](#)
`\l__enumext_listparindent_vii_dim` [3637](#)
`\l__enumext_listparindent_viii_dim` ... [3981](#)
`__enumext_log_answer_vars:` . [32](#), [318](#), [326](#), [1930](#)
`__enumext_log_global_vars:` . [31](#), [318](#), [318](#), [1929](#)
`__enumext_make_label:` [35](#), [80](#), [82](#), [2788](#), [2788](#), [2879](#)
`\l__enumext_mark_answer_sym_tl` . [70](#), [122](#), [1970](#),
[2174](#), [2397](#), [2606](#), [2619](#), [3928](#)
`\l__enumext_mark_position_str` [122](#), [1974](#), [1975](#),
[1998](#), [1999](#), [2172](#)
`\l__enumext_mark_ref_sym_tl` .. [122](#), [1984](#), [2295](#),
[2558](#)

__enumext_mini_addvspace: . . 49, 87, 1149, 1149, 3020
 __enumext_mini_addvspace_vii: 52, 1302, 1302, 3404
 __enumext_mini_addvspace_viii: 52, 1302, 1308, 3762
 __enumext_mini_env* 1043
 __enumext_mini_right_cmd:n . 52, 53, 1328, 1330, 1330
 __enumext_mini_set_vskip: . 47, 1050, 1050, 1151
 __enumext_mini_set_vskip_vii: 51, 1245, 1245, 1304
 __enumext_mini_set_vskip_viii: 51, 1245, 1267, 1310
 __enumext_minipage:w . . 32, 339, 341, 1045, 3315, 3636, 3980
 \l__enumext_minipage_active_v_bool . . 90, 91, 3142, 3167, 3180, 3188
 \g__enumext_minipage_active_vii_bool . . . 96, 3415, 3420, 3432
 \l__enumext_minipage_active_vii_bool . 3400, 3411
 \g__enumext_minipage_active_viii_bool 3773, 3778, 3790
 \l__enumext_minipage_active_viii_bool 3758, 3769
 \g__enumext_minipage_active_X_bool . . . 155
 \l__enumext_minipage_active_X_bool 74
 \g__enumext_minipage_after_skip 74, 1249, 1261, 3430, 3788
 \l__enumext_minipage_after_skip 47, 49, 89, 91, 74, 1066, 1081, 1101, 1117, 1132, 1138, 1144, 1158, 1168, 1177, 1180, 1192, 1210, 1221, 1237, 1269, 1282, 1296, 3080, 3197
 \g__enumext_minipage_center_vii_bool . 3424, 3433
 \g__enumext_minipage_center_viii_bool 3782, 3791
 \g__enumext_minipage_center_X_bool . . . 155
 \l__enumext_minipage_hsep_v_dim . . . 90, 3140
 \l__enumext_minipage_hsep_vii_dim 3398
 \l__enumext_minipage_hsep_viii_dim . . . 3756
 \l__enumext_minipage_left_skip 47, 90, 74, 1058, 1073, 1092, 1107, 1154, 1164, 1169, 1175, 1184, 1201, 1213, 1233, 1243, 1247, 1252, 1256, 1270, 1274, 1288, 1306, 1312
 \l__enumext_minipage_left_v_dim 90, 3138, 3146
 \l__enumext_minipage_left_vii_dim 3394, 3406
 \l__enumext_minipage_left_viii_dim 3752, 3764
 \l__enumext_minipage_left_X_dim 74
 \g__enumext_minipage_right_skip 74, 1248, 1253, 1257, 3423, 3781
 \l__enumext_minipage_right_skip . 47, 74, 1062, 1077, 1097, 1112, 1170, 1176, 1188, 1206, 1217, 1271, 1278, 1292, 1340, 1357
 \l__enumext_minipage_right_v_dim . . 90, 1351, 1356, 3136, 3140
 \g__enumext_minipage_right_vii_dim 96, 3402, 3422, 3435
 \l__enumext_minipage_right_vii_dim 96, 3392, 3397, 3403
 \g__enumext_minipage_right_viii_dim . . 3760, 3780, 3793
 \l__enumext_minipage_right_viii_dim . . 3750, 3755, 3761
 \g__enumext_minipage_right_X_dim 155
 \g__enumext_minipage_right_X_skip 155
 \g__enumext_minipage_stat_int . 87, 90, 74, 1345, 1362, 3019, 3073, 3078, 3143, 3190, 3195
 \g__enumext_miniright_code_vii_tl . 97, 3428, 3434
 \g__enumext_miniright_code_viii_tl 3786, 3792
 \g__enumext_miniright_code_X_tl 155
 __enumext_multi_addvspace: . . . 46, 88, 998, 998, 3051
 __enumext_multi_set_vskip: . 45, 962, 962, 1000
 \l__enumext_multicols_above_ii_skip . . . 981
 \l__enumext_multicols_above_iii_skip . . . 987
 \l__enumext_multicols_above_iv_skip . . . 993
 \l__enumext_multicols_above_v_skip 1017, 1031, 1041
 \l__enumext_multicols_above_X_skip 68
 \l__enumext_multicols_below_v_skip 1021, 1035, 3182
 \l__enumext_multicols_below_X_skip 68
 __enumext_multicols_start: . 87, 88, 3025, 3027, 3027
 __enumext_multicols_stop: 88, 1335, 3057, 3057, 3082
 __enumext_newlabel:nn 28, 33, 74, 382, 382, 2371, 2517
 \l__enumext_newlabel_arg_one_tl 28, 33, 74, 76, 144, 2294, 2364, 2372, 2506, 2518, 2556
 \l__enumext_newlabel_arg_two_tl 28, 33, 73, 144, 2318, 2328, 2342, 2358, 2373, 2493, 2498, 2503, 2519
 __enumext_parse_keys:n . . . 56, 2926, 2951, 2951
 __enumext_parse_keys_vii:n 56, 3441, 3472, 3472
 __enumext_parse_keys_viii:n . 3798, 3832, 3832
 __enumext_parse_save_key:n 67, 2038, 2043, 2043
 __enumext_parse_save_key_vii:n 67, 2033, 2043, 2051
 __enumext_parse_serie:n 98
 __enumext_parse_series:n . . 56, 86, 1519, 1519, 2959, 3478
 __enumext_parse_store_keys:n 86
 \l__enumext_parsep_i_skip 979, 981, 1130, 1178
 \l__enumext_parsep_ii_skip 985, 987, 1136
 \l__enumext_parsep_iii_skip . . . 991, 993, 1142
 \l__enumext_parsep_vii_skip 3638
 \l__enumext_parsep_viii_skip 3982
 \l__enumext_partopsep_v_skip . 1033, 1037, 1204, 1208, 1215, 1219, 1235, 1239
 \l__enumext_partopsep_viii_skip 1280
 __enumext_phantomsection: 33, 346, 375, 379, 395
 __enumext_print_footnote: . . . 2653, 2676, 3656, 4007
 __enumext_print_keyans_box:NN 70, 2166, 2166, 2179, 2384, 2598, 2633, 3924, 3939
 \l__enumext_print_keyans_i_tl 4068, 4090
 \l__enumext_print_keyans_ii_tl . . . 4072, 4091
 \l__enumext_print_keyans_iii_tl . . . 4076, 4092
 \l__enumext_print_keyans_iv_tl . . . 4080, 4093
 \l__enumext_print_keyans_starred_tl 109, 110, 112, 4064, 4112
 \l__enumext_print_keyans_vii_tl 109, 4084, 4094
 \l__enumext_print_keyans_X_tl 112
 __enumext_printkeyans:nnn 110, 4095, 4098, 4098
 __enumext_redefine_item: . 81, 2727, 2727, 2878
 \l__enumext_ref_key_arg_tl 36, 38, 203, 532, 533,

546, 577, 580, 591, 597, 608, 649, 650, 661
 \l__enumext_ref_the_count_tl . 36, 38, 539, 542,
 545, 585, 587, 590, 602, 604, 607, 655, 657, 660
 __enumext_regex_counter_style: . . 29, 36, 198,
 198, 540, 586, 603, 656
 __enumext_register_counter_style:Nn . . 414,
 414, 419, 420, 421, 422, 423
 __enumext_remove_extra_parsep_vii: . . 3456,
 3665, 3665
 __enumext_remove_extra_parsep_viii: . 3812,
 4017, 4017
 __enumext_renew_footnote: . . . 2653, 2657, 3608,
 3961
 \l__enumext_renew_the_count_v_tl 658, 667, 669
 \l__enumext_renew_the_count_vii_tl 588, 617,
 619
 \l__enumext_renew_the_count_viii_tl 605, 624,
 626
 \l__enumext_renew_the_count_X_tl 38
 __enumext_reset_global_bool: . . 294, 297, 306
 __enumext_reset_global_int: . . . 294, 296, 300
 __enumext_reset_global_tl: . . . 294, 298, 312
 __enumext_reset_global_vars: . 31, 65, 294, 294,
 1937
 \l__enumext_resume_active_bool 56, 59, 49, 1523,
 1643
 __enumext_resume_counter: . . 58, 59, 1641, 1647,
 1654
 __enumext_resume_counter:n . 56, 59, 1612, 1617,
 1641, 1641, 1711, 1719
 __enumext_resume_counter_save_ans: . . 59, 60,
 1641, 1652, 1684
 __enumext_resume_counter_series: . 59, 1641,
 1650, 1667
 \g__enumext_resume_int . . . 49, 1564, 1658, 1659
 __enumext_resume_last:n 56, 57, 1519, 1525, 1538
 \l__enumext_resume_name_tl 49, 1560, 1568, 1571,
 1587, 1595, 1598, 1644, 1645, 1673, 1680
 __enumext_resume_save_counter: 57, 1551, 1551,
 3087, 3496
 __enumext_resume_series:n . 58, 1483, 1608, 1608
 __enumext_resume_starred: . 60, 1484, 1705, 1705
 \g__enumext_resume_vii_int . . 98, 49, 1591, 1663,
 1664
 __enumext_safe_exec: 86, 2925, 2942, 2942
 __enumext_safe_exec_vii: . . . 3440, 3461, 3461
 __enumext_safe_exec_viii: . . . 3797, 3817, 3817
 \l__enumext_series_name_tl 59
 \l__enumext_series_str . 57, 86, 1481, 1521, 1529,
 1530, 1532, 1534, 1555, 1558, 1562, 1582, 1585, 1589,
 2955, 3476
 __enumext_set_error:nn 4184, 4194, 4196
 __enumext_set_parse:n 4167, 4184, 4184
 \l__enumext_setkey_tmpa_int . . . 107, 4160, 4164
 \l__enumext_setkey_tmpa_seq . . . 107, 4158, 4168,
 4174, 4176, 4178, 4191
 \l__enumext_setkey_tmpa_tl 107, 4166, 4170
 \l__enumext_setkey_tmpb_seq . . . 107, 4159, 4162,
 4166, 4167
 \l__enumext_setkey_tmpb_tl 107, 4186, 4188, 4189
 \l__enumext_show_answer_bool . 122, 1978, 2002,
 2391, 2573, 2587, 3234, 3922
 __enumext_show_length:nnn . . 42, 206, 206, 4260,
 4261, 4262, 4263, 4264, 4265, 4266, 4267, 4268, 4269,
 4275, 4276, 4277, 4278, 4279, 4280, 4281, 4282, 4283,
 4284
 \l__enumext_show_position_bool 122, 1981, 2005,
 2395, 2577, 2588, 3235, 3926
 \g__enumext_standar_bool 30, 86, 26, 219, 222, 240,
 309, 1553, 1618, 1630, 1656, 1669, 1707, 1842, 1855
 \l__enumext_standar_bool . 86, 89, 26, 2323, 2336,
 2352, 2948, 3086
 \l__enumext_standar_first_bool 30, 86, 26, 245,
 1540, 1687, 1749, 1756
 __enumext_standar_item_vii:w . 99, 3536, 3538,
 3538
 __enumext_standar_item_viii:w 106, 3869, 3871,
 3871
 __enumext_standar_ref: 37, 530, 550, 2880
 __enumext_standar_ref:n 36, 522, 530, 530
 \g__enumext_standar_series_tl . 49, 1542, 1543,
 1709, 1712
 \g__enumext_starred_bool 30, 97, 98, 26, 228, 231,
 254, 310, 1580, 1623, 1634, 1661, 1676, 1715, 1820,
 1861, 2314, 2324, 2354, 2487, 2974, 2987, 3436
 \l__enumext_starred_bool . 97, 98, 26, 2249, 2255,
 2339, 2380, 3469, 3495
 __enumext_starred_columns_set_vii: . . 3322,
 3322, 3449
 __enumext_starred_columns_set_viii: . 3680,
 3680, 3805
 \l__enumext_starred_first_bool . . 30, 26, 259,
 1545, 1696, 1749, 1756
 __enumext_starred_item:nn . . . 2704, 2704, 2733
 __enumext_starred_item_exec: . 107, 3914, 3914,
 3965
 __enumext_starred_item_vii:w . . 99, 100, 3535,
 3554, 3554
 __enumext_starred_item_vii_aux_i:w . . 3554,
 3559, 3562
 __enumext_starred_item_vii_aux_ii:w . 3554,
 3560, 3565, 3567
 __enumext_starred_item_vii_aux_iii:w 3554,
 3570, 3579
 __enumext_starred_item_viii:w 106, 3868, 3887,
 3887
 __enumext_starred_item_viii_aux_i:w . . 106,
 3887, 3892, 3895
 __enumext_starred_item_viii_aux_ii:w . 106,
 3887, 3893, 3907, 3909
 __enumext_starred_joined_item_vii:n . 95, 99,
 3341, 3341, 3533
 __enumext_starred_joined_item_viii:n . 102,
 106, 3699, 3699, 3866
 __enumext_starred_ref: 38, 575, 613, 2908
 __enumext_starred_ref:n 37, 569, 575, 575
 \g__enumext_starred_series_tl . 49, 1547, 1548,
 1717, 1720
 __enumext_start_from:NNn 39, 672, 672, 685, 707
 \l__enumext_start_i_int 1659, 1671, 1690
 __enumext_start_item_tmp_vii: 97, 3452, 3518,
 3518
 __enumext_start_item_tmp_viii: . . 104, 3808,
 3851, 3851
 __enumext_start_item_vii:w 99, 100, 3546, 3551,
 3576, 3583, 3585, 3585
 __enumext_start_item_viii:w . . 106, 3879, 3884,
 3912, 3942, 3942

```

\g__enumext_start_line_tl 30, 132, 247, 261, 315,
    1885, 1890, 1895, 1909, 1914, 1919
\__enumext_start_list:nn 32, 83, 94, 333, 335, 2929,
    3095, 3248, 3444, 3800
\__enumext_start_mini_vii: . 98, 3390, 3390, 3487
\__enumext_start_mini_viii: . . . 105, 3748, 3748,
    3843
\__enumext_start_save_ans_msg: 61, 1733, 1733,
    1758
\__enumext_start_store_level: . 86, 2928, 2968,
    2968
\__enumext_start_store_level_vii: . 98, 3443,
    3498, 3498
\l__enumext_start_vii_int . . . 1664, 1678, 1699
\l__enumext_start_X_int . . . . . 86, 702
\__enumext_stop_item_tmp_vii: 97, 99, 100, 3451,
    3455, 3520, 3587
\__enumext_stop_item_tmp_viii: 104, 105, 3807,
    3811, 3853, 3944
\__enumext_stop_item_vii: 100, 101, 3587, 3641,
    3641
\__enumext_stop_item_viii: 108, 3944, 3992, 3992
\__enumext_stop_list: . . 32, 333, 336, 2938, 3105,
    3261, 3457, 3814
\__enumext_stop_mini_vii: 96, 98, 3409, 3409, 3491
\__enumext_stop_mini_viii: 105, 3748, 3767, 3847
\__enumext_stop_save_ans_msg: . 61, 1733, 1738,
    1927
\__enumext_stop_store_level: . . 86, 2939, 2968,
    2997
\__enumext_stop_store_level_vii: . . 98, 3458,
    3498, 3508
\l__enumext_store_active_bool 26, 61, 86, 98, 98,
    1688, 1697, 1765, 2209, 2424, 2972, 2985, 3110, 3118,
    3206, 3265, 3500, 3510, 3826
\__enumext_store_active_keys:n 67, 2011, 2011,
    2965
\__enumext_store_active_keys_vii:n . . 67, 98,
    2011, 2021, 3479
\__enumext_store_addto_prop:n 68, 75, 2086, 2086,
    2094, 2236, 2468, 3917
\__enumext_store_addto_seq:n 68, 77, 2095, 2095,
    2099, 2106, 2120, 2128, 2137, 2155, 2163, 2298, 2561
\l__enumext_store_anskey_arg_tl 26, 71, 72, 98,
    2246, 2251, 2253, 2258, 2265, 2268, 2278, 2283, 2286,
    2292, 2298
\__enumext_store_anskey_code:nn . 70, 71, 2203,
    2234, 2234, 2418
\__enumext_store_anskey_safe_inner:n . . 2417
\__enumext_store_anskey_safe_outer: . . . . 71
\__enumext_store_anskey_show_left:n 74, 2241,
    2389, 2389
\__enumext_store_anskey_show_wrap:n 74, 2377,
    2377, 2393, 2408
\l__enumext_store_columns_break_bool . 2182,
    2248
\l__enumext_store_columns_join_int . . . . 98
\__enumext_store_internal_ref: . . 71, 72, 2239,
    2300, 2300
\l__enumext_store_item_join_int . . 2185, 2256,
    2260
\l__enumext_store_item_star_bool . 2187, 2263
\l__enumext_store_item_symbol_sep_dim 2192,
    2275, 2280
\l__enumext_store_item_symbol_tl . 2190, 2266,
    2270
\l__enumext_store_keyans_item_opt_sep_
    tl . . . . 1967, 2462, 2464, 2535, 2539, 3900, 3902
\l__enumext_store_keyans_item_opt_tl . . . 98
\l__enumext_store_keyans_label_tl 26, 75, 77,
    106, 98, 2451, 2454, 2457, 2464, 2466, 2468, 2525,
    2528, 2531, 2537, 2542, 2552, 2561, 3897, 3902, 3903,
    3916, 3917, 3919
\__enumext_store_level_close: . 69, 2100, 2124,
    3001
\__enumext_store_level_close_vii: 2131, 2159,
    3514
\__enumext_store_level_open: . . 69, 2100, 2100,
    2980, 2993
\__enumext_store_level_open_vii: . 2131, 2131,
    3504
\g__enumext_store_name_tl . 26, 89, 98, 314, 321,
    322, 323, 324, 1741, 1767, 1884, 1889, 1894, 1908,
    1913, 1918, 1925
\l__enumext_store_name_tl . 26, 61, 62, 98, 1574,
    1577, 1601, 1604, 1692, 1701, 1736, 1745, 1746, 1767,
    1768, 1770, 1771, 1773, 1775, 1776, 1778, 1780, 1781,
    1805, 2088, 2090, 2097, 2366, 2367, 2403, 2508, 2509,
    2612, 2625, 3934
\l__enumext_store_ref_key_bool 71, 1987, 2237,
    2289, 2472, 2549
\l__enumext_store_save_key_vii_bool . . 2023,
    2053
\l__enumext_store_save_key_vii_tl 2025, 2026,
    2054, 2055, 2135, 2145, 2151, 2155
\l__enumext_store_save_key_X_bool . . . . . 67
\l__enumext_store_save_key_X_tl . . . . 67, 112
\l__enumext_store_upper_level_X_bool . . 112
\l__enumext_store_write_aux_file_tl 28, 74, 77,
    144, 2369, 2375, 2515, 2521
\__enumext_storing_exec: . . 61, 1743, 1759, 1763
\__enumext_storing_set:n . . 61, 1728, 1743, 1743
\l__enumext_the_counter_v_tl . . . . . 657
\l__enumext_the_counter_vii_tl . . . . . 587
\l__enumext_the_counter_viii_tl . . . . . 604
\l__enumext_the_counter_X_tl . . . . . 38
\__enumext_tmp:n 33, 37, 42, 48, 60, 67, 68, 73, 80, 85,
    86, 97, 113, 121, 147, 151, 155, 174, 785, 789, 1477,
    1488, 1724, 1732, 1784, 1802, 1957, 1992, 1993, 2010,
    2029, 2042, 2302, 2309, 2310, 2331, 2345, 2348, 2360,
    2474, 2481, 2850, 2889, 2890, 2922
\__enumext_tmp:nn 446, 467, 468, 496, 497, 509, 702,
    721, 766, 784, 842, 850, 851, 865, 930, 946, 947, 961,
    1366, 1382, 2637, 2652
\__enumext_tmp:nnn 510, 526, 527, 528, 529, 557, 573,
    574
\__enumext_tmp:nnnnn 722, 747, 750, 753, 755, 757,
    760, 763
\__enumext_tmp:w . . . . . 4044, 4047
\l__enumext_tmpa_vii_int . . . . . 3332, 3335
\l__enumext_tmpa_viii_int . . . . . 3690, 3693
\l__enumext_tmpa_X_int . . . . . 155
\l__enumext_topsep_v_skip 1019, 1023, 1173, 1186,
    1194, 1199, 1219, 1223, 3264, 3296
\l__enumext_topsep_vii_skip . . 1250, 1259, 1263
\l__enumext_topsep_viii_skip . 1272, 1294, 1298
\l__enumext_vspace_a_star_v_bool . . . . 1415
\l__enumext_vspace_a_star_vii_bool . . . 1437
\l__enumext_vspace_a_star_viii_bool . . 1448

```


<code>\l__enumext_vspace_a_star_X_bool</code>	86
<code>__enumext_vspace_above:</code>	54, 1383, 1383, 3006
<code>__enumext_vspace_above_v:</code>	54, 1411, 1411, 3134
<code>\l__enumext_vspace_above_v_skip</code>	1413, 1417, 1419
<code>__enumext_vspace_above_vii:</code>	55, 1433, 1433, 3484
<code>\l__enumext_vspace_above_vii_skip</code>	1435, 1439, 1441
<code>__enumext_vspace_above_viii:</code>	55, 1433, 1444, 3841
<code>\l__enumext_vspace_above_viii_skip</code>	1446, 1450, 1452
<code>\l__enumext_vspace_b_star_v_bool</code>	1426
<code>\l__enumext_vspace_b_star_vii_bool</code>	1459
<code>\l__enumext_vspace_b_star_viii_bool</code>	1470
<code>\l__enumext_vspace_b_star_X_bool</code>	86
<code>__enumext_vspace_below:</code>	54, 1397, 1397, 3085
<code>__enumext_vspace_below_v:</code>	54, 1422, 1422, 3202
<code>\l__enumext_vspace_below_v_skip</code>	1424, 1428, 1430
<code>__enumext_vspace_below_vii:</code>	55, 1455, 1455, 3494
<code>\l__enumext_vspace_below_vii_skip</code>	1457, 1461, 1463
<code>__enumext_vspace_below_viii:</code>	55, 1455, 1466, 3849
<code>\l__enumext_vspace_below_viii_skip</code>	1468, 1472, 1474
<code>__enumext_widest_from:nNNn</code>	39, 686, 686, 701, 713
<code>\g__enumext_widest_label_tl</code>	25, 34, 56, 434, 438, 442
<code>\l__enumext_wrap_label_opt_v_bool</code>	2746
<code>\l__enumext_wrap_label_opt_vii_bool</code>	99, 3545
<code>\l__enumext_wrap_label_opt_viii_bool</code>	106, 3878
<code>\l__enumext_wrap_label_opt_X_bool</code>	86
<code>\l__enumext_wrap_label_v_bool</code>	2742, 2746, 2754, 2810
<code>\l__enumext_wrap_label_vii_bool</code>	99, 3544, 3549, 3557, 3625
<code>\l__enumext_wrap_label_viii_bool</code>	106, 3877, 3882, 3890, 3969
<code>\l__enumext_wrap_label_X_bool</code>	86
<code>__enumext_wrapper_label_v:n</code>	2812, 3243
<code>__enumext_wrapper_label_vii:n</code>	3628
<code>__enumext_wrapper_label_viii:n</code>	3972
<code>__enumext_zero_parsep:</code>	49, 1070, 1125, 1125
<code>enumext*</code>	5, 3438
<code>enumXi</code>	406
<code>enumXii</code>	406
<code>enumXiii</code>	406
<code>enumXiv</code>	406
<code>enumXv</code>	406
<code>enumXvi</code>	406
<code>enumXvii</code>	406
<code>enumXviii</code>	406
Environments provide by enumext :	
<code>enumext*</code>	24, 25, 27-30, 34, 37, 38, 41, 42, 44, 45, 51, 52, 55-58, 60-63, 65-73, 76, 79, 85, 86, 97, 98, 100, 102, 103, 105, 107, 109, 110, 113, 115
<code>enumext</code>	24, 25, 27, 29, 30, 34-38, 40-58, 60-63, 65-71, 73, 76, 79-86, 89, 93, 95, 96, 98, 109, 111, 113, 114

<code>keyans*</code>	24-30, 34, 37-39, 41, 42, 44, 45, 51, 52, 55, 61, 62, 65, 66, 68, 76, 79, 85, 104, 105, 113, 115
<code>keyanspic</code>	24-27, 30, 34, 35, 38, 52, 61, 62, 65, 68, 75-78, 92-94, 115
<code>keyans</code>	24-27, 29, 30, 34, 35, 38, 40-44, 46, 50, 52-54, 61, 62, 65, 66, 68, 75-78, 82-84, 89, 90, 92-94, 96, 105, 113, 115
Environments:	
<code>list</code>	29, 32, 83, 85
<code>lrbox</code>	95, 101, 107, 108
<code>minipage</code>	29, 32, 45, 47, 92-95, 101, 108
<code>multicols</code>	45-48, 52, 87-91
exp commands:	
<code>\exp_after:wN</code>	4047
<code>\exp_args:Ne</code>	2962, 4035
<code>\exp_not:N</code>	46, 437, 545, 590, 607, 660, 799, 813, 814, 825, 826, 837, 838, 2294, 2400, 2401, 2554, 2609, 2610, 2622, 2623, 3931, 3932, 4044
<code>\exp_not:n</code>	249, 263, 275, 282, 289, 545, 546, 590, 591, 607, 608, 660, 661, 800, 1505, 1517, 2072, 2084, 2260, 2270, 2280, 2294, 2295, 2372, 2518, 2556, 2558
F	
<code>\fbox</code>	1962
file commands:	
<code>\file_input_stop:</code>	4429
<code>first</code>	851
<code>font</code>	446
<code>\footnote</code>	79
<code>\footnote</code>	79, 2661
<code>\footnotemark</code>	2671
<code>\footnotesize</code>	2401, 2610, 2623, 3932
<code>\footnotetext</code>	2655
G	
<code>\getkeyans</code>	14, 109, 4033
group commands:	
<code>\group_begin:</code>	2198, 2399, 2608, 2621, 3604, 3623, 3930, 3957, 3967, 4055, 4089
<code>\group_end:</code>	2205, 2406, 2615, 2628, 3633, 3645, 3937, 3977, 3996, 4057, 4096
H	
<code>\hbadness</code>	3652, 4003
hbox commands:	
<code>\hbox_set:Nn</code>	426
<code>\hfill</code>	476, 480, 485, 486, 1337, 1355, 2294, 2554, 3414, 3772
hook commands:	
<code>\hook_gput_code:nnn</code>	9, 182, 186, 344
<code>\hook_gset_rule:nnnn</code>	345
<code>\hspace</code>	3663, 4015
<code>\hyperlink</code>	72, 77
<code>\hyperlink</code>	2294, 2554
<code>\hypertarget</code>	33
<code>\hypertarget</code>	374
I	
<code>\IfHyperBoolean</code>	352
<code>\IfPackageLoadedTF</code>	11, 348, 362
<code>\ignorespaces</code>	802
<code>\inputlineno</code>	249, 263, 275, 282, 289
int commands:	
<code>\int_add:Nn</code>	3374, 3732
<code>\int_case:nn</code>	976, 1127, 1815, 1837, 1875, 1899
<code>\int_compare:nNTF</code>	578, 595, 615, 622, 1052, 1171, 1316, 1320, 1324, 1923, 1942, 1948, 2213, 2217, 2229,

2428, 2432, 2444, 2452, 2491, 2496, 2501, 2526, 2604, 2946, 2956, 2977, 2990, 3029, 3045, 3059, 3073, 3119, 3123, 3152, 3177, 3190, 3210, 3214, 3270, 3344, 3354, 3370, 3465, 3502, 3512, 3658, 3667, 3702, 3712, 3728, 3820, 3827, 4009, 4019, 4164	
\int_compare_p:nNn . . .	220, 229, 241, 242, 255, 256, 1821, 1843, 2256, 2315, 2325, 2337, 2338, 2353, 2355
\int_decr:N	3373, 3731
\int_eval:n	331, 2090, 2367, 2401, 2509, 2610, 2623, 2865, 2907, 3362, 3720, 3932
\int_from_alph:n	680, 694
\int_from_roman:n	682, 696
\int_gadd:Nn	3375, 3733
\int_gdecr:N	1824, 1828, 1831, 1834, 1846
\int_gincr:N	1658, 1663, 2201, 2416, 2564, 2692, 2722, 2760, 3019, 3143, 3232, 3522, 3600, 3855, 3921
\int_gset:Nn	1868, 2669
\int_gset_eq:NN	1557, 1564, 1570, 1576, 1584, 1591, 1597, 1603, 2666
\int_gzero:N	302, 303, 304, 1345, 1362, 1954, 3078, 3195, 3676, 4030
\int_if_exist:NTF	1532, 1568, 1574, 1595, 1601, 1778
\int_incr:N	2228, 2443, 2945, 3114, 3269, 3464, 3521, 3819, 3854
\int_mod:nn	3669, 4021
\int_new:N	20, 21, 22, 23, 24, 25, 49, 50, 74, 90, 102, 109, 129, 130, 138, 139, 140, 141, 152, 158, 159, 160, 161, 162, 1534, 1781
\int_set:Nn	676, 680, 682, 1671, 1678, 1690, 1699, 3309, 3310, 3332, 3343, 3349, 3365, 3652, 3690, 3701, 3707, 3723, 4003, 4160
\int_set_eq:NN	1659, 1664, 3372, 3730
\int_sign:n	1870
\int_step_function:nnN	2331, 2345, 2360
\int_step_inline:nnn	3311
\int_to_roman:n	190, 2311, 2349
\int_use:N	324, 329, 330, 1053, 1673, 1680, 1692, 1701, 2865, 2884, 2907, 2963, 3030, 3039, 3054, 3060, 3347, 3348, 3360, 3705, 3706, 3718
\int_zero:N	3661, 4013
\c_one_int	3332, 3351, 3357, 3363, 3367, 3370, 3690, 3709, 3715, 3721, 3725, 3728
\c_zero_int	2315, 2325, 2337, 2338, 2353, 2355, 3502, 3512, 3672, 4026
\item	32, 43, 44, 69, 80, 92, 94, 95, 97, 104
\item	80, 81, 99, 100, 105, 107, 337, 2108, 2114, 2139, 2147, 2253, 2528, 2531, 2729, 2764, 3450, 3452, 3806, 3808, 3919
\item*	5, 13, 65, 2762
item-pos*	2637
item-sym*	2637
\itemindent	25, 84
\itemindent	83
itemindent	766
\itemsep	93, 94
\itemsep	3285, 3291
\itemwidth	3339, 3383, 3387, 3697, 3741, 3745

K

keyans	13, 3090
keyans*	13, 3795
keyanspic	14, 3245
Keys for command provide by enumext:	
break-col	70, 71
item-join	70, 72

item-pos*	70, 72
item-star	70, 72
item-sym*	70, 72
Keys for environments provide by enumext:	
above*	26, 53–55
above	26, 53–55, 87, 90, 98, 105
after	42–44, 89, 92, 98, 105
align	26, 35, 82, 101
before*	42–44, 87, 98, 105
before	42–44, 90
below*	26, 53–55
below	26, 53–55, 89, 92, 98, 105
check-ans	26, 27, 29, 30, 60–65, 68, 77, 80, 81, 87, 89, 102, 114
columns-sep	44, 88, 91
columns	26, 44, 45, 47, 53, 88, 91
first	42–44, 101
font	35, 82, 101
item-pos*	79
item-sym*	25, 79, 80
item*-sep	80
itemindent	26, 41, 82, 101
itemsep	40, 85
labelsep	35, 80, 84, 101
labelwidth	34–39, 84
label	25, 34, 36, 39, 95
lisparindent	85
list-indent	25, 41, 94
list-offset	41
listparindent	41, 101
mark-ans	27, 66, 68, 74
mark-pos	66
mark-ref	27, 66, 68, 72
mini-env	26, 44, 47, 52, 53, 68, 79, 87, 90, 96, 98, 103, 105
mini-right*	26, 45, 68, 97
mini-right	26, 45, 52, 68, 97
mini-sep	26, 44, 68, 87, 90
minirigth*	29
minirigth	29
no-store	27, 60–62, 68, 70
noitemsep	40, 49
nosep	40, 49
parindent	85
parsep	40, 85, 101
partopsep	40
ref	25, 29, 36–38, 113
resume*	25, 55, 56, 60, 61, 68, 89
resume	25, 31, 55–61, 68, 89, 98
rightmargin	41
save-ans	26, 31, 56–62, 64–68, 70, 71, 75, 77, 81, 86, 89, 92, 98, 105, 107, 109, 113
save-key	27, 56, 67, 86
save-pos	68
save-ref	28, 33, 66, 68, 71, 72, 76, 77, 82, 107
save-sep	66, 68, 106
series	25, 55–60, 68, 86, 89
show-ans	27, 66, 68, 70, 71, 74, 81, 107
show-length	29, 42, 113
show-pos	27, 66, 70, 71, 74, 78, 81, 107
start	26, 29, 39, 40, 56
store-key	67
topsep	40
widest	25, 29, 39, 40
wrap-ans	66, 68, 70, 74
wrap-label*	35, 80, 82, 99, 101, 106

wrap-label	35, 82, 99, 101, 106
wrap-opt	66, 68
keys commands:	
\keys_define:nn	448, 470, 499, 512, 559, 630, 704, 724, 768, 787, 844, 853, 932, 949, 1368, 1479, 1726, 1786, 1959, 1995, 2031, 2036, 2180, 2639, 4060, 4129
\l_keys_key_str	4245
\keys_precompile:nnN	110, 4059, 4062, 4066, 4070, 4074, 4078, 4082
\keys_set:nn	462, 956, 1373, 1378, 1620, 1625, 1712, 1720, 2244, 2958, 2962, 3130, 3477, 3836, 4131, 4132, 4133, 4134, 4135, 4136, 4137, 4138, 4139, 4140, 4141, 4142, 4143, 4181
keyval commands:	
\keyval_parse:NNn	1493, 2061

L

label	510, 557, 630
Labels provide by enumext:	
\Alph*	34
\Roman*	34
\alph*	34
\arabic*	29, 34
\roman*	34
\labelsep	94
\labelsep	3286, 3289
labelsep	446
\labelwidth	34, 94
\labelwidth	3286, 3287
labelwidth	446
\leftmargin	25, 84
\leftmargin	83, 3286
legacy commands:	
\legacy_if:nTF	3588, 3591, 3945, 3948
\legacy_if_gset_false:n	1046
\legacy_if_set_false:n	3590, 3947
\legacy_if_set_true:n	3550, 3575, 3582, 3595, 3883, 3911, 3952
\linewidth	87, 90
\linewidth	3014, 3140, 3308, 3335, 3396, 3693, 3754
\list	32
\list	335
list-indent	766
list-offset	766
\listparindent	3288
listparindent	766
\lrbox	3605, 3958

M

\makebox	95
\makebox	2170, 2172, 2784, 3619, 3627, 3631, 3971, 3975
\makelabel	80, 82, 83, 95
\makelabel	82, 83, 2790, 2806
\makesavenoteenv	368
mark-ans	1957
mark-pos	1957, 1993
mark-ref	1957
mini-env	930
mini-sep	930
\minipage	32
\minipage	341
\miniright	10, 52, 1314, 3076, 3193
\miniright*	10
mode commands:	
\mode_if_vertical:TF	1001, 1029, 1152, 1231

\mode_leave_vertical:	799, 813, 825, 837, 2139, 2147, 2168, 2782, 3617
msg commands:	
\msg_error:nn	2226, 2231, 2441, 2446, 3121, 3125, 3212, 3272, 3467, 3822, 3829, 4144
\msg_error:nnn	535, 582, 599, 652, 1318, 1322, 1347, 1364, 1632, 1636, 1751, 4049, 4054, 4126, 4197
\msg_error:nnnn	2211, 2215, 2219, 2426, 2430, 2434, 3112, 3208, 3216, 4109
\msg_fatal:nn	2947
\msg_fatal:nnn	400
\msg_info:nnn	13, 16, 350, 364
\msg_line_context:	4217, 4221, 4225, 4249, 4254, 4259, 4274, 4289, 4293, 4297, 4301, 4305, 4309, 4315, 4321, 4327, 4341, 4345, 4350, 4354, 4359, 4363, 4367, 4372, 4377, 4381, 4386, 4391, 4395, 4400, 4404, 4409, 4414, 4419, 4423, 4427
\msg_log:nnn	1770, 1775, 1780
\msg_log:nnnnn	328, 1908, 1913, 1918
\msg_log:nnnnnn	320
\msg_new:nnn	4198, 4202, 4206, 4210, 4215, 4219, 4223, 4227, 4233, 4239, 4243, 4247, 4252, 4257, 4272, 4287, 4291, 4295, 4299, 4303, 4307, 4311, 4318, 4324, 4330, 4334, 4338, 4343, 4348, 4352, 4357, 4361, 4365, 4370, 4375, 4379, 4384, 4389, 4393, 4398, 4402, 4407, 4412, 4417, 4421, 4425
\msg_term:nnnn	1735, 1740, 2874, 2884, 2913, 2918
\msg_term:nnnnn	1889
\msg_warning:nn	3075, 3192
\msg_warning:nnnn	1945, 1951, 2822, 2827, 3346, 3359, 3704, 3717
\msg_warning:nnnnn	1884, 1894
\multicolsep	88, 91
\multicolsep	3044, 3165

N

\NeedsTeXFormat	3
\newcounter	403
\NewDocumentCommand	1314, 2195, 3204, 4033, 4087, 4151
\NewDocumentEnvironment	2411, 2923, 3090, 3245, 3438, 3795
\newlabel	33
\newlabel	386
no-store	1784
\noindent	97, 104
\noindent	3021, 3145, 3405, 3451, 3660, 3763, 3807, 4012
\nointerlineskip	3021, 3145, 3405, 3763
noitemsep	722
\nopagebreak	1012, 1040, 1163, 1242, 1305, 1311
\normalfont	2400, 2609, 2622, 3931
nosep	722

P

Packages:	
enumext	24, 36, 60, 83, 92, 112
enumitem	33, 34
expl3	95
footnotehyper	33
hyperref	28, 29, 32, 33, 72, 77, 100, 112
lua-visual-debug	47
multicol	24, 112
shortlst	95
\par	1012, 1040, 1163, 1242, 1305, 1311, 1340, 1357, 2379, 3065, 3080, 3182, 3197, 3320, 3423, 3430, 3660, 3674, 3781, 3788, 4012, 4028
\parindent	3637, 3981

\parsep	46, 49, 93, 94
\parsep	2140, 2148, 2904, 3285, 3292, 3297
parsep	722
\parskip	3638, 3982
\partopsep	94
\partopsep	2905, 3290
partopsep	722
peek commands:	
\peek_meaning:NTF	3527, 3541, 3558, 3569, 3860, 3874, 3891
\peek_meaning_remove:NTF	3534, 3867
\peek_remove_spaces:n	2768
\phantomsection	33
\phantomsection	375
prg commands:	
\prg_do_nothing:	379
\prg_new_protected_conditional:Npnn ...	192
\prg_replicate:nn	209
\prg_return_false:	196
\prg_return_true:	195
\printkeyans	15, 109, 4087
prop commands:	
\prop_count:N	322, 2090, 2367, 2403, 2509, 2612, 2625, 3934
\prop_gput_if_not_in:Nnn	2088
\prop_if_exist:NTF	1768, 4053
\prop_item:Nn	4056
\prop_new:N	1771
\ProvidesExplPackage	4
R	
\raggedcolumns	3053, 3171
\ref	72, 76
ref	510, 557, 630
\refstepcounter	3597, 3954
regex commands:	
\regex_match:nnTF	194, 679, 681, 693, 695
\regex_replace_once:nnN	202
\renewcommand	545, 590, 607, 660
\RenewDocumentCommand ...	2661, 2729, 2764, 2790, 2806
\RequirePackage	17
resume	1477
resume*	1477
rightmargin	766
\Roman	34, 39
\Roman	422
\roman	34, 39
\roman	423, 528, 4077
S	
save-ans	1724
save-key	2029
save-ref	1957
save-sep	1957
scan commands:	
\scan_stop:	94, 3299, 3450, 3806, 4044, 4047
seq commands:	
\seq_clear:N	4158
\seq_const_from_clist:Nn	4146
\seq_count:N	323, 3258, 4162
\seq_gclear:N	2659, 2660
\seq_gput_right:Nn	2097, 2672, 2673
\seq_if_empty:NTF	2678, 4102, 4176
\seq_if_exist:NTF	1773, 4100
\seq_if_in:NnTF	4107
\seq_item:Nn	3317
\seq_map_function:NN	4167
\seq_map_inline:Nn ..	4114, 4120, 4155, 4177, 4178
\seq_map_pairwise_function:NNN	2680
\seq_new:N	110, 111, 127, 153, 154, 1776
\seq_pop_left:NN	4166
\seq_put_right:Nn	3218, 4174, 4191
\seq_set_from_clist:Nn	4159
\seq_set_map_e:NNn	4168
\seq_show:N	4104
series	1477
\setcounter	690, 694, 696, 2865, 2907, 3263
\setenumext	6, 111, 4151
\setlength	2141, 2149
show-ans	1957, 1993
show-length	842
show-pos	1993
skip commands:	
\skip_add:Nn ..	981, 987, 993, 1003, 1007, 1031, 1035, 1132, 1138, 1144, 1154, 1158, 1180, 1233, 1237, 3285
\skip_eval:n	2140, 2148
\skip_gset:Nn	1253, 1257, 1261
\skip_gzero_new:N	1248, 1249
\skip_horizontal:N ..	814, 826, 838, 3620, 3634, 3978
\skip_horizontal:n ...	800, 2169, 2177, 2783, 2785, 3618, 3986
\skip_if_eq:nnTF ..	979, 985, 991, 1055, 1089, 1130, 1136, 1142, 1173, 1178, 1199, 1250, 1272, 1385, 1399, 1413, 1424, 1435, 1446, 1457, 1468
\skip_new:N	70, 71, 75, 76, 77, 78, 79, 131, 172
\skip_set:Nn ..	964, 968, 1017, 1021, 1058, 1062, 1066, 1073, 1077, 1081, 1092, 1097, 1101, 1107, 1112, 1117, 1175, 1176, 1177, 1184, 1188, 1192, 1201, 1206, 1210, 1213, 1217, 1221, 1252, 1256, 1274, 1278, 1282, 1288, 1292, 1296, 3279, 3293
\skip_set_eq:NN ..	2863, 2903, 2904, 3637, 3638, 3981, 3982
\skip_use:N ..	966, 970, 1005, 1009, 1013, 1033, 1037, 1056, 1075, 1084, 1090, 1095, 1099, 1110, 1114, 1115, 1120, 1156, 1160, 1186, 1386, 1390, 1393, 1400, 1404, 1407, 3065
\skip_zero:N	2905, 3044, 3165, 3290, 3291
\skip_zero_new:N ..	1168, 1169, 1170, 1247, 1269, 1270, 1271
\c_zero_skip ..	979, 985, 991, 1056, 1090, 1130, 1136, 1142, 1173, 1178, 1199, 1250, 1272, 1386, 1400, 1413, 1424, 1435, 1446, 1457, 1468
\small	4065, 4069, 4073, 4077, 4081, 4085
\star	2643
start	702
\stepcounter	2665, 3225
str commands:	
\c_backslash_str ..	4217, 4221, 4225, 4229, 4230, 4231, 4235, 4236, 4332, 4336, 4340, 4354, 4355, 4359, 4367, 4368, 4372, 4373, 4404, 4405, 4409, 4414, 4415
\c_colon_str	2366, 2508, 4044
\c_left_brace_str ..	4297, 4301, 4314, 4320, 4326
\c_right_brace_str ..	4297, 4301, 4314, 4320, 4326
\str_case:nn	214, 269
\str_case:nnTF	1500, 1509, 2068, 2076
\str_clear:N	2955, 3476
\str_count:n	209
\str_if_empty:NTF	1521, 1562, 1589
\str_if_eq:nnTF	2866, 2909
\str_if_in:nnTF	4040

<code>\str_new:N</code>	126, 167	<code>2253, 2258, 2265, 2268, 2278, 2283, 2286, 2292, 2318,</code>	
<code>\str_set:Nn</code> . . .	502, 503, 504, 1974, 1975, 1998, 1999	<code>2328, 2342, 2358, 2364, 2369, 2454, 2457, 2464, 2466,</code>	
<code>\string</code>	368	<code>2493, 2498, 2503, 2506, 2515, 2528, 2531, 2537, 2542,</code>	
<code>\strutbox</code> .	1060, 1064, 1068, 1079, 1083, 1094, 1103, 1109,	<code>2552, 3902, 3903</code>	
	1119, 1132, 1138, 1144, 1175, 1176, 1177, 1180, 1190,	<code>\tl_remove_all:Nn</code>	4188
	1194, 1203, 1210, 1215, 1223, 1252, 1253, 1256, 1263,	<code>\tl_remove_once:Nn</code>	2306, 2478
	1276, 1284, 1290, 1298, 3295	<code>\tl_replace_all:Nnn</code>	437
T		<code>\tl_reverse:N</code>	2305, 2307, 2477, 2479
T _E X and L ^A T _E X 2 _ε commands:			
<code>\@auxout</code>	384	<code>\tl_set:Nn</code> .	46, 273, 280, 287, 402, 476, 480, 485, 486,
<code>\@currenvir</code>	214, 269		532, 577, 649, 797, 811, 823, 835, 1644, 1745, 2016,
<code>\protected@write</code>	384		2026, 2047, 2055, 2397, 2571, 2606, 2619, 2708, 3905,
text commands:			
<code>\text_expand:n</code>	4036		3928, 4186
<code>\textasteriskcentered</code>	1971, 1985	<code>\tl_set_eq:NN</code>	443, 538, 541, 585, 587, 602, 604, 655,
<code>\thepage</code>	390		657, 2304, 2476, 2489, 2752, 2756, 3237, 3239
tl commands:			
<code>\c_space_tl</code>	2590, 4259, 4274, 4297, 4301	<code>\tl_to_str:n</code>	1615, 1621, 1626, 4036
<code>\tl_clear:N</code> . .	475, 481, 1955, 2015, 2025, 2046, 2054,	<code>\tl_trim_spaces:n</code>	433, 4174, 4186, 4192
	2246, 2451, 2525, 3897	<code>\tl_use:N</code> .	439, 442, 554, 619, 626, 669, 868, 872, 876,
<code>\tl_clear_new:N</code>	432		880, 884, 888, 892, 896, 900, 904, 908, 912, 916, 920,
<code>\tl_const:Nn</code>	38, 416		924, 928, 2174, 2311, 2319, 2330, 2344, 2349, 2361,
<code>\tl_gclear:N</code> .	314, 315, 316, 1542, 1547, 2801, 3434,		2695, 2701, 2725, 2743, 2747, 2755, 2792, 2793, 2800,
	3621, 3792		2808, 2809, 2815, 2930, 3096, 3242, 3428, 3624, 3635,
<code>\tl_gclear_new:N</code>	1529		3639, 3786, 3968, 3979, 3985, 3989, 4090, 4091, 4092,
<code>\tl_gput_right:Nn</code>	417		4093, 4094, 4112, 4170
<code>\tl_greplace_all:Nnn</code>	438	token commands:	
<code>\tl_gset:Nn</code> 246, 247, 260, 261, 1530, 1543, 1548, 1767,		<code>\token_to_str:N</code>	386
	3564	<code>\topsep</code>	2141, 2149
<code>\tl_gset_eq:NN</code>	434, 2710, 3614	<code>topsep</code>	<u>722</u>
<code>\tl_if_blank:nTF</code>	3612	<code>\typeout</code>	354, 357, 367, 368
<code>\tl_if_empty:NTF</code> .	533, 552, 580, 597, 617, 624, 650,	U	
	667, 1555, 1560, 1582, 1587, 1645, 1709, 1717, 1746,	<code>\u</code>	203
	1805, 1925, 2104, 2135, 2266, 2462, 2535, 2584, 2780,	use commands:	
	3900, 4189	<code>\use:N</code>	210, 2797, 2932
<code>\tl_if_empty:nTF</code>	1610, 2224, 2439	<code>\use:n</code>	1491, 2059, 4042
<code>\tl_if_exist:NTF</code>	1615	<code>\use_none:nn</code>	378
<code>\tl_if_novalue:nTF</code> . .	2242, 2459, 2533, 2569, 2663,	<code>\usecounter</code>	2864, 2906
	2688, 2706, 2711, 2740, 2953, 3256, 3474, 3834, 3898,	V	
	4153	<code>\value</code>	1558, 1564, 1571, 1577, 1585, 1591, 1598, 1604
<code>\tl_map_inline:Nn</code>	200, 435	<code>\vspace</code>	1047, 1390, 1393, 1404, 1407, 1417, 1419, 1428, 1430,
<code>\tl_new:N</code> 35, 40, 41, 44, 45, 51, 53, 54, 55, 57, 58, 91, 92,			1439, 1441, 1450, 1452, 1461, 1463, 1472, 1474, 2140,
	93, 99, 100, 101, 103, 104, 105, 106, 107, 108, 112, 115,		2148, 3253, 3264, 3675, 4029
	117, 118, 124, 125, 135, 136, 137, 144, 145, 146, 149,	W	
	166, 169	<code>widest</code>	<u>702</u>
<code>\tl_put_left:Nn</code>	2112, 2145, 2251, 2596, 2631, 3916,	<code>wrap-ans</code>	<u>1957</u>
	3919	<code>wrap-label</code>	<u>446</u>
<code>\tl_put_right:Nn</code>	433, 543, 588, 605, 658, 2116, 2151,	<code>wrap-label*</code>	<u>446</u>