

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-03^{*}

©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 <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	12
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
2	The environments provided	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	13
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	14
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	15
3	The command <code>\setenumext</code>	6	6	Full examples	16
4	The keyval system	6	7	The way of non-enumerated lists	19
4.1	Keys for label and ref	6	8	References	21
4.2	Keys for spaces	7	9	Change history	21
4.2.1	Vertical spaces	7	10	Index of Documentation	22
4.2.2	Horizontal spaces	8	11	Implementation	24
4.3	Keys for add code	8	12	Index of Implementation	117
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 2024-06-03.

[†]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

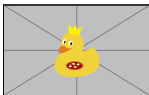
B) value


C) value


D) value


E) value

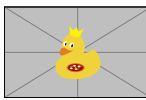
4. Question with image and label below:

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 / 129

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 multicol

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

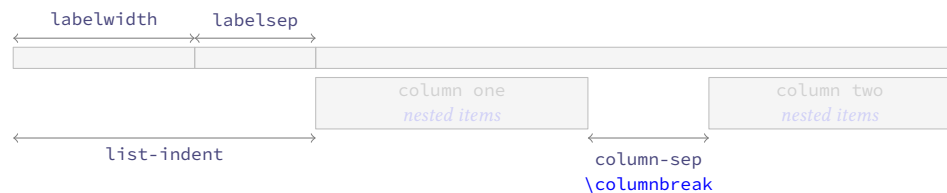


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “*non starred*” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. 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 second level.

i. This text is in the third level.
- A. This text is in the fourth level.

X This text is in the first 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.

X This text is in the first level.
2. This text is in the first level.

★ 3. This text is in the first level.

2.3 The command `\item*`

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

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

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

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

2.3.1 Keys for `\item*`

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

4 The keyval system

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

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

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

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

4.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2 Keys for spaces

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

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

4.2.1 Vertical spaces

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

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

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

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

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

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

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

The `\anskey` command is available for all levels of the `enumext` environment and the `enumext*` environment, but is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

5.2.1 Keys for `\anskey`

By default the {⟨*content*⟩} 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

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

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

5.3 The environment anskey*

anskey* `\begin{anskey*}[\langle key = val \rangle] \langle body content \rangle \end{anskey*}`

The environment `anskey*` takes a mandatory `\langle body content \rangle` and “stores” it in the *sequence* and *prop list* `\langle store name \rangle` set by `save-ans` key. By design the environment cannot be nested or passed *verbatim* in the body and it is assumed that each `\item` or `\item*` within the environment in which it is active it has a “single execution” 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 L^AT_EX will be used.

The `anskey*` environment uses the same `\langle keys \rangle` as the `\anskey` command and is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

Example

```
\begin{enumext}[save-ans=test,show-pos=true,start=5]
  \item* Text containing our instructions or questions.
    \begin{anskey*}
      \langle first answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          \langle second answer \rangle
        \end{anskey*}
    \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle third answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle fourth answer \rangle
    \end{anskey*}
\end{enumext}
```

- | | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> ★ 5. Text containing our instructions or questions. [5] first answer 6. Text containing our instructions or questions. (a) Question. [6] second answer | <ul style="list-style-type: none"> 7. Text containing our instructions or questions. [7] third answer 8. Text containing our instructions or questions. [8] fourth answer |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

5.4 The environments keyans and keyans*

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

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

The `\langle keys \rangle` set in the optional argument of the environment are the same (almost) as those of the `enumext` and `enumext*` environments and have higher precedence than those set by `\setenumext[\langle keyans \rangle]{\langle key`

= val}} or \setenumext[⟨keyans*⟩]{⟨key = val⟩}. If the optional argument is not passed or the ⟨keys⟩ are not set by \setenumext, the default values will be the same as the second level of the enumext environment with the difference in the ⟨label⟩ which will be set to label=\Alph*).

5.4.1 The \item* in keyans and keyans*

\item* \item*
\item* \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

5.5 The environment keyanspic

keyanspic \begin{keyanspic}[⟨n° above, n° below⟩]\anspic{⟨drawing⟩}\anspic*[⟨content⟩]{⟨drawing⟩}

The keyanspic is a “fake enumerated list” environment that which uses the \anspic command instead of \item. It is activated by the save-ans key and has the same settings as the keyans environment. It is intended for placing “drawings” or “tabular” with an in-line or above and below layout. A representation of the output can be seen in the figure 6.

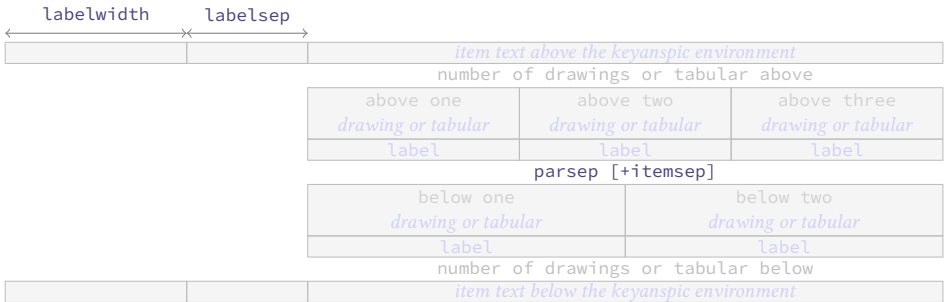


Figure 6: Representation of the keyanspic environment with optional argument [3,2] in enumext.

The optional argument determines the number drawings or tabular “above” and “below” within the environment. The vertical separation between “above” and “below” is controlled by the values set by

`parsep` and `itemsep` keys passed to `keyans` environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

5.5.1 The command `\anspic`

```
\anspic {<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]

5.6 Printing stored content

5.6.1 The command `\getkeyans`

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

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

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

5.6.2 The command `\printkeyans`

```
\printkeyans [<keys>] {<store name>}
\printkeyans* [<keys>] {<store name>}
```

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1]

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

(a)

~~TeX~~Xe 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:

```
$ pdftdetach -saveall enumext.pdf
```

and then you can use the excellent `arara`¹ tool to compile them.

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

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: 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.

- ☐ A 36 km/h.
☐ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

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

- ☐ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

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

- ☐ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.


1. B

2. A

3. B

4. A

Example 2

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

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

- ☐ A 36 km/h.
☒ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

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

- ☒ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

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

- ☐ A 36 km/h.
☒ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

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

- ☒ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

1. B

2. A

3. B

4. A

*

*

*

*

Example 3

A “simple multiple choice” test .

1. First type of questions

- ☐ A value
☐ B correct
☐ C value
☐ D value

2. Second type of questions

- I. $2\alpha + 2\delta = 90^\circ$
 II. $\alpha = \delta$
 III. $\angle EDF = 45^\circ$

- A I only

B II only

C I and II only

D I and III only

E I, II, and III
3. Third type of questions

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

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

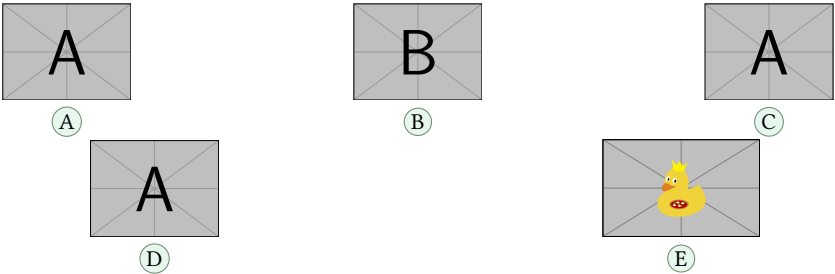
A value

B value

C value

D value

E value
4. Question with image and label below:



5. Question with image on left side:
- A value

B value

C value

D correct

E value



Test keys

1. B, $x = 5$

2. D

3. C, some note
4. E, A duck

5. D, other note

Example 4

A “simple worksheet” using ducks :) 🦆.

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

Factor $3x + 3y + 3z$
- The following questions need to be cuaqtified :)
- True False

(a) $\alpha > \delta$

(b) \LaTeX is cool?
- Related to Linux

(a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

- (b) Yes, dnf

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

ii. very good

iii. obsolete

Example 5

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

1

Which choice best describes what happens in the passage?

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

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

C) One character reminisces about choices she

2

has made over the years.

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

Which choice best describes what happens in the passage?

A) One character argues with another character

ter 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
 - Second level item
 - * Third level item
 - Fourth level item
 - First level item
- * First level item
 - ◊ Second level item
 - Third level item
 - ★ Fourth level item
 - * First level item

Fake description environment

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

Something A short one-line description.
This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Something A short one-line description.
This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Description indented by label

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

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

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

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[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 2024-06-03 – First public release.

10 Index of Documentation

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

C		Keys for environments provide by enumext:	
Document class:		above*	8
article	2	above	8
book	2	after	9, 10
exam	2	align	7, 20
letter	2	before*	9
report	2	before	9
\columnbreak	4, 12	below*	8
\columnsep	10	below	8
Commands provide by enumext:		check-ans	12
\anskey	4, 10–13	columns-sep	4, 10
\anspic*	4, 10–12, 14, 15	columns	4, 8, 10
\anspic	14, 15	first	9
\getkeyans	4, 12, 15	font	7
\item*	4–7, 10–14	item-pos*	5, 6
\itemwidth	5	item-sym*	5, 6
\item	5–7, 9, 10, 12, 13	itemindent	8
\miniright	4, 10	itemsep	8, 15
\printkeyans	4, 6, 11, 15	labelsep	3, 5–10, 12, 19, 20
\setenumext	4–7, 11, 13, 14, 16	labelwidth	3, 6, 7, 9, 10, 12, 19, 20
Counters defined by enumext:		labelwith	5
enumXiii	4	label	7, 9, 14, 19, 20
enumXii	4	list-indent	3, 9
enumXiv	4	list-offset	3, 9, 20
enumXi	4	listparindent	9
enumXviii	4	mark-ans	11, 12
enumXvii	4	mark-pos	12
enumXvi	4	mark-ref	11
enumXv	4	mini-env	4, 8, 10
E		mini-right*	6, 10
Environments provide by enumext:		mini-right	6, 10
anskey*	13	mini-sep	4, 10
enumext*	4–16	no-store	11–13
enumext	4–16, 19	noitemsep	8
keyans*	4–14	nosep	8, 19
keyanspic	4, 6, 8, 10–14, 19	parsep	8, 15
keyans	4–15, 19	partopsep	8
Environments:		ref	4, 7
enumerate	1, 3, 5, 20	resume*	6, 9, 10
figure	5	resume	6, 9, 10
list	3, 9, 20	rightmargin	8
minipage	3–5, 10, 20	save-ans	4, 6, 9–15
multicols	3, 4, 10	save-key	9, 11, 16
table	5	save-ref	4, 7, 11–13, 15
task	5	save-sep	11
F		series	6, 9, 10
\footnote	5	show-ans	11
I		show-length	7
\item	3, 5	show-pos	11, 12, 15
\itemsep	8	start	9
K		topsep	8
Keys for command provide by enumext:		widest	7
break-col	12	wrap-ans	11
item-join	12	wrap-label*	7, 20
item-pos*	12	wrap-label	7
item-star	12	wrap-opt	11
item-sym*	12	L	
Keys for environments provide by enumext:		\label	4
above*	8	Labels provide by enumext:	
above	8	\Alph*	7, 14
after	9, 10		
align	7, 20		
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, 15		
labelsep	3, 5–10, 12, 19, 20		
labelwidth	3, 6, 7, 9, 10, 12, 19, 20		
labelwith	5		
label	7, 9, 14, 19, 20		
list-indent	3, 9		
list-offset	3, 9, 20		
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	11–13		
noitemsep	8		
nosep	8, 19		
parsep	8, 15		
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, 16		
save-ref	4, 7, 11–13, 15		
save-sep	11		
series	6, 9, 10		
show-ans	11		
show-length	7		
show-pos	11, 12, 15		
start	9		
topsep	8		
widest	7		
wrap-ans	11		
wrap-label*	7, 20		
wrap-label	7		
wrap-opt	11		

\Roman*	7	l3keys	6
\alph*	7	l3prop	1, 20
\arabic*	7	l3seq	1, 20
\roman*	7	multicol	1, 2, 4, 20
\labelsep	3, 7	task	5, 6
\labelwidth	3, 7	xsim	2
\linewidth	10	\parsep	8
\listparindent	9	\partopsep	8
P		R	
Packages:		\raggedcolumns	4
enumerate	20	\ref	4
enumext	1–6, 14, 20	\rightmargin	8
enumitem	3–5, 9, 20		
footnotehyper	4, 5	T	
hyperref	4, 5, 11–13, 20	\topsep	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   {2024-06-03}
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.

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

```
\__enumext_level_int
\__enumext_level_h_int
\__enumext_anskey_level_int
\__enumext_keyans_level_int
\__enumext_keyans_level_h_int
\__enumext_keyans_pic_level_int
20 \int_new:N \__enumext_level_int
21 \int_new:N \__enumext_level_h_int
22 \int_new:N \__enumext_anskey_level_int
23 \int_new:N \__enumext_keyans_level_int
24 \int_new:N \__enumext_keyans_level_h_int
25 \int_new:N \__enumext_keyans_pic_level_int
```

(End of definition for `__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` (§11.9) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.12).

```

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 (§11.12).

```

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 (§11.28).

```

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` (§11.13) and `label` (§11.11) 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 (§11.15).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.32.1) 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.)

```

\__enumext_multicols_above_X_skip
\__enumext_multicols_below_X_skip

```

Internal variables used by `columns` key (§11.19).

```

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 (§11.20.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.18, §11.20).

```

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 (§11.13), 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 (§11.11). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```

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` (§11.25) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§11.30.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§11.35.1) 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.)

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

Internal variables used by the command `\setenumext` (§11.40).

```

107 \tl_new:N \l__enumext_setkey_tmpa_tl
108 \tl_new:N \l__enumext_setkey_tmpb_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_tmpb_seq

```

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

```

\l__enumext_print_keyans_starred_tl
\l__enumext_store_save_key_X_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_upper_level_X_bool

```

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§11.39) 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.)

```

\l__enumext_show_answer_bool
\l__enumext_show_position_bool
\l__enumext_mark_ref_sym_tl
\l__enumext_mark_answer_sym_tl
\l__enumext_mark_position_str

```

Internal variables for “*storage system*” mechanism used by `\anskey` (§11.25), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§11.24).

```

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.)

```

\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

```

Internal variables used by `keyanspic` environment (§11.35.2).

```

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.)

```

\l__enumext_check_answers_bool
\l__enumext_check_ans_key_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\g__enumext_start_line_tl
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int

```

Internal variables used by “*check answer*” mechanism (§11.23.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

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 (§11.8). 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.

```

\__enumext_reset_global_vars:
\__enumext_reset_global_int:
\__enumext_reset_global_bool:
\__enumext_reset_global_tl:
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 \__enumext_item_number_int }
330   { \int_use:N \__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 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` environment (*custom version* of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\vspace{0pt}` to maintain alignment on “top”. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§11.33) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.36)

```

344 \cs_new_protected:Nn \__enumext_internal_mini_page:
345 {
346   \int_compare:nNt { \l__enumext_level_int } = { 0 }
347   {
348     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
349     {
350       \__enumext_minipage:w [ t ] { ##1 }
351       \legacy_if_gset_false:n { @minipage }
352       \vspace { 0pt }
353     }
354   }
355 }

```



```

354         { \__enumext_endminipage: }
355     }
356 }

```

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

11.8 Compatibility with hyperref and footnotehyper

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

```

357 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
358 \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”.

```

359 \cs_new_protected:Nn \__enumext_after_hyperref:
360 {
361     \IfPackageLoadedTF { hyperref }
362     {
363         \msg_info:nnn { enumext } { package-load } { hyperref }
364         \bool_set_true:N \l__enumext_hyperref_bool
365         \IfHyperBoolean{hyperfootnotes}
366         {
367             \typeout{hyperfootnotes=true}
368             \bool_set_true:N \l__enumext_footnotes_key_bool
369         }
370         { \typeout{hyperfootnotes=false} }
371     }
372     { }

```

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`.

```

373 \bool_if:NT \l__enumext_footnotes_key_bool
374 {
375     \IfPackageLoadedTF { footnotehyper }
376     {
377         \msg_info:nnn { enumext } { package-load } { footnotehyper }
378     }
379     {
380         \typeout{No ~ footnotehyper ~ load}
381         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
382         \bool_set_false:N \l__enumext_footnotes_key_bool
383     }
384 }

```

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.

```

385 \bool_if:NTF \l__enumext_hyperref_bool
386 {
387     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
388     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
389 }
390 {
391     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
392     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
393 }
394 }

```

(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.

```

395 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
396 {
397   \protected@write \@auxout { }
398   {
399     \token_to_str:N \newlabel {#1}
400     {
401       {#2}
402       \bool_if:NT \l__enumext_hyperref_bool
403       { { \thepage } {#2} {#1} }
404       { }
405     }
406   }
407   \__enumext_hypertarget:nn {#1} { }
408   \__enumext_phantomsection:
409 }

```

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

11.9 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

```

410 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
411 {
412   \cs_if_exist:cTF { c@ #2 }
413   { \msg_fatal:nnn { enumext } { counters } { #2 } }
414   {
415     \tl_set:Nn #1 { #2 }
416     \newcounter { #2 }
417   }
418 }

```

(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      419 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     420 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    421 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii    422 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii   423 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXv      424 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXvi     425 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXviii   426 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

11.10 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```

427 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
428 {
429   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
430   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
431 }
432 \__enumext_register_counter_style:Nn \arabic { 0 }
433 \__enumext_register_counter_style:Nn \Alph { M }
434 \__enumext_register_counter_style:Nn \alph { m }
435 \__enumext_register_counter_style:Nn \Roman { VIII }
436 \__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.

```
437 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
438 {
439   \hbox_set:Nn \__enumext_label_width_by_box {#2}
440   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
441 }
442 \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.

```
443 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
444 {
445   \tl_clear_new:N #1
446   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
447   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
448   \tl_map_inline:Nn \g__enumext_counter_styles_tl
449   {
450     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
451     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
452     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
453   }
454   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
455   { \tl_use:N \g__enumext_widest_label_tl }
456   \tl_set_eq:cN { the #2 } #1
457 }
458 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

11.11 Setting keys associated with label

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

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

```
459 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
460 {
461   \keys_define:nn { enumext / #1 }
462   {
463     font      .tl_set:c    = { l__enumext_label_font_style_#2_tl },
464     font      .value_required:n = true,
465     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
466     labelsep  .initial:n   = {0.3333em},
467     labelsep  .value_required:n = true,
468     labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
469     labelwidth .value_required:n = true,
470     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
471     wrap-label .initial:n   = {##1},
472     wrap-label .value_required:n = true,
473     wrap-label* .code:n = {
474       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
475       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
476     },
477     wrap-label* .value_required:n = true,
478   }
479 }
480 \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.

```

481 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
482 {
483   \keys_define:nn { enumext / #1 }
484   {
485     align .choice:,
486     align / left .code:n =
487       {
488         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
489         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
490       },
491     align / right .code:n =
492       {
493         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
494         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
495       },
496     align / center .code:n =
497       {
498         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
499         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
500       },
501     align .initial:n = left,
502     align .value_required:n = true,
503   }
504 }
505 \clist_map_inline:nn
506 {
507   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
508 }
509 { \__enumext_tmp:nn #1 }

510 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
511 {
512   \keys_define:nn { enumext / #1 }
513   {
514     align .choice:,
515     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
516     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
517     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
518     align .initial:n = left,
519     align .value_required:n = true,
520   }
521 }
522 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

11.12 Setting label and ref keys

The implementation of the keys `label` and `ref` are part of the core of the package `enumext`, here the default values for $\langle label \rangle$, the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “label and ref” system.

11.12.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

523 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
524 {
525   \keys_define:nn { enumext / #1 }
526   {
527     label .code:n = {
528       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
529       { l__enumext_counter_#2_tl } {##1}
530       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
531       \l__enumext_current_widest_dim
532     },
533     label .initial:n = #3,
534     label .value_required:n = true,
535     ref .code:n = \__enumext_standar_ref:n {##1},
536     ref .value_required:n = true,
537   }

```

```

538   }
539   \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
540   \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
541   \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
542   \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

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

```

543 \cs_new_protected:Npn \__enumext_standar_ref:n #1
544 {
545   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
546   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
547   {
548     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
549   }
550   {
551     \tl_set_eq:Nc
552     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
553     \__enumext_regex_counter_style:
554     \tl_set_eq:Nc
555     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
556     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
557     {
558       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
559       { \exp_not:V \l__enumext_ref_key_arg_tl }
560     }
561   }
562 }

```

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

```

563 \cs_new_protected:Npn \__enumext_standar_ref:
564 {
565   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
566   {
567     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
568   }
569 }

```

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

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

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
570 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
571 {
572   \keys_define:nn { enumext / #1 }
573   {
574     label .code:n = {
575       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
576       { \l__enumext_counter_#2_tl } {##1}
577       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
578       \l__enumext_current_widest_dim
579     },
580     label .initial:n = #3,
581     label .value_required:n = true,
582     ref .code:n = \__enumext_starred_ref:n {##1},
583     ref .value_required:n = true,
584   }
585 }
586 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
587 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*. }

```

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

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

```

588 \cs_new_protected:Npn \__enumext_starred_ref:n #1
589 {
590   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
591   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
592   {
593     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
594     {
595       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
596     }
597     {
598       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
599       \__enumext_regex_counter_style:
600       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
601       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
602       {
603         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
604         { \exp_not:V \l__enumext_ref_key_arg_tl }
605       }
606     }
607   }
608   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
609   {
610     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
611     {
612       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
613     }
614     {
615       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
616       \__enumext_regex_counter_style:
617       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
618       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
619       {
620         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
621         { \exp_not:V \l__enumext_ref_key_arg_tl }
622       }
623     }
624   }
625 }

```

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.

```

626 \cs_new_protected:Npn \__enumext_starred_ref:
627 {
628   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
629   {
630     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
631     {
632       \tl_use:N \l__enumext_renew_the_count_vii_tl
633     }
634   }
635   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
636   {
637     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
638     {
639       \tl_use:N \l__enumext_renew_the_count_viii_tl
640     }
641   }
642 }

```

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

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

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

```

\l__enumext_label_v_tl 643 \keys_define:nn { enumext / keyans }
\l__enumext_label_vi_tl 644 {
645   label .code:n = {
646     \__enumext_label_style:cnv { \l__enumext_label_v_tl }
647     { \l__enumext_counter_v_tl } {#1}

```



```

648         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
649         \l__enumext_current_widest_dim
650         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
651         { \l__enumext_counter_vi_tl } {#1}
652         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
653         \l__enumext_current_widest_dim
654     },
655     label .initial:n = \Alph*),
656     label .value_required:n = true,
657     ref .code:n = \__enumext_keyans_ref:n {#1},
658     ref .value_required:n = true,
659 }

```

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:

```

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

```

660 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
661 {
662     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
663     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
664     {
665         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
666     }
667     {
668         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
669         \__enumext_regex_counter_style:
670         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
671         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
672         {
673             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
674             { \exp_not:V \l__enumext_ref_key_arg_tl }
675         }
676     }
677 }

```

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

```

678 \cs_new_protected:Nn \__enumext_keyans_ref:
679 {
680     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
681     {
682         \tl_use:N \l__enumext_renew_the_count_v_tl
683     }
684 }

```

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

11.13 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.

```

685 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
686 {
687     \__enumext_if_is_int:nTF { #3 }
688     {
689         \int_set:Nn #2 {#3}
690     }
691     {
692         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
693         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
694         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
695         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
696     }
697 }
698 \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: $\langle integer \text{ or } string \rangle$

The second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an $\langle integer \rangle$ or $\langle string \rangle$ 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.

```
699 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
700 {
701   \__enumext_if_is_int:nTF {#4}
702   {
703     \setcounter{enumX#1} { #4 }
704   }
705   {
706     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
707     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
708     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
709     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
710   }
711   \__enumext_label_width_by_box:cv
712   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
713 }
714 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

```
start
widest
```

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

```
\l__enumext_start_X_int
```

```
715 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
716 {
717   \keys_define:nn { enumext / #1 }
718   {
719     start .code:n = {
720       \__enumext_start_from:ccn
721       { \l__enumext_label_#2_tl }
722       { \l__enumext_start_#2_int } {##1}
723     },
724     start .initial:n = 1,
725     widest .code:n = {
726       \__enumext_widest_from:nccn {#2}
727       { \l__enumext_label_#2_tl }
728       { \l__enumext_labelwidth_#2_dim } {##1}
729     },
730     widest .value_required:n = true,
731     start .value_required:n = true,
732   }
733 }
734 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.14 Setting keys for vertical spaces

```
topsep
partopsep
parsep
noitemsep
nosep
```

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

```
735 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
736 {
737   \keys_define:nn { enumext / #1 }
738   {
739     topsep .skip_set:c = { \l__enumext_topsep_#2_skip },
740     topsep .initial:n = {#3},
741     topsep .value_required:n = true,
742     partopsep .skip_set:c = { \l__enumext_partopsep_#2_skip },
743     partopsep .initial:n = {#4},
744     partopsep .value_required:n = true,
745     parsep .skip_set:c = { \l__enumext_parsep_#2_skip },
746     parsep .initial:n = {#5},
```

```

747     parsep      .value_required:n = true,
748     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
749     itemsep     .initial:n = {#6},
750     itemsep     .value_required:n = true,
751     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
752     noitemsep   .value_forbidden:n = true,
753     nosepe     .meta:n = {
754         itemsep = 0pt, parsep= 0pt,
755         topsep = 0pt, partopsep = 0pt,
756     },
757     nosepe     .value_forbidden:n = true,
758 }
759 }

```

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

```

760 \__enumext_tmp:nnnnnn { level-1 } { i } { 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 { level-2 } { ii } { 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 }
766 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
767 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
768 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
769 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
770 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
771 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
772 { 2.0pt plus 1.0pt minus 1.0pt }
773 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
774 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
775 { 4.0pt plus 2.0pt minus 1.0pt }
776 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
777 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
778 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

11.15 Setting keys for horizontal spaces

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

```

779 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
780 {
781     \keys_define:nn { enumext / #1 }
782     {
783         itemindent      .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
784         itemindent      .value_required:n = true,
785         rightmargin     .dim_set:c = { l__enumext_rightmargin_#2_dim },
786         rightmargin     .value_required:n = true,
787         listparindent   .dim_set:c = { l__enumext_listparindent_#2_dim },
788         listparindent   .value_required:n = true,
789         list-offset     .dim_set:c = { l__enumext_listoffset_#2_dim },
790         list-offset     .value_required:n = true,
791         list-indent     .code:n =
792             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
793             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
794         list-indent     .value_required:n = true,
795     }
796 }
797 \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.

```

798 \cs_set_protected:Npn \__enumext_tmp:n #1
799 {
800     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
801 }
802 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

11.15.1 Functions for setting the fake itemindent

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

```

803 \cs_set_protected:Nn \__enumext_fake_item:
804 {
805   \dim_compare:nNnT
806     { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
807     >
808     { \c_zero_dim }
809   {
810     \tl_set:ce { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
811     {
812       \exp_not:N \mode_leave_vertical:
813       \exp_not:n { \skip_horizontal:n }
814       { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
815       \ignorespaces
816     }
817   }
818 }
819 \cs_set_protected:Nn \__enumext_keyans_fake_item:
820 {
821   \dim_compare:nNnT
822     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
823   {
824     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
825     {
826       \exp_not:N \mode_leave_vertical:
827       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
828     }
829   }
830 }
831 \cs_set_protected:Nn \__enumext_fake_item_vii:
832 {
833   \dim_compare:nNnT
834     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
835   {
836     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
837     {
838       \exp_not:N \mode_leave_vertical:
839       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
840     }
841   }
842 }
843 \cs_set_protected:Nn \__enumext_fake_item_viii:
844 {
845   \dim_compare:nNnT
846     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
847   {
848     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
849     {
850       \exp_not:N \mode_leave_vertical:
851       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
852     }
853   }
854 }

```

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

11.16 Setting show-length key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\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.

```

855 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
856 {
857   \keys_define:nn { enumext / #1 }
858   {
859     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },

```

```

860         show-length .initial:n = false,
861     }
862 }
863 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.17 Setting before, after and first keys

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

```

before
before*
after
first
864 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
865 {
866     \keys_define:nn { enumext / #1 }
867     {
868         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
869         before .value_required:n = true,
870         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
871         before* .value_required:n = true,
872         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
873         after .value_required:n = true,
874         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
875         first .value_required:n = true,
876     }
877 }
878 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.17.1 Functions for before, after and first keys in enumext

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.

```

879 \cs_new_protected:Nn \__enumext_before_args_exec:
880 {
881     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
882 }

```

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`.

```

883 \cs_new_protected:Nn \__enumext_before_keys_exec:
884 {
885     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
886 }

```

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

```

887 \cs_new_protected:Nn \__enumext_after_stop_list:
888 {
889     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
890 }

```

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`.

```

891 \cs_new_protected:Nn \__enumext_after_args_exec:
892 {
893     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
894 }

```

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

11.17.2 Functions for before, after and first keys in keyans

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.

```

895 \cs_new_protected:Nn \__enumext_before_args_exec_v:
896 {
897     \tl_use:N \l__enumext_before_starred_key_v_tl
898 }

```

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`.

```
899 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
900 {
901     \tl_use:N \l__enumext_before_no_starred_key_v_tl
902 }
```

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

```
903 \cs_new_protected:Nn \__enumext_after_stop_list_v:
904 {
905     \tl_use:N \l__enumext_after_stop_list_v_tl
906 }
```

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`.

```
907 \cs_new_protected:Nn \__enumext_after_args_exec_v:
908 {
909     \tl_use:N \l__enumext_after_list_args_v_tl
910 }
```

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

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

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

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

```
911 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
912 {
913     \tl_use:N \l__enumext_before_starred_key_vii_tl
914 }
915 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
916 {
917     \tl_use:N \l__enumext_before_starred_key_viii_tl
918 }
```

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`.

```
919 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
920 {
921     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
922 }
923 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
924 {
925     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
926 }
```

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

```
927 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
928 {
929     \tl_use:N \l__enumext_after_stop_list_vii_tl
930 }
931 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
932 {
933     \tl_use:N \l__enumext_after_stop_list_viii_tl
934 }
```

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`.

```
935 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
936 {
937     \tl_use:N \l__enumext_after_list_args_vii_tl
938 }
939 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
940 {
941     \tl_use:N \l__enumext_after_list_args_viii_tl
942 }
```

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

11.18 Setting keys for multicols and minipage

mini-env 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.

mini-sep

columns-sep Define and set `mini-env`, `mini-sep`, `columns-sep` and `columns` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

columns

```

943 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
944 {
945   \keys_define:nn { enumext / #1 }
946   {
947     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
948     mini-env .value_required:n = true,
949     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
950     mini-sep .initial:n = 0.3333em,
951     mini-sep .value_required:n = true,
952     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
953     columns-sep .value_required:n = true,
954     columns .int_set:c = { \__enumext_columns_#2_int },
955     columns .initial:n = 1,
956     columns .value_required:n = true,
957   }
958 }
959 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

For `enumext*` and `keyans*` environments the situation is a bit different, the command `\miniright` is not available, so we will add the keys `mini-right` and `mini-right*` to implement support for `minipage` environment.

```

960 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
961 {
962   \keys_define:nn { enumext / #1 }
963   {
964     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
965     mini-right .value_required:n = true,
966     mini-right* .code:n = {
967       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
968       \keys_set:nn { enumext / #1 } { miniright = {##1} }
969     },
970     mini-right* .value_required:n = true,
971   }
972 }
973 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.19 Adjustment of vertical spaces for multicols

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



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

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

🔗 I should make it clear that the implementation here is a “*bit questionable*”. At first glance doing `\multicolsep=\topsep` seemed right, but the results were not always as expected. An almost *imperceptible* detail is that in some cases the `\itemsep` values 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.

11.19.1 Adjustment of vertical spaces for multicols in enumext

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

We will set the default values taking into account that T_EX 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*.

```

974 \cs_new_protected:Nn \__enumext_multi_set_vskip:
975 {
976   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
977   {
978     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
979   }
980   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
981   {
982     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
983   }
984   \__enumext_add_pre_parsep:
985 }

```

(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*.

```

986 \cs_new_protected:Nn \__enumext_add_pre_parsep:
987 {
988   \int_case:nn { \l__enumext_level_int }
989   {
990     { 2 }{
991       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
992       {
993         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
994       }
995     }
996     { 3 }{
997       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
998       {
999         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1000       }
1001     }
1002     { 4 }{
1003       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1004       {
1005         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1006       }
1007     }
1008   }
1009 }

```

(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 T_EX is in *horizontal mode* or *vertical mode*.

```

1010 \cs_new_protected:Nn \__enumext_multi_addvspace:
1011 {
1012   \__enumext_multi_set_vskip:
1013   \mode_if_vertical:T
1014   {
1015     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1016     {
1017       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1018     }
1019     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1020     {
1021       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1022     }
1023   }
1024   \par\nopagebreak
1025   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1026 }

```

(End of definition for `__enumext_multi_addvspace:`.)

11.19.2 Adjustment of vertical spaces for multicol in keyans

`__enumext_keyans_multi_set_vskip:`
`__enumext_keyans_multi_addvspace:`

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

```

1027 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1028 {
1029   \skip_set:Nn \l__enumext_multicol_above_v_skip
1030   {
1031     \l__enumext_topsep_v_skip
1032   }
1033   \skip_set:Nn \l__enumext_multicol_below_v_skip
1034   {
1035     \l__enumext_topsep_v_skip
1036   }
1037 }
1038 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1039 {
1040   \__enumext_keyans_multi_set_vskip:
1041   \mode_if_vertical:T
1042   {
1043     \skip_add:Nn \l__enumext_multicol_above_v_skip
1044     {
1045       \skip_use:N \l__enumext_partopsep_v_skip
1046     }
1047     \skip_add:Nn \l__enumext_multicol_below_v_skip
1048     {
1049       \skip_use:N \l__enumext_partopsep_v_skip
1050     }
1051   }
1052   \par\nopagebreak
1053   \addvspace{ \l__enumext_multicol_above_v_skip }
1054 }

```

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

11.20 Adjustment of vertical spaces for minipage

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

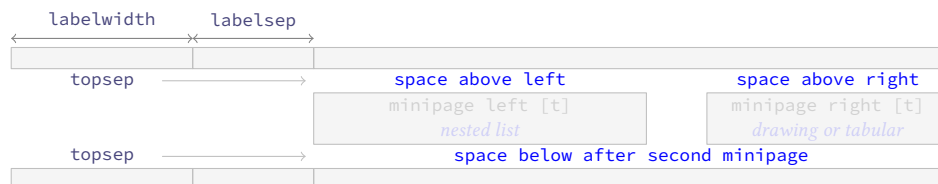


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

Here there are several complications that we must circumvent, the `minipage` environment eliminates the “top” spaces, the `multicol` 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 `multicol` it is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug` package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

11.20.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_mini_set_vskip:`

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

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

First determine if the `multicol` 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`.

```

1055 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1056 {
1057   \int_compare:nNnTF
1058     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1059     {

```

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.

```

1060   \skip_if_eq:nnTF
1061     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1062   {
1063     \skip_set:Nn \l__enumext_minipage_left_skip
1064       {
1065         -0.150\box_dp:N \strutbox
1066       }
1067     \skip_set:Nn \l__enumext_minipage_right_skip
1068       {
1069         0.695\box_dp:N \strutbox
1070       }
1071     \skip_set:Nn \l__enumext_minipage_after_skip
1072       {
1073         \box_dp:N \strutbox
1074       }
1075     \__enumext_zero_parsep:
1076   }
1077   {
1078     \skip_set:Nn \l__enumext_minipage_left_skip
1079       {
1080         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1081       }
1082     \skip_set:Nn \l__enumext_minipage_right_skip
1083       {
1084         0.695\box_dp:N \strutbox
1085       }
1086     \skip_set:Nn \l__enumext_minipage_after_skip
1087       {
1088         1.85\box_dp:N \strutbox
1089         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1090       }
1091   }
1092 }
1093 {

```

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.

```

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

```

```

1116         }
1117         \skip_set:Nn \l__enumext_minipage_right_skip
1118         {
1119             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1120             + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1121         }
1122         \skip_set:Nn \l__enumext_minipage_after_skip
1123         {
1124             0.325\box_dp:N \strutbox
1125             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1126         }
1127     }
1128 }
1129 }

```

(End of definition for `__enumext_mini_set_vskip:`)

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

```

1130 \cs_new_protected:Nn \__enumext_zero_parsep:
1131 {
1132     \int_case:nn { \l__enumext_level_int }
1133     {
1134         { 2 }{
1135             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1136             {
1137                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1138             }
1139         }
1140         { 3 }{
1141             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1142             {
1143                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1144             }
1145         }
1146         { 4 }{
1147             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1148             {
1149                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1150             }
1151         }
1152     }
1153 }

```

(End of definition for `__enumext_zero_parsep:`)

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

```

1154 \cs_new_protected:Nn \__enumext_mini_addvspace:
1155 {
1156     \__enumext_mini_set_vskip:
1157     \mode_if_vertical:T
1158     {
1159         \skip_add:Nn \l__enumext_minipage_left_skip
1160         {
1161             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1162         }
1163         \skip_add:Nn \l__enumext_minipage_after_skip
1164         {
1165             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1166         }
1167     }
1168     \par\nopagebreak
1169     \addvspace { \l__enumext_minipage_left_skip }
1170 }

```

(End of definition for `__enumext_mini_addvspace:`)

11.20.2 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`.

```

1171 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1172 {
1173   \skip_zero_new:N \l__enumext_minipage_after_skip
1174   \skip_zero_new:N \l__enumext_minipage_left_skip
1175   \skip_zero_new:N \l__enumext_minipage_right_skip
1176   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1177   {
1178     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1179     {
1180       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1181       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1182       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1183       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1184       {
1185         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1186       }
1187     }
1188     {
1189       \skip_set:Nn \l__enumext_minipage_left_skip
1190       {
1191         \skip_use:N \l__enumext_topsep_v_skip
1192       }
1193       \skip_set:Nn \l__enumext_minipage_right_skip
1194       {
1195         0.705\box_dp:N \strutbox
1196       }
1197       \skip_set:Nn \l__enumext_minipage_after_skip
1198       {
1199         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1200       }
1201     }
1202   }
1203   {
1204     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1205     {
1206       \skip_set:Nn \l__enumext_minipage_left_skip
1207       {
1208         0.5\box_dp:N \strutbox
1209         + \l__enumext_partopsep_v_skip
1210       }
1211       \skip_set:Nn \l__enumext_minipage_right_skip
1212       {
1213         \l__enumext_partopsep_v_skip
1214       }
1215       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1216     }
1217     {
1218       \skip_set:Nn \l__enumext_minipage_left_skip
1219       {
1220         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1221       }
1222       \skip_set:Nn \l__enumext_minipage_right_skip
1223       {
1224         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1225       }
1226       \skip_set:Nn \l__enumext_minipage_after_skip
1227       {
1228         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1229       }
1230     }
1231   }
1232 }

```

(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`.

```

1233 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1234 {
1235   \__enumext_keyans_mini_set_vskip:
1236   \mode_if_vertical:T
1237   {
1238     \skip_add:Nn \l__enumext_minipage_left_skip
1239     {
1240       \l__enumext_partopsep_v_skip
1241     }
1242     \skip_add:Nn \l__enumext_minipage_after_skip
1243     {
1244       \l__enumext_partopsep_v_skip
1245     }
1246   }
1247   \par\nopagebreak
1248   \addvspace { \l__enumext_minipage_left_skip }
1249 }

```

(End of definition for `__enumext_keyans_mini_addvspace:.`)

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1250 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1251 {
1252   \skip_zero_new:N \l__enumext_minipage_left_skip
1253   \skip_gzero_new:N \g__enumext_minipage_right_skip
1254   \skip_gzero_new:N \g__enumext_minipage_after_skip
1255   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1256   {
1257     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1258     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1259   }
1260   {
1261     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1262     \skip_gset:Nn \g__enumext_minipage_right_skip
1263     {
1264       \l__enumext_topsep_vii_skip
1265     }
1266     \skip_gset:Nn \g__enumext_minipage_after_skip
1267     {
1268       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1269     }
1270   }
1271 }
1272 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1273 {
1274   \skip_zero_new:N \l__enumext_minipage_after_skip
1275   \skip_zero_new:N \l__enumext_minipage_left_skip
1276   \skip_zero_new:N \l__enumext_minipage_right_skip
1277   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1278   {
1279     \skip_set:Nn \l__enumext_minipage_left_skip
1280     {
1281       0.5\box_dp:N \strutbox
1282     }
1283     \skip_set:Nn \l__enumext_minipage_right_skip
1284     {
1285       \l__enumext_partopsep_viii_skip
1286     }
1287     \skip_set:Nn \l__enumext_minipage_after_skip
1288     {
1289       1.6\box_dp:N \strutbox
1290     }
1291   }
1292 }

```

```

1293     \skip_set:Nn \l__enumext_minipage_left_skip
1294     {
1295         0.5875\box_dp:N \strutbox
1296     }
1297     \skip_set:Nn \l__enumext_minipage_right_skip
1298     {
1299         \l__enumext_topsep_viii_skip
1300     }
1301     \skip_set:Nn \l__enumext_minipage_after_skip
1302     {
1303         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1304     }
1305 }
1306 }

```

(End of definition for `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:`)

`\l__enumext_mini_addvspace_vii:`
`\l__enumext_mini_addvspace_viii:`

The functions `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `\l__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.

```

1307 \cs_new_protected:Nn \l__enumext_mini_addvspace_vii:
1308 {
1309     \l__enumext_mini_set_vskip_vii:
1310     \par\nopagebreak
1311     \addvspace { \l__enumext_minipage_left_skip }
1312 }
1313 \cs_new_protected:Nn \l__enumext_mini_addvspace_viii:
1314 {
1315     \l__enumext_mini_set_vskip_viii:
1316     \par\nopagebreak
1317     \addvspace { \l__enumext_minipage_left_skip }
1318 }

```

(End of definition for `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:`)

11.20.4 The command `\miniright`

The command `\miniright` will close the `\l__enumext_mini_env*` environment on the “left side”, open the `\l__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 `\l__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.

```

1319 \NewDocumentCommand \miniright { s }
1320 {
1321     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1322     {
1323         \msg_error:nnn { enumext } { wrong-miniright-place }
1324     }
1325     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1326     {
1327         \msg_error:nnn { enumext } { wrong-miniright-place }
1328     }
1329     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1330     {
1331         \l__enumext_keyans_mini_right_cmd:n {#1}
1332     }
1333     { \l__enumext_mini_right_cmd:n {#1} }
1334 }

```

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

`\l__enumext_mini_right_cmd:n`

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

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

```

1335 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1336 {
1337   \dim_compare:nNnTF
1338   { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1339   {
1340     \__enumext_multicols_stop:
1341     \end{__enumext_mini_env*}
1342     \hfill
1343     \begin{__enumext_mini_env*}
1344     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1345     \par\addvspace { \l__enumext_minipage_right_skip }
1346     \bool_if:nF {#1}
1347     {
1348       \centering
1349     }
1350     \int_gzero:N \g__enumext_minipage_stat_int
1351   }
1352   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1353 }

```

(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.

```

1354 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1355 {
1356   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1357   {
1358     \__enumext_keyans_multicols_stop:
1359     \end{__enumext_mini_env*}
1360     \hfill
1361     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1362     \par\addvspace { \l__enumext_minipage_right_skip }
1363     \bool_if:nF {#1}
1364     {
1365       \centering
1366     }
1367     \int_gzero:N \g__enumext_minipage_stat_int
1368   }
1369   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1370 }

```

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

11.21 Setting above and below keys

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

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

```

above* 1371 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1372 {
below* 1373   \keys_define:nn { enumext / #1 }
1374   {
1375     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1376     above .value_required:n = true,
1377     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1378               \keys_set:nn { enumext / #1 } { above = {##1} },
1379     above* .value_required:n = true,
1380     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1381     below .value_required:n = true,
1382     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1383               \keys_set:nn { enumext / #1 } { below = {##1} },
1384     below* .value_required:n = true,

```

```

1385     }
1386   }
1387   \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

11.21.1 Functions for above and below keys in enumext

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

```

1388 \cs_new_protected:Nn \__enumext_vspace_above:
1389 {
1390   \skip_if_eq:nnF
1391   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1392   {
1393     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1394     {
1395       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1396     }
1397     {
1398       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1399     }
1400   }
1401 }

```

(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.

```

1402 \cs_new_protected:Nn \__enumext_vspace_below:
1403 {
1404   \skip_if_eq:nnF
1405   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1406   {
1407     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1408     {
1409       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1410     }
1411     {
1412       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1413     }
1414   }
1415 }

```

(End of definition for `__enumext_vspace_below:`.)

11.21.2 Functions for above and below keys in keyans

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

```

1416 \cs_new_protected:Nn \__enumext_vspace_above_v:
1417 {
1418   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1419   {
1420     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1421     {
1422       \vspace*{ \l__enumext_vspace_above_v_skip }
1423     }
1424     { \vspace { \l__enumext_vspace_above_v_skip } }
1425   }
1426 }

```

(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.

```

1427 \cs_new_protected:Nn \__enumext_vspace_below_v:
1428 {
1429   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1430   {
1431     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1432     {
1433       \vspace*{ \l__enumext_vspace_below_v_skip }

```

```

1434     }
1435     { \vspace { \l__enumext_vspace_below_v_skip } }
1436   }
1437 }

```

(End of definition for `__enumext_vspace_below_v:`)

11.21.3 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.

```

1438 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1439 {
1440   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1441   {
1442     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1443     {
1444       \vspace*{ \l__enumext_vspace_above_vii_skip }
1445     }
1446     { \vspace { \l__enumext_vspace_above_vii_skip } }
1447   }
1448 }
1449 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1450 {
1451   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1452   {
1453     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1454     {
1455       \vspace*{ \l__enumext_vspace_above_viii_skip }
1456     }
1457     { \vspace { \l__enumext_vspace_above_viii_skip } }
1458   }
1459 }

```

(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.

```

1460 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1461 {
1462   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1463   {
1464     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1465     {
1466       \vspace*{ \l__enumext_vspace_below_vii_skip }
1467     }
1468     { \vspace { \l__enumext_vspace_below_vii_skip } }
1469   }
1470 }
1471 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1472 {
1473   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1474   {
1475     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1476     {
1477       \vspace*{ \l__enumext_vspace_below_viii_skip }
1478     }
1479     { \vspace { \l__enumext_vspace_below_viii_skip } }
1480   }
1481 }

```

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

11.22 Setting series, resume and resume* keys

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

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1482 \cs_set_protected:Npn \__enumext_tmp:n #1
1483 {
1484   \keys_define:nn { enumext / #1 }
1485   {
1486     series .str_set:N = \__enumext_series_str,
1487     series .value_required:n = true,
1488     resume .code:n = \__enumext_resume_series:n {##1},
1489     resume* .code:n = \__enumext_resume_starred:,
1490     resume* .value_forbidden:n = true,
1491   }
1492 }
1493 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.22.1 Internal functions for series key

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

```

1494 \cs_new:Npn \__enumext_filter_series:n #1
1495 {
1496   \use:e
1497   {
1498     \keyval_parse:NNn
1499     \__enumext_filter_series_key:n
1500     \__enumext_filter_series_pair:nn {#1}
1501   }
1502 }

```

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.

```

1503 \cs_new:Npn \__enumext_filter_series_key:n #1
1504 {
1505   \str_case:nnF {#1}
1506   {
1507     { resume } {}
1508     { resume* } {}
1509   }
1510   { , { \exp_not:n {#1} } }
1511 }

```

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.

```

1512 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1513 {
1514   \str_case:nnF {#1}
1515   {
1516     { series } {}
1517     { resume } {}
1518     { start } {}
1519     { save-ans } {}
1520     { save-key } {}
1521   }
1522   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1523 }

```

(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 (§11.33) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.36).

```

1524 \cs_new_protected:Npn \__enumext_parse_series:n #1
1525 {

```



```

1526 \str_if_empty:NTF \l__enumext_series_str
1527 {
1528     \bool_if:NF \l__enumext_resume_active_bool
1529     {
1530         \__enumext_resume_last:n {#1}
1531     }
1532 }
1533 {
1534     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1535     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1536     { \__enumext_filter_series:n {#1} }
1537     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1538     {
1539         \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1540     }
1541 }
1542 }

```

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.

```

1543 \cs_new_protected:Npn \__enumext_resume_last:n #1
1544 {
1545     \bool_if:NT \l__enumext_standar_first_bool
1546     {
1547         \tl_gclear:N \g__enumext_standar_series_tl
1548         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1549     }
1550     \bool_if:NT \l__enumext_starred_first_bool
1551     {
1552         \tl_gclear:N \g__enumext_starred_series_tl
1553         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1554     }
1555 }

```

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

11.22.2 Internal function to save counter value

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same *⟨series name⟩* but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§11.33) and the `enumext*` environment definition (§11.36).

```

1556 \cs_new_protected:Nn \__enumext_resume_save_counter:
1557 {
1558     \bool_if:NT \g__enumext_standar_bool
1559     {
1560         \tl_if_empty:NF \l__enumext_series_str
1561         {
1562             \int_gset_eq:cN
1563             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1564         }
1565         \tl_if_empty:NTF \l__enumext_resume_name_tl
1566         {
1567             \str_if_empty:NF \l__enumext_series_str
1568             {
1569                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1570             }
1571         }
1572         {
1573             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1574             {
1575                 \int_gset_eq:cN

```

```

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

```

(End of definition for __enumext_resume_save_counter:.)

11.22.3 Internal functions for resume key

__enumext_resume_series:n

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

```

1613 \cs_new_protected:Npn \__enumext_resume_series:n #1
1614 {
1615     \tl_if_empty:nTF {#1}
1616     {
1617         \__enumext_resume_counter:n { }
1618     }
1619     {
1620         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1621         {
1622             \__enumext_resume_counter:n {#1}
1623             \bool_if:NT \g__enumext_standar_bool
1624             {
1625                 \keys_set:nv { enumext / level-1 }
1626                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1627             }
1628             \bool_if:NT \g__enumext_starred_bool
1629             {
1630                 \keys_set:nv { enumext / enumext* }
1631                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1632             }
1633         }
1634     }
1635     \bool_if:NT \g__enumext_standar_bool

```

```

1636         {
1637             \msg_error:nnn { enumext } { unknown-series } {#1}
1638         }
1639         \bool_if:NT \g__enumext_starred_bool
1640         {
1641             \msg_error:nnn { enumext } { unknown-series } {#1}
1642         }
1643     }
1644 }
1645 }

```

(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`.

```

1646 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1647 {
1648     \bool_set_true:N \l__enumext_resume_active_bool
1649     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1650     \tl_if_empty:NTF \l__enumext_resume_name_tl
1651     {
1652         \__enumext_resume_counter:
1653     }
1654     {
1655         \__enumext_resume_counter_series:
1656     }
1657     \__enumext_resume_counter_save_ans:
1658 }

```

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.

```

1659 \cs_new_protected:Nn \__enumext_resume_counter:
1660 {
1661     \bool_if:NT \g__enumext_standar_bool
1662     {
1663         \int_gincr:N \g__enumext_resume_int
1664         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1665     }
1666     \bool_if:NT \g__enumext_starred_bool
1667     {
1668         \int_gincr:N \g__enumext_resume_vii_int
1669         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1670     }
1671 }

```

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.

```

1672 \cs_new_protected:Nn \__enumext_resume_counter_series:
1673 {
1674     \bool_if:NT \g__enumext_standar_bool
1675     {
1676         \int_set:Nn \l__enumext_start_i_int
1677         {
1678             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1679         }
1680     }
1681     \bool_if:NT \g__enumext_starred_bool
1682     {
1683         \int_set:Nn \l__enumext_start_vii_int
1684         {
1685             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1686         }
1687     }
1688 }

```

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.

```

1689 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1690 {
1691   \bool_lazy_and:nnT
1692     { \bool_if_p:N \__enumext_standar_first_bool }
1693     { \bool_if_p:N \__enumext_store_active_bool }
1694   {
1695     \int_set:Nn \__enumext_start_i_int
1696       {
1697         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1698       }
1699   }
1700   \bool_lazy_and:nnT
1701     { \bool_if_p:N \__enumext_starred_first_bool }
1702     { \bool_if_p:N \__enumext_store_active_bool }
1703   {
1704     \int_set:Nn \__enumext_start_vii_int
1705       {
1706         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1707       }
1708   }
1709 }

```

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

11.22.4 Internal function for `resume*` key

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

```

1710 \cs_new_protected:Nn \__enumext_resume_starred:
1711 {
1712   \bool_if:NT \g__enumext_standar_bool
1713   {
1714     \tl_if_empty:NF \g__enumext_standar_series_tl
1715     {
1716       \__enumext_resume_counter:n { }
1717       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1718     }
1719   }
1720   \bool_if:NT \g__enumext_starred_bool
1721   {
1722     \tl_if_empty:NF \g__enumext_starred_series_tl
1723     {
1724       \__enumext_resume_counter:n { }
1725       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1726     }
1727   }
1728 }

```

(End of definition for `__enumext_resume_starred:`.)

11.23 Setting `save-ans`, `check-ans` and `no-store` keys

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

11.23.1 Setting `save-ans` key

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

```

1729 \cs_set_protected:Npn \__enumext_tmp:n #1
1730 {
1731   \keys_define:nn { enumext / #1 }
1732   {
1733     save-ans .code:n = \__enumext_storing_set:n {##1},
1734     save-ans .value_required:n = true,
1735   }
1736 }
1737 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

11.23.2 Internal functions for save-ans key

```

\__enumext_start_save_ans_msg:
\__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:`.

```

1738 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1739 {
1740   \msg_term:nnVV { enumext } { save-ans-log }
1741   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1742 }
1743 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1744 {
1745   \msg_term:nnVV { enumext } { save-ans-log-hook }
1746   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1747 }

```

(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.

```

1748 \cs_new_protected:Npn \__enumext_storing_set:n #1
1749 {
1750   \tl_set:Nx \l__enumext_store_name_tl {#1}
1751   \tl_if_empty:NTF \l__enumext_store_name_tl
1752   {
1753     \bool_lazy_or:nnT
1754     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1755     {
1756       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1757     }
1758   }
1759   {
1760     \bool_lazy_or:nnT
1761     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1762     {
1763       \__enumext_start_save_ans_msg:
1764       \__enumext_storing_exec:
1765     }
1766   }
1767 }

```

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*`.

```

1768 \cs_new_protected:Nn \__enumext_storing_exec:
1769 {
1770   \bool_set_true:N \l__enumext_store_active_bool
1771   \bool_set_true:N \l__enumext_check_answers_bool
1772   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1773   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1774   {
1775     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1776     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1777   }
1778   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1779   {
1780     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1781     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1782   }
1783   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1784   {
1785     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl

```

```

1786         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1787     }
1788 }

```

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

11.23.3 The check answer mechanism

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

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

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

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

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

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

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

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

11.23.4 Setting check-ans and no-store keys

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

```

check-ans 1789 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1790 {
1791     \keys_define:nn { enumext / #1 }
1792     {
1793         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1794         check-ans .initial:n = false,
1795         check-ans .value_required:n = true,
1796         no-store .code:n = {
1797             \bool_set_false:N \l__enumext_check_answers_bool
1798             \bool_set_false:N \l__enumext_check_ans_key_bool
1799         },
1800         no-store .value_forbidden:n = true,
1801     }
1802 }
1803 \clist_map_inline:nn
1804 {
1805     level-1, level-2, level-3, level-4, enumext*
1806 }
1807 { \__enumext_tmp:n {#1} }

```

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

11.23.5 Set-up check answer mechanism

The function `__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `__enumext_check_ans_level:` only if “true”, i.e. the key `no-store` is not active.

```

1808 \cs_new_protected:Nn \__enumext_check_ans_active:
1809 {
1810     \tl_if_empty:NF \l__enumext_store_name_tl

```

```

1811     {
1812         \bool_if:NT \l__enumext_check_answers_bool
1813         {
1814             \__enumext_check_ans_level:
1815         }
1816     }
1817 }

```

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.

```

1818 \cs_new_protected:Nn \__enumext_check_ans_level:
1819 {
1820     \int_case:nn { \l__enumext_level_int }
1821     {
1822         { 1 }{
1823             \bool_lazy_all:nT
1824             {
1825                 { \bool_if_p:N \g__enumext_starred_bool }
1826                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1827             }
1828             {
1829                 \int_gdecr:N \g__enumext_item_number_int
1830             }
1831         }
1832         { 2 }{
1833             \int_gdecr:N \g__enumext_item_number_int
1834         }
1835         { 3 }{
1836             \int_gdecr:N \g__enumext_item_number_int
1837         }
1838         { 4 }{
1839             \int_gdecr:N \g__enumext_item_number_int
1840         }
1841     }

```

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.

```

1842     \int_case:nn { \l__enumext_level_h_int }
1843     {
1844         { 1 }{
1845             \bool_lazy_all:nT
1846             {
1847                 { \bool_if_p:N \g__enumext_standar_bool }
1848                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1849             }
1850             {
1851                 \int_gdecr:N \g__enumext_item_number_int
1852             }
1853         }
1854     }
1855 }

```

(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.

```

1856 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1857 {
1858     \bool_lazy_and:nnT
1859     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1860     { \bool_if_p:N \g__enumext_standar_bool }
1861     {
1862         \bool_gset_true:N \g__enumext_check_ans_key_bool
1863     }
1864     \bool_lazy_and:nnT
1865     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1866     { \bool_if_p:N \g__enumext_starred_bool }
1867     {

```



```

1868         \bool_gset_true:N \g__enumext_check_ans_key_bool
1869     }
1870 }

```

(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:`.

```

1871 \cs_new_protected:Nn \__enumext_item_answer_diff:
1872 {
1873     \int_gset:Nn \g__enumext_item_answer_diff_int
1874     {
1875         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1876     }
1877 }

```

(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:`.

```

1878 \cs_new_protected:Nn \__enumext_check_ans_show:
1879 {
1880     \int_case:nn { \g__enumext_item_answer_diff_int }
1881     {
1882         { -1 } { \__enumext_check_ans_msg_less: }
1883         { 0 } { \__enumext_check_ans_msg_same_ok: }
1884         { 1 } { \__enumext_check_ans_msg_greater: }
1885     }
1886 }
1887 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1888 {
1889     \msg_warning:nneee { enumext } { item-less-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_same_ok:
1893 {
1894     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1895     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1896 }
1897 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1898 {
1899     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1900     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1901 }

```

(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:`.

```

1902 \cs_new_protected:Nn \__enumext_check_ans_log:
1903 {
1904     \int_case:nn { \g__enumext_item_answer_diff_int }
1905     {
1906         { -1 } { \__enumext_check_ans_log_msg_less: }
1907         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1908         { 1 } { \__enumext_check_ans_log_msg_greater: }
1909     }
1910 }
1911 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1912 {
1913     \msg_log:nneee { enumext } { item-less-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_same_ok:

```

```

1917 {
1918   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1919   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1920 }
1921 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1922 {
1923   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1924   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1925 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.23.6 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:n for the environments `enumext` and `enumext*` we must make sure that we are not nested at any level.

```

1926 \cs_new_protected:Nn \__enumext_execute_after_env:
1927 {
1928   \int_compare:nNt { \l__enumext_level_int } = { 0 }
1929   {
1930     \tl_if_empty:NF \g__enumext_store_name_tl
1931     {
1932       \__enumext_stop_save_ans_msg:
1933       \__enumext_item_answer_diff:
1934       \__enumext_log_global_vars:
1935       \__enumext_log_answer_vars:
1936       \bool_if:NTF \g__enumext_check_ans_key_bool
1937       {
1938         \__enumext_check_ans_show:
1939       }
1940       { \__enumext_check_ans_log: }
1941     }
1942     \__enumext_reset_global_vars:
1943   }
1944 }

```

(End of definition for __enumext_execute_after_env:.)

11.23.7 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.

```

1945 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1946 {
1947   \int_compare:nNt
1948   { \g__enumext_check_starred_cmd_int } = { 0 }
1949   {
1950     \msg_warning:nnnV
1951     { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1952   }
1953   \int_compare:nNt
1954   { \g__enumext_check_starred_cmd_int } > { 1 }
1955   {
1956     \msg_warning:nnnV
1957     { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1958   }
1959   \int_gzero:N \g__enumext_check_starred_cmd_int
1960   \tl_clear:N \l__enumext_check_start_line_env_tl
1961 }

```

(End of definition for __enumext_check_starred_cmd:n.)

11.24 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*`.

```

1962 \cs_set_protected:Npn \__enumext_tmp:n #1
1963 {
1964   \keys_define:nn { enumext / #1 }
1965   {
1966     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1967     wrap-ans .initial:n = \fbox{##1},
1968     wrap-ans .value_required:n = true,
1969     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1970     wrap-opt .initial:n = [{##1}],
1971     wrap-opt .value_required:n = true,
1972     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1973     save-sep .initial:n = {, ~ },
1974     save-sep .value_required:n = true,
1975     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
1976     mark-ans .initial:n = \textasteriskcentered,
1977     mark-ans .value_required:n = true,
1978     mark-pos .choice:,
1979     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1980     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1981     mark-pos .initial:n = right,
1982     mark-pos .value_required:n = true,
1983     show-ans .bool_set:N = \__enumext_show_answer_bool,
1984     show-ans .initial:n = false,
1985     show-ans .value_required:n = true,
1986     show-pos .bool_set:N = \__enumext_show_position_bool,
1987     show-pos .initial:n = false,
1988     show-pos .value_required:n = true,
1989     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
1990     mark-ref .initial:n = \textasteriskcentered,
1991     mark-ref .value_required:n = true,
1992     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
1993     save-ref .initial:n = false,
1994     save-ref .value_required:n = true,
1995   }
1996 }
1997 \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`.

```

1998 \cs_set_protected:Npn \__enumext_tmp:n #1
1999 {
2000   \keys_define:nn { enumext / #1 }
2001   {
2002     mark-pos .choice:,
2003     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2004     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2005     mark-pos .initial:n = right,
2006     mark-pos .value_required:n = true,
2007     show-ans .bool_set:N = \__enumext_show_answer_bool,
2008     show-ans .initial:n = false,
2009     show-ans .value_required:n = true,
2010     show-pos .bool_set:N = \__enumext_show_position_bool,
2011     show-pos .initial:n = false,
2012     show-pos .value_required:n = true,
2013   }
2014 }
2015 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

11.24.1 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⟩*.

```

2016 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2017 {
2018   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2019   {
2020     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2021     \tl_set:ce
2022       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2023       { \__enumext_filter_save_key:n {#1} }
2024   }
2025 }
2026 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2027 {
2028   \bool_if:NF \l__enumext_store_save_key_vii_bool
2029   {
2030     \tl_clear:N \l__enumext_store_save_key_vii_tl
2031     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2032   }
2033 }

```

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

11.24.2 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.

```

2034 \cs_set_protected:Npn \__enumext_tmp:n #1
2035 {
2036   \keys_define:nn { enumext / enumext* }
2037   {
2038     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2039     save-key .value_required:n = true,
2040   }
2041   \keys_define:nn { enumext / #1 }
2042   {
2043     save-key .code:n = \__enumext_parse_save_key:n {##1},
2044     save-key .value_required:n = true,
2045   }
2046 }
2047 \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*`.

```

2048 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2049 {
2050   \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2051   \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2052   \tl_set:ce
2053     { \l__enumext_store_save_key_ \__enumext_level: _tl }
2054     { \__enumext_filter_save_key:n {#1} }
2055 }
2056 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2057 {
2058   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2059   \tl_clear:N \l__enumext_store_save_key_vii_tl
2060   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2061 }

```

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

11.24.3 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.

```
2062 \cs_new:Npn \__enumext_filter_save_key:n #1
2063 {
2064   \use:e
2065   {
2066     \keyval_parse:NNn
2067     \__enumext_filter_save_key_key:n
2068     \__enumext_filter_save_key_pair:nn {#1}
2069   }
2070 }
```

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.

```
2071 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2072 {
2073   \str_case:nnF {#1}
2074   {
2075     { resume } {} { resume* } {} { no-store } {}
2076   }
2077   { , { \exp_not:n {#1} } }
2078 }
```

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.

```
2079 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2080 {
2081   \str_case:nnF {#1}
2082   {
2083     { series } {} { resume } {} { save-ans } {}
2084     { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2085     { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2086     { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2087     { mini-right } {} { mini-right* } {}
2088   }
2089   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2090 }
```

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

11.24.4 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.

```
2091 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2092 {
2093   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2094   {
2095     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2096   }
2097   { #1 }
2098 }
2099 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

11.24.5 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.

```
2100 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2101 {
```

```

2102 \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2103 }
2104 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

11.24.6 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.

```

2105 \cs_new_protected:Nn \__enumext_store_level_open:
2106 {
2107   \bool_if:NT \l__enumext_check_answers_bool
2108   {
2109     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2110     {
2111       \__enumext_store_addto_seq:n
2112       {
2113         \item \begin{enumext}
2114       }
2115     }
2116     {
2117       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2118       {
2119         \item \begin{enumext} [
2120       }
2121       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2122       {
2123         ]
2124       }
2125       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2126     }
2127   }
2128 }
2129 \cs_new_protected:Nn \__enumext_store_level_close:
2130 {
2131   \bool_if:NT \l__enumext_check_answers_bool
2132   {
2133     \__enumext_store_addto_seq:n { \end{enumext} }
2134   }
2135 }

```

(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.

```

2136 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2137 {
2138   \bool_if:NT \l__enumext_check_answers_bool
2139   {
2140     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2141     {
2142       \__enumext_store_addto_seq:n
2143       {
2144         \item \mode_leave_vertical:
2145         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2146         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2147       }
2148     }
2149     {
2150       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2151       {
2152         \item \mode_leave_vertical:
2153         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2154         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2155       }
2156       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2157       {
2158         ]

```

```

2159         }
2160         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2161     }
2162 }
2163 }
2164 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2165 {
2166     \bool_if:NT \l__enumext_check_answers_bool
2167     {
2168         \__enumext_store_addto_seq:n { \end{enumext*} }
2169     }
2170 }

```

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

11.24.7 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`

```

2171 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2172 {
2173     \mode_leave_vertical:
2174     \skip_horizontal:n { -\dim_use:N #2 }
2175     \makebox[0pt][ r ]
2176     {
2177         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2178         {
2179             \tl_use:N \l__enumext_mark_answer_sym_tl
2180         }
2181     }
2182     \skip_horizontal:n { \dim_use:N #2 }
2183 }
2184 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

11.25 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*`.

```

2185 \keys_define:nn { enumext / anskey }
2186 {
2187     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2188     break-col .default:n = true,
2189     break-col .value_forbidden:n = true,
2190     item-join .int_set:N = \l__enumext_store_item_join_int,
2191     item-join .value_required:n = true,
2192     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2193     item-star .default:n = true,
2194     item-star .value_forbidden:n = true,
2195     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2196     item-sym* .value_required:n = true,
2197     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2198     item-pos* .value_required:n = true,
2199 }

```

🟢 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`.


```

2200 \NewDocumentCommand \anskey { o +m }
2201 {
2202   \__enumext_anskey_safe_outer:
2203   \group_begin:
2204     \bool_if:NT \l__enumext_check_answers_bool
2205     {
2206       \int_gincr:N \g__enumext_item_anskey_int
2207       \__enumext_anskey_safe_inner:n {#2}
2208       \__enumext_store_anskey_code:nn {#1} {#2}
2209     }
2210   \group_end:
2211 }

```

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

11.25.1 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.

```

2212 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2213 {
2214   \bool_if:NF \l__enumext_store_active_bool
2215   {
2216     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2217   }
2218   \int_compare:nNtT { \l__enumext_keyans_level_int } = { 1 }
2219   {
2220     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2221   }
2222   \int_compare:nNtT { \l__enumext_keyans_pic_level_int } = { 1 }
2223   {
2224     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2225   }
2226 }

```

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.

```

2227 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2228 {
2229   \tl_if_empty:nT {#1}
2230   {
2231     \msg_error:nn { enumext } { anskey-empty-arg }
2232   }
2233   \int_incr:N \l__enumext_anskey_level_int
2234   \int_compare:nNtT { \l__enumext_anskey_level_int } > { 1 }
2235   {
2236     \msg_error:nn { enumext } { anskey-nested }
2237   }
2238 }

```

(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.

```

2239 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2240 {
2241   \__enumext_store_addto_prop:n {#2}
2242   \bool_if:NT \l__enumext_store_ref_key_bool
2243   {
2244     \__enumext_store_internal_ref:
2245   }
2246   \__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`.

```

2247   \tl_if_novalue:nF {#1}
2248   {

```

```

2249     \keys_set:nn { enumext / anskey } {#1}
2250   }
2251   \tl_clear:N \l__enumext_store_anskey_arg_tl
2252   \bool_lazy_and:nnT
2253     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2254     { \bool_not_p:n { \l__enumext_starred_bool } }
2255   {
2256     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2257   }
2258   \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`.

```

2259   \bool_lazy_and:nnT
2260     { \bool_not_p:n { \l__enumext_starred_bool } }
2261     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2262   {
2263     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2264       {
2265         ( \exp_not:V \l__enumext_store_item_join_int )
2266       }
2267   }

```

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`).

```

2268   \bool_if:NTF \l__enumext_store_item_star_bool
2269   {
2270     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2271     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2272     {
2273       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2274         {
2275           [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2276         }
2277     }
2278   \dim_compare:nT
2279   {
2280     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2281   }
2282   {
2283     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2284       {
2285         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2286       }
2287   }
2288   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2289 }
2290 {
2291   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2292 }

```

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`).

```

2293   \bool_lazy_and:nnT
2294     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2295     { \bool_if_p:N \l__enumext_hyperref_bool }
2296   {
2297     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2298     {
2299       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2300       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2301     }
2302   }
2303   \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2304 }

```

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

`\l__enumext_store_internal_ref:`

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

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

```

2305 \cs_new_protected:Nn \__enumext_store_internal_ref:
2306 {
2307   \cs_set_protected:Npn \__enumext_tmp:n ##1
2308   {
2309     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2310     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2311     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2312     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2313   }
2314   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2315   \cs_set:Npn \__enumext_tmp:n ##1
2316   { . \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.

```

2317 \bool_lazy_all:nT
2318 {
2319   { \bool_if_p:N \g__enumext_starred_bool }
2320   { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
2321 }
2322 {
2323   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2324   { \tl_use:N \__enumext_label_copy_vii_tl }
2325 }
2326 \bool_lazy_all:nT
2327 {
2328   { \bool_if_p:N \l__enumext_standar_bool }
2329   { \bool_if_p:N \g__enumext_starred_bool }
2330   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2331 }
2332 {
2333   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2334   {
2335     \tl_use:N \__enumext_label_copy_vii_tl
2336     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2337   }
2338 }

```

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.

```

2339 \bool_lazy_all:nT
2340 {
2341   { \bool_if_p:N \l__enumext_standar_bool }
2342   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2343   { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_zero_int } }
2344   { \bool_not_p:n { \__enumext_starred_bool } }
2345 }
2346 {
2347   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2348   {
2349     \tl_use:N \__enumext_label_copy_i_tl
2350     \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
2351   }
2352 }
2353 \cs_set:Npn \__enumext_tmp:n ##1
2354 { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2355 \bool_lazy_all:nT
2356 {
2357   { \bool_if_p:N \l__enumext_standar_bool }
2358   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2359   { \bool_not_p:n { \g__enumext_starred_bool } }
2360   { \int_compare_p:nNn { \__enumext_level_h_int } > { \c_zero_int } }
2361 }
2362 {
2363   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2364   {
2365     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2366     . \tl_use:N \__enumext_label_copy_vii_tl
2367   }
2368 }

```

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

```

2369 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2370 {
2371   \l__enumext_store_name_tl \c_colon_str
2372   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2373 }

```

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.

```

2374 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2375 {
2376   \__enumext_newlabel:nn
2377   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2378   { \l__enumext_newlabel_arg_two_tl }
2379 }
2380 \l__enumext_store_write_aux_file_tl
2381 }

```

(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.

```

2382 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2383 {
2384   \par
2385   \bool_if:NT \l__enumext_starred_bool
2386   {
2387     \cs_set:Nn \__enumext_level: { vii }
2388   }
2389   \__enumext_print_keyans_box:cc
2390   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2391   { \l__enumext_labelsep_ \__enumext_level: _dim }
2392   \__enumext_anskey_wrapper:n { #1 }
2393 }

```

(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.

```

2394 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2395 {
2396   \bool_if:NT \l__enumext_show_answer_bool
2397   {
2398     \__enumext_store_anskey_show_wrap:n { #1 }
2399   }
2400   \bool_if:NT \l__enumext_show_position_bool
2401   {
2402     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2403     {
2404       \group_begin:
2405       \exp_not:N \normalfont
2406       \exp_not:N \footnotesize [ \int_eval:n
2407       {
2408         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2409       }
2410       ]
2411       \group_end:
2412     }
2413     \__enumext_store_anskey_show_wrap:n { #1 }
2414   }
2415 }

```

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

11.26 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

2416 \NewDocumentEnvironment{anskey*}{ o +b }
2417 {
2418   \__enumext_anskey_env_safe_outer:
2419   \bool_if:NT \l__enumext_check_answers_bool
2420   {
2421     \int_gincr:N \g__enumext_item_anskey_int
2422     \__enumext_anskey_env_safe_inner:n {#2}
2423     \__enumext_store_anskey_code:nn {#1} {#2}
2424   }
2425 }
2426 {}
```

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.

```
2427 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
2428 {
2429   \bool_if:NF \l__enumext_store_active_bool
2430   {
2431     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2432   }
2433   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2434   {
2435     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2436   }
2437   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2438   {
2439     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2440   }
2441 }
```

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.

```
2442 \cs_new_protected:Npn \__enumext_anskey_env_safe_inner:n #1
2443 {
2444   \tl_if_empty:nT {#1}
2445   {
2446     \msg_error:nn { enumext } { anskey-empty-arg }
2447   }
2448   \int_incr:N \l__enumext_anskey_level_int
2449   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2450   {
2451     \msg_error:nn { enumext } { anskey-nested-env }
2452   }
2453 }
```

(End of definition for `anskey*`, `__enumext_anskey_env_safe_outer:`, and `__enumext_anskey_env_safe_inner:n`. This function is documented on page 12.)

11.27 Common functions for keyans, keyans* and keyanspic

11.27.1 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`.

```
2454 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2455 {
2456   \tl_clear:N \l__enumext_store_keyans_label_tl
2457   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2458   {
2459     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2460   }
2461   {
2462     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2463   }
2464   \tl_if_novalue:nF { #1 }
```

```

2465     {
2466       % Set save-sep
2467       \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2468       {
2469         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2470       }
2471       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2472     }
2473     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2474   }

```

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

11.27.2 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.

```

2475 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2476 {
2477   \bool_if:NT \l__enumext_store_ref_key_bool
2478   {
2479     \cs_set_protected:Npn \__enumext_tmp:n ##1
2480     {
2481       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2482       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2483       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2484       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2485     }
2486     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2487     \__enumext_keyans_store_ref_aux_i:
2488   }
2489 }

```

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`.

```

2490 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2491 {
2492   \bool_if:NT \g__enumext_starred_bool
2493   {
2494     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2495   }
2496   \int_compare:nNnT { \l__enumext_keyans_pic_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_vi_tl }
2500   }
2501   \int_compare:nNnT { \l__enumext_keyans_level_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_v_tl }
2505   }
2506   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2507   {
2508     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2509     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2510   }
2511   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2512   {
2513     \l__enumext_store_name_tl \c_colon_str
2514     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2515   }
2516   \__enumext_keyans_store_ref_aux_ii:
2517 }

```

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.

```

2518 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2519 {
2520   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2521   {
2522     \__enumext_newlabel:nn
2523     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2524     { \l__enumext_newlabel_arg_two_tl }
2525   }
2526   \l__enumext_store_write_aux_file_tl
2527 }

```

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

11.27.3 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.

```

2528 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2529 {
2530   \tl_clear:N \l__enumext_store_keyans_label_tl
2531   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2532   {
2533     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2534   }
2535   {
2536     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2537   }
2538   \tl_if_novalue:nF { #1 }
2539   {
2540     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2541     {
2542       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2543       {
2544         \l__enumext_store_keyans_item_opt_sep_tl
2545       }
2546     }
2547     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2548   }
2549   \__enumext_keyans_addto_seq_link:
2550 }

```

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.

```

2551 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2552 {
2553   \bool_lazy_and:nnT
2554   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2555   { \bool_if_p:N \l__enumext_hyperref_bool }
2556   {
2557     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2558     {
2559       \hfill \exp_not:N \hyperlink
2560       {
2561         \exp_not:V \l__enumext_newlabel_arg_one_tl
2562       }
2563       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2564     }
2565   }
2566   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2567   \bool_if:NT \l__enumext_check_answers_bool
2568   {
2569     \int_gincr:N \g__enumext_item_anskey_int

```



```

2570     }
2571 }

```

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

11.27.4 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:
2572 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2573 {
2574   \tl_if_novalue:nF { #1 }
2575   {
2576     \tl_set:Nx \__enumext_keyans_item_opt_tl { #1 }
2577   }
2578   \bool_if:NT \l__enumext_show_answer_bool
2579   {
2580     \__enumext_keyans_show_ans:
2581   }
2582   \bool_if:NT \l__enumext_show_position_bool
2583   {
2584     \__enumext_keyans_show_pos:
2585   }
2586 }
2587 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2588 {
2589   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2590   {
2591     \bool_lazy_or:nnT
2592     { \bool_if_p:N \l__enumext_show_answer_bool }
2593     { \bool_if_p:N \l__enumext_show_position_bool }
2594     {
2595       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2596     }
2597   }
2598 }
2599 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2600 {
2601   \tl_put_left:Nn \l__enumext_label_v_tl
2602   {
2603     \__enumext_print_keyans_box:NN
2604     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2605   }
2606 }
2607 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2608 {
2609   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2610   {
2611     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2612     {
2613       \group_begin:
2614       \exp_not:N \normalfont
2615       \exp_not:N \footnotesize [ \int_eval:n
2616       {
2617         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2618       }
2619       ]
2620       \group_end:
2621     }
2622   }
2623   {
2624     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2625     {
2626       \group_begin:
2627       \exp_not:N \normalfont
2628       \exp_not:N \footnotesize [ \int_eval:n
2629       {
2630         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2631       }

```

```

2632         ]
2633     \group_end:
2634 }
2635 }
2636 \tl_put_left:Nn \__enumext_label_v_tl
2637 {
2638     \__enumext_print_keyans_box:NN
2639     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2640 }
2641 }

```

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

11.28 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*`.

```

2642 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2643 {
2644     \keys_define:nn { enumext / #1 }
2645     {
2646         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2647         item-sym* .value_required:n = true,
2648         item-sym* .initial:n = {$\star$},
2649         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2650         item-pos* .value_required:n = true,
2651     }
2652 }
2653 \clist_map_inline:nn
2654 {
2655     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2656 }
2657 { \__enumext_tmp:nn #1 }

```

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

11.29 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](#).

```

2658 \cs_new_protected:Nn \__enumext_footnotetext:nn
2659 {
2660     \footnotetext[#1]{#2}
2661 }
2662 \cs_new_protected:Nn \__enumext_renew_footnote:
2663 {
2664     \seq_gclear:N \g__enumext_footnote_arg_seq
2665     \seq_gclear:N \g__enumext_footnote_int_seq
2666     \RenewDocumentCommand \footnote { o +m }
2667     {
2668         \tl_if_novalue:nTF {##1}
2669         {
2670             \stepcounter{footnote}
2671             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2672         }
2673         {
2674             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2675         }
2676         \footnotemark [ \g__enumext_footnote_int ]
2677         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2678         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2679     }
2680 }
2681 \cs_new_protected:Nn \__enumext_print_footnote:
2682 {
2683     \seq_if_empty:NF \g__enumext_footnote_int_seq
2684     {
2685         \seq_map_pairwise_function:NNN
2686         \g__enumext_footnote_int_seq

```

```

2687         \g__enumext_footnote_arg_seq
2688         \__enumext_footnotetext:nn
2689     }
2690 }

```

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

11.30 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.

11.30.1 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: (§11.31)`.

```

2691 \cs_new_protected:Npn \__enumext_default_item:n #1
2692 {
2693     \tl_if_novalue:nTF {#1}
2694     {
2695         \bool_if:NT \l__enumext_check_answers_bool
2696         {
2697             \int_gincr:N \g__enumext_item_number_int
2698         }
2699         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2700         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2701     }
2702     {
2703         \bool_set_eq:cc
2704         { \l__enumext_wrap_label_ \__enumext_level: _bool }
2705         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2706         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2707     }
2708 }

```

(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: (§11.31)`.

```

2709 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2710 {
2711     \tl_if_novalue:nF {#1}
2712     {
2713         \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2714     }
2715     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2716     \tl_if_novalue:nTF {#2}

```

```

2717     {
2718         \dim_set_eq:cc
2719         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2720         { \__enumext_labelsep_ \__enumext_level: _dim }
2721     }
2722     {
2723         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2724     }
2725     \bool_if:NT \__enumext_check_answers_bool
2726     {
2727         \int_gincr:N \g__enumext_item_number_int
2728     }
2729     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2730     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2731 }

```

(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 (§11.32.2).

```

2732 \cs_new_protected:Nn \__enumext_redefine_item:
2733 {
2734     \RenewDocumentCommand \item { s o o }
2735     {
2736         \bool_if:nTF {##1}
2737         {
2738             \__enumext_starred_item:nn {##2} {##3}
2739         }
2740         { \__enumext_default_item:n {##2} }
2741     }
2742 }

```

(End of definition for __enumext_redefine_item:.)

11.30.2 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`.

```

2743 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2744 {
2745     \tl_if_novalue:nTF { #1 }
2746     {
2747         \bool_set_true:N \__enumext_wrap_label_v_bool
2748         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2749     }
2750     {
2751         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2752         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2753     }
2754 }

```

(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 $\langle label \rangle$.

```

2755 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2756 {
2757     \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2758     \__enumext_keyans_show_left:n { #1 }
2759     \bool_set_true:N \__enumext_wrap_label_v_bool
2760     \__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 $\langle sequence \rangle$.

```

2761 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2762 \__enumext_keyans_addto_prop:n { #1 }
2763 \__enumext_keyans_store_ref:
2764 \__enumext_keyans_addto_seq:n { #1 }
2765 \int_gincr:N \g__enumext_check_starred_cmd_int
2766 }

```

(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 (§11.32.2).

```

2767 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2768 {
2769   \RenewDocumentCommand \item { s o }
2770   {
2771     \bool_if:nTF {##1}
2772     {
2773       \peek_remove_spaces:n
2774       {
2775         \__enumext_keyans_starred_item:n {##2}
2776       }
2777     }
2778     {
2779       \__enumext_keyans_default_item:n {##2}
2780     }
2781   }
2782 }

```

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

11.31 Redefining `\makeLabel` command

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

11.31.1 Redefining `\makeLabel` for `enumext`

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

```

2783 \cs_new_protected:Nn \__enumext_item_starred:
2784 {
2785   \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2786   {
2787     \mode_leave_vertical:
2788     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2789     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2790     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2791   }
2792 }

```

(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 (§11.32.2).

```

2793 \cs_new_protected:Nn \__enumext_make_label:
2794 {
2795   \RenewDocumentCommand \makeLabel { m }
2796   {
2797     \tl_use:c { \l__enumext_label_fill_left_ \l__enumext_level: _tl }
2798     \tl_use:c { \l__enumext_label_font_style_ \l__enumext_level: _tl }
2799     \bool_if:cTF { \l__enumext_wrap_label_ \l__enumext_level: _bool }
2800     {
2801       \__enumext_item_starred:

```

```

2802         \use:c { __enumext_wrapper_label_ \_enumext_level: :n } { ##1 }
2803     }
2804     { ##1 }
2805     \tl_use:c { l__enumext_label_fill_right_ \_enumext_level: _tl }
2806     \tl_gclear:N \g__enumext_item_symbol_tl
2807 }
2808 }

```

(End of definition for `_enumext_make_label:`)

11.31.2 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 (§11.32.2).

```

2809 \cs_new_protected:Nn \_enumext_keyans_make_label:
2810 {
2811     \RenewDocumentCommand \makeLabel { m }
2812     {
2813         \tl_use:N \l__enumext_label_fill_left_v_tl
2814         \tl_use:N \l__enumext_label_font_style_v_tl
2815         \bool_if:NTF \l__enumext_wrap_label_v_bool
2816         {
2817             \_enumext_wrapper_label_v:n { ##1 }
2818         }
2819         { ##1 }
2820         \tl_use:N \l__enumext_label_fill_right_v_tl
2821     }
2822 }

```

(End of definition for `_enumext_keyans_make_label:`)

11.32 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`.

11.32.1 Calculation of `\leftmargin` and `\itemindent`

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

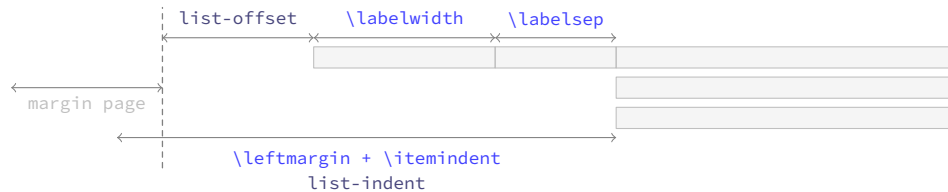


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

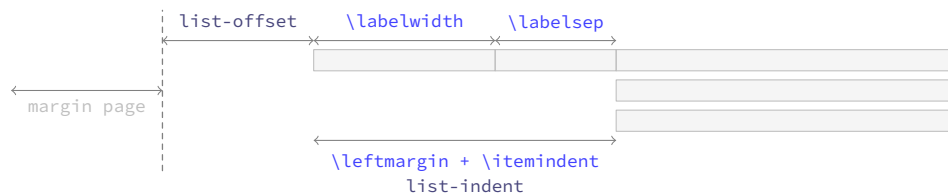


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

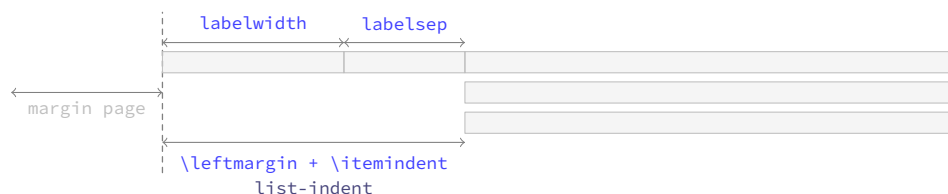


Figure 11: Default horizontal lengths in `enumext`.

```
\__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 (§11.32.2).

```
2823 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2824 {
2825   \dim_compare:nNt { #1 } < { \c_zero_dim }
2826   {
2827     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2828     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2829   }
2830   \dim_compare:nNt { #2 } < { \c_zero_dim }
2831   {
2832     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2833     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2834   }
2835 }
```

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

```
2835   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2836   \dim_compare:nNtF { #4 } < { \c_zero_dim }
2837   {
2838     \dim_set:Nn #6 { #1 + #2 - #4 }
2839     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2840   }
2841   {
2842     \dim_compare:nNt { #4 } = { #1 + #2 }
2843     { \dim_set:Nn #6 { \c_zero_dim } }
2844     \dim_compare:nNt { #4 } < { #1 + #2 }
2845     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2846     \dim_compare:nNt { #4 } > { #1 + #2 }
2847     {
2848       \dim_set:Nn #6 { -#1 - #2 + #4 }
2849       \dim_set:Nn #6 { #6*-1 }
2850     }
2851     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2852   }
2853 }
2854 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

11.32.2 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.

```
2855 \cs_set_protected:Npn \__enumext_tmp:n #1
2856 {
2857   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2858   {
2859     \__enumext_calc_hspace:ccccccc
2860     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2861     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2862     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2863     { \__enumext_leftmargin_tmp_#1_bool }
2864     \clist_map_inline:nn
2865     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2866     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2867     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2868     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2869     \usecounter { enumX#1 }
2870     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
```



```

2871 \str_if_eq:nnTF {#1} { v }
2872 {
2873   \__enumext_keyans_redefine_item:
2874   \__enumext_keyans_make_label:
2875   \__enumext_keyans_ref:
2876   \__enumext_keyans_fake_item:
2877   \bool_if:cT { l__enumext_show_length_#1_bool }
2878   {
2879     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2880   }
2881 }
2882 {
2883   \__enumext_redefine_item:
2884   \__enumext_make_label:
2885   \__enumext_standar_ref:
2886   \__enumext_fake_item:
2887   \bool_if:cT { l__enumext_show_length_#1_bool }
2888   {
2889     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
2890   }
2891 }
2892 }
2893 }
2894 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

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

```

2895 \cs_set_protected:Npn \__enumext_tmp:n #1
2896 {
2897   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2898   {
2899     \__enumext_calc_hspace:ccccc
2900     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2901     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2902     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2903     { l__enumext_leftmargin_tmp_#1_bool }
2904     \clist_map_inline:nn
2905     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2906     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2907     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2908     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2909     \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
2910     \skip_zero:N \partopsep
2911     \usecounter { enumX#1 }
2912     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2913     \__enumext_starred_ref:
2914     \str_if_eq:nnTF {#1} { vii }
2915     {
2916       \__enumext_fake_item_vii:
2917       \bool_if:cT { l__enumext_show_length_vii_bool }
2918       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2919     }
2920     {
2921       \__enumext_fake_item_viii:
2922       \bool_if:cT { l__enumext_show_length_#1_bool }
2923       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2924     }
2925   }
2926 }
2927 \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:.)

11.33 The environment `enumext`

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

```

2928 \NewDocumentEnvironment{enumext}{0}{ }
2929 {

```

```

2930     \__enumext_safe_exec:
2931     \__enumext_parse_keys:n {#1}
2932     \__enumext_before_list:
2933     \__enumext_start_store_level:
2934     \__enumext_start_list:nn
2935     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2936     {
2937         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2938         \__enumext_before_keys_exec:
2939     }
2940     \__enumext_after_args_exec:
2941 }
2942 {
2943     \__enumext_stop_list:
2944     \__enumext_stop_store_level:
2945     \__enumext_after_list:
2946 }

```

(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:`.

```

2947 \cs_new_protected:Nn \__enumext_safe_exec:
2948 {
2949     \__enumext_internal_mini_page:
2950     \__enumext_is_not_nested:
2951     \int_incr:N \l__enumext_level_int
2952     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2953     { \msg_fatal:nn { enumext } { list-too-deep } }
2954     \bool_set_true:N \l__enumext_standar_bool
2955     \__enumext_is_on_first_level:
2956 }

```

(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`.

```

2957 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2958 {
2959     \tl_if_novalue:nF {#1}
2960     {
2961         \str_clear:N \l__enumext_series_str
2962         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2963         {
2964             \keys_set:nn { enumext / level-1 } {#1}
2965             \__enumext_parse_series:n {#1}
2966         }
2967         {
2968             \exp_args:Ne \keys_set:nn
2969             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2970         }
2971         \__enumext_store_active_keys:n {#1}
2972     }
2973 }

```

(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.

```

2974 \cs_new_protected:Nn \__enumext_start_store_level:

```

```

2975 {
2976   \bool_lazy_all:nT
2977   {
2978     { \bool_if_p:N \l__enumext_store_active_bool }
2979     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2980     { \bool_not_p:n { \g__enumext_starred_bool } }
2981   }
2982   {
2983     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2984     {
2985       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2986       \__enumext_store_level_open:
2987     }
2988   }
2989   \bool_lazy_all:nT
2990   {
2991     { \bool_if_p:N \l__enumext_store_active_bool }
2992     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2993     { \bool_if_p:N \g__enumext_starred_bool }
2994   }
2995   {
2996     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2997     {
2998       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2999       \__enumext_store_level_open:
3000     }
3001   }
3002 }
3003 \cs_new_protected:Nn \__enumext_stop_store_level:
3004 {
3005   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3006   {
3007     \__enumext_store_level_close:
3008   }
3009 }

```

(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.

```

3010 \cs_new_protected:Nn \__enumext_before_list:
3011 {
3012   \__enumext_vspace_above:
3013   \__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.

```

3014   \__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.

```

3015   \dim_compare:nNnT
3016   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3017   {
3018     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3019     {
3020       \linewidth
3021       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3022       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3023     }

```

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

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

```

3024         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3025         \int_gincr:N \g__enumext_minipage_stat_int
3026         \__enumext_mini_addvspace:
3027         \nointerlineskip\noindent
3028         \begin{__enumext_mini_env*}
3029         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3030     }
3031     \__enumext_multicols_start:
3032 }

```

(End of definition for __enumext_before_list:.)

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

```

3033 \cs_new_protected:Nn \__enumext_multicols_start:
3034 {
3035     \int_compare:nNt
3036     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3037     {
3038         \dim_compare:nNt
3039         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3040         {
3041             \dim_set_eq:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3042             {
3043                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3044                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3045                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3046                   - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3047             }
3048         }
3049         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3050         \skip_zero:N \multicolsep
3051         \int_compare:nNt { \l__enumext_level_int } > { 1 }
3052         {
3053             \dim_zero:N \columnseprule
3054         }
3055     }

```

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.

```

3055         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3056         {
3057             \__enumext_multi_addvspace:
3058         }
3059         \raggedcolumns
3060         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3061     }
3062 }

```

(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.

```

3063 \cs_new_protected:Nn \__enumext_multicols_stop:
3064 {
3065     \int_compare:nNt
3066     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3067     {
3068         \end{multicols}
3069         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3070         {
3071             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3072         }
3073     }
3074 }

```

(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.

```

3075 \cs_new_protected:Nn \__enumext_after_list:
3076 {
3077   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3078   {
3079     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3080     {
3081       \msg_warning:nn { enumext } { missing-miniright }
3082       \miniright
3083     }
3084     \int_gzero:N \g__enumext_minipage_stat_int
3085     \end{\__enumext_mini_env*}
3086     \par\addvspace { \l__enumext_minipage_after_skip }
3087   }
3088   { \__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`.

```

3089   \__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.

```

3090   \__enumext_after_stop_list:
3091   \__enumext_vspace_below:
3092   \bool_set_false:N \l__enumext_standar_bool
3093   \__enumext_resume_save_counter:
3094 }

```

(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`.

```

3095 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

11.34 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.

```

3096 \NewDocumentEnvironment{keyans}{0}{}
3097 {
3098   \__enumext_keyans_safe_exec:
3099   \__enumext_keyans_parse_keys:n {#1}
3100   \__enumext_before_list_v:
3101   \__enumext_start_list:nn
3102   { \tl_use:N \l__enumext_label_v_tl }
3103   {
3104     \__enumext_list_arg_two_v:
3105     \__enumext_before_keys_exec_v:
3106   }
3107   \__enumext_after_args_exec_v:
3108 }
3109 {
3110   \__enumext_check_starred_cmd:n { item }
3111   \__enumext_stop_list:
3112   \__enumext_after_list_v:
3113 }

```

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

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

```

3114 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3115 {

```

```

3116 \bool_if:NF \l__enumext_store_active_bool
3117 {
3118   \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3119 }
3120 \int_incr:N \l__enumext_keyans_level_int
3121 \bool_set_true:N \l__enumext_keyans_env_bool
3122 \__enumext_keyans_save_start_line:
3123 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3124 \bool_set_false:N \l__enumext_store_active_bool
3125 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3126 {
3127   \msg_error:nn { enumext } { keyans-nested }
3128 }
3129 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3130 {
3131   \msg_error:nn { enumext } { keyans-wrong-level }
3132 }
3133 }

```

(End of definition for __enumext_keyans_safe_exec:.)

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

```

3134 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3135 {
3136   \keys_set:nn { enumext / keyans } {#1}
3137 }

```

(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.

```

3138 \cs_new_protected:Nn \__enumext_before_list_v:
3139 {
3140   \__enumext_vspace_above_v:
3141   \__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.

```

3142 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3143 {
3144   \dim_set:Nn \l__enumext_minipage_left_v_dim
3145   {
3146     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3147   }

```

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*.

```

3148 \bool_set_true:N \l__enumext_minipage_active_v_bool
3149 \int_gincr:N \g__enumext_minipage_stat_int
3150 \__enumext_keyans_mini_addvspace:
3151 \nointerlineskip\noindent
3152 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3153 }

```

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

```

3154 \__enumext_keyans_multicols_start:
3155 }

```

(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.

```

3156 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3157 {
3158   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3159   {

```

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

```

3160     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3161     {
3162       \dim_set:Nn \__enumext_columns_sep_v_dim
3163       {
3164         (
3165           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3166         ) / \__enumext_columns_v_int
3167         - \__enumext_listoffset_v_dim
3168       }
3169     }
3170     \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).

```

3171     \skip_zero:N \multicolsep
3172     \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.

```

3173     \bool_if:NF \__enumext_minipage_active_v_bool
3174     {
3175       \__enumext_keyans_multi_addvspace:
3176     }
3177     \raggedcolumns
3178     \begin{multicols}{\__enumext_columns_v_int }
3179   }
3180 }

```

(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.

```

3181 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3182 {
3183   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3184   {
3185     \end{multicols}
3186     \bool_if:NF \__enumext_minipage_active_v_bool
3187     {
3188       \par\addvspace{ \__enumext_multicols_below_v_skip }
3189     }
3190   }
3191 }

```

(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.

```

3192 \cs_new_protected:Nn \__enumext_after_list_v:
3193 {
3194   \bool_if:NTF \__enumext_minipage_active_v_bool
3195   {
3196     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3197     {
3198       \msg_warning:nn { enumext } { missing-miniright }
3199       \miniright
3200     }
3201     \int_gzero:N \g__enumext_minipage_stat_int

```



```

3202         \end{__enumext_mini_env*}
3203         \par\addvspace{ \l__enumext_minipage_after_skip }
3204     }
3205     { __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.

```

3206     \bool_set_false:N \l__enumext_keyans_env_bool
3207     \__enumext_after_stop_list_v:
3208     \__enumext_vspace_below_v:
3209 }

```

(End of definition for `__enumext_after_list_v:`)

11.35 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.

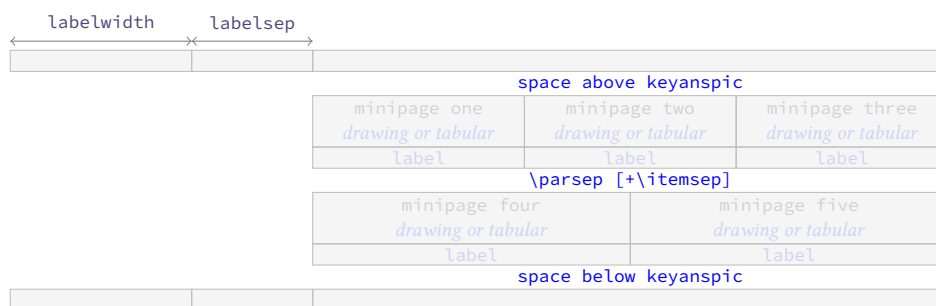


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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?](#).

11.35.1 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.

```

3210 \NewDocumentCommand \anspic { s o +m }
3211 {

```

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

```

3212     \bool_if:NF \l__enumext_store_active_bool
3213     {
3214         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3215     }
3216     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3217     {
3218         \msg_error:nn { enumext } { keyanspic-wrong-level }
3219     }
3220     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3221     {
3222         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3223     }

```

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.

```

3224     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3225     {
3226         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3227     }
3228 }

```

(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.

```

3229 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3230 {
3231   \stepcounter { enumXvi }
3232   #3 \\\
3233   \bool_if:nT { #1 }
3234   {
3235     \__enumext_keyans_addto_prop:n { #2 }
3236     \__enumext_keyans_store_ref:
3237     \__enumext_keyans_addto_seq:n { #2 }
3238     \int_gincr:N \g__enumext_check_starred_cmd_int
3239     \bool_lazy_or:nnT
3240     { \bool_if_p:N \l__enumext_show_answer_bool }
3241     { \bool_if_p:N \l__enumext_show_position_bool }
3242     {
3243       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3244       \__enumext_keyans_show_left:n { #2 }
3245       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3246     }
3247   }
3248   \tl_use:N \l__enumext_label_font_style_v_tl
3249   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3250 }

```

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

11.35.2 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.

```

3251 \NewDocumentEnvironment{keyanspic}{ o }
3252 {
3253   \__enumext_keyans_pic_safe_exec:
3254   \__enumext_start_list:nn
3255   { }
3256   {
3257     \__enumext_keyans_pic_arg_two:
3258   }

```

We apply the “adjusted” vertical spacing above the environment

```

3259   \vspace { \l__enumext_keyans_pic_above_skip }
3260 }

```

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.

```

3261 {
3262   \tl_if_novalue:nTF { #1 }
3263   {
3264     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3265   }
3266   { \__enumext_keyans_pic_do:n { #1 } }
3267   \__enumext_stop_list:
3268   \__enumext_check_starred_cmd:n { anspic }
3269   \setcounter { enumXvi } { 0 }
3270   \vspace { \l__enumext_topsep_v_skip }
3271   %\bool_set_false:N \l__enumext_store_active_bool
3272 }

```

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

`__enumext_keyans_pic_safe_exec:`

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

```

3273 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3274 {
3275   \int_incr:N \l__enumext_keyans_pic_level_int
3276   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3277   {
3278     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3279     }
3280     \__enumext_keyans_save_start_line:
3281 }

```

(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.

```

3282 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3283 {
3284     \dim_compare:nNnT { #1 } < { 0pt }
3285     { \skip_set:Nn #1 { -#1 } }
3286 }

```

(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.

```

3287 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3288 {

```

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.

```

3289     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3290     \__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.

```

3291     \skip_add:Nn \parsep { \itemsep }
3292     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3293     \dim_zero:N \labelwidth
3294     \dim_zero:N \listparindent
3295     \dim_zero:N \labelsep
3296     \skip_zero:N \partopsep
3297     \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.

```

3298     \__enumext_keyans_pic_skip_abs:N \parsep
3299     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3300     {
3301         \box_dp:N \strutbox
3302         + \l__enumext_topsep_v_skip
3303         - \parsep
3304     }
3305     \__enumext_item_std:w \scan_stop:
3306 }

```

(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.

```

3307 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3308 {
3309     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3310 }
3311 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

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

```

3312 \cs_new_protected:Npn \__enumext_keyans_pic_row:n
3313 {
3314     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3315     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3316     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }

```

```

3317 \int_step_inline:nnn
3318 { \l__enumext_keyans_pic_above_int + 1 }
3319 { \l__enumext_keyans_pic_below_int }
3320 {
3321   \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3322   \centering
3323   \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3324   \__enumext_endminipage:
3325 }
3326 \par
3327 }

```

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

11.36 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`).

11.36.1 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.

```

3328 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3329 {
3330   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3331   {
3332     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3333     {
3334       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3335       / \l__enumext_columns_vii_int
3336     }
3337   }
3338   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3339   \dim_set:Nn \l__enumext_item_width_vii_dim
3340   {
3341     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3342     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3343     - \l__enumext_labelsep_vii_dim
3344   }
3345   \dim_zero_new:N \itemwidth
3346 }

```

(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`.

```

3347 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3348 {
3349   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3350   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3351   {
3352     \msg_warning:nnee { enumext } { item-joined }
3353     { \int_use:N \l__enumext_joined_item_vii_int }
3354     { \int_use:N \l__enumext_columns_vii_int }
3355     \int_set:Nn \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   }
3360   \int_compare:nNnT
3361   { \l__enumext_joined_item_vii_int }
3362   >
3363   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```

```

3364 {
3365     \msg_warning:nnee { enumext } { item-joined-columns }
3366     { \int_use:N \l__enumext_joined_item_vii_int }
3367     {
3368         \int_eval:n
3369         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3370     }
3371     \int_set:Nn \l__enumext_joined_item_vii_int
3372     {
3373         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3374     }
3375 }

```

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

```

3376 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3377 {
3378     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3379     \int_decr:N \l__enumext_joined_item_aux_vii_int
3380     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3381     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3382     \dim_set:Nn \l__enumext_joined_width_vii_dim
3383     {
3384         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3385         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3386             + \l__enumext_columns_sep_vii_dim
3387             ) * \l__enumext_joined_item_aux_vii_int
3388     }
3389     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3390 }
3391 {
3392     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3393     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3394 }
3395 }

```

(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`.

```

3396 \cs_new_protected:Nn \__enumext_start_mini_vii:
3397 {
3398     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3399     {
3400         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3401         {
3402             \linewidth
3403             - \l__enumext_minipage_right_vii_dim
3404             - \l__enumext_minipage_hsep_vii_dim
3405         }
3406         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3407         \dim_gset_eq:NN
3408             \g__enumext_minipage_right_vii_dim
3409             \l__enumext_minipage_right_vii_dim
3410         \__enumext_mini_addvspace_vii:
3411         \nointerlineskip\noindent
3412         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3413     }
3414 }

```

(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”.

```

3415 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3416 {
3417     \bool_if:NT \l__enumext_minipage_active_vii_bool

```

```

3418     {
3419         \end{__enumext_mini_env*}
3420         \hfill
3421         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3422     }
3423 }

```

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”.

```

3424 \__enumext_after_env:nn {enumext*}
3425 {
3426     \bool_if:NT \g__enumext_minipage_active_vii_bool
3427     {
3428         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3429         \par\addvspace { \g__enumext_minipage_right_skip }
3430         \bool_if:NF \g__enumext_minipage_center_vii_bool
3431         {
3432             \centering
3433         }
3434         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3435         \end{__enumext_mini_env*}
3436         \par\addvspace{ \g__enumext_minipage_after_skip }
3437     }
3438     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3439     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3440     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3441     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3442     \bool_gset_false:N \g__enumext_starred_bool
3443 }

```

(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.

```

3444 \NewDocumentEnvironment{enumext*}{ o }
3445 {
3446     \__enumext_safe_exec_vii:
3447     \__enumext_parse_keys_vii:n {#1}
3448     \__enumext_before_list_vii:
3449     \__enumext_start_store_level_vii:
3450     \__enumext_start_list:nn { }
3451     {
3452         \__enumext_list_arg_two_vii:
3453         \__enumext_before_keys_exec_vii:
3454     }
3455     \__enumext_starred_columns_set_vii:
3456     \item[] \scan_stop:
3457     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3458     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3459 }
3460 {
3461     \__enumext_stop_item_tmp_vii:
3462     \__enumext_remove_extra_parsep_vii:
3463     \__enumext_stop_list:
3464     \__enumext_stop_store_level_vii:
3465     \__enumext_after_list_vii:
3466 }

```

(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`.

```

3467 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3468 {
3469     \__enumext_internal_mini_page:
3470     \__enumext_is_not_nested:
3471     \int_incr:N \l__enumext_level_h_int
3472     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3473     {
3474         \msg_error:nn { enumext } { nested }

```

```

3475     }
3476     \bool_set_true:N \l__enumext_starred_bool
3477     \__enumext_is_on_first_level:
3478 }

```

(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*.

```

3479 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3480 {
3481     \tl_if_novalue:nF {#1}
3482     {
3483         \str_clear:N \l__enumext_series_str
3484         \keys_set:nn { enumext / enumext* } {#1}
3485         \__enumext_parse_series:n {#1}
3486         \__enumext_store_active_keys_vii:n {#1}
3487     }
3488 }

```

(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**.

```

3489 \cs_new_protected:Nn \__enumext_before_list_vii:
3490 {
3491     \__enumext_vspace_above_vii:
3492     \__enumext_check_ans_active:
3493     \__enumext_before_args_exec_vii:
3494     \__enumext_start_mini_vii:
3495 }

```

(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.

```

3496 \cs_new_protected:Nn \__enumext_after_list_vii:
3497 {
3498     \__enumext_stop_mini_vii:
3499     \__enumext_after_stop_list_vii:
3500     \__enumext_check_ans_key_hook:
3501     \__enumext_vspace_below_vii:
3502     \bool_set_false:N \l__enumext_starred_bool
3503     \__enumext_resume_save_counter:
3504 }

```

(End of definition for __enumext_after_list_vii:.)

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

```

3505 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3506 {
3507     \bool_if:NT \l__enumext_store_active_bool
3508     {
3509         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3510         {
3511             \__enumext_store_level_open_vii:
3512         }
3513     }
3514 }
3515 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3516 {
3517     \bool_if:NT \l__enumext_store_active_bool

```



```

3518     {
3519         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3520         {
3521             \__enumext_store_level_close_vii:
3522         }
3523     }
3524 }

```

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

11.36.2 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.

```

3525 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3526 {
3527     \__enumext_stop_item_tmp_vii:
3528     \int_incr:N \l__enumext_item_column_pos_vii_int
3529     \int_gincr:N \g__enumext_item_count_all_vii_int
3530     \__enumext_item_peek_args_vii:
3531 }

```

(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).

```

3532 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3533 {
3534     \peek_meaning:NTF (
3535         { \__enumext_joined_item_vii:w }
3536         { \__enumext_joined_item_vii:w (1) }
3537     }

```

(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.

```

3538 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3539 {
3540     \__enumext_starred_joined_item_vii:n {#1}
3541     \peek_meaning_remove:NTF *
3542     { \__enumext_starred_item_vii:w }
3543     { \__enumext_standar_item_vii:w }
3544 }

```

(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].

```

3545 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3546 {
3547     \bool_set_false:N \l__enumext_item_starred_vii_bool
3548     \peek_meaning:NTF [
3549     {
3550         \bool_set_eq:NN
3551         \l__enumext_wrap_label_vii_bool
3552         \l__enumext_wrap_label_opt_vii_bool
3553         \__enumext_start_item_vii:w

```

```

3554     }
3555     {
3556         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3557         \legacy_if_set_true:n { @noitemarg }
3558         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3559     }
3560 }

```

(End of definition for __enumext_standar_item_vii:w)

The function __enumext_starred_item_vii:w together with the specified auxiliary functions aux_i:w, aux_ii:w, and aux_iii:w execute \item*, \item*[\langle symbol \rangle] and \item*[\langle symbol \rangle][\langle offset \rangle].

```

\__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
3561 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3562 {
3563     \bool_set_true:N \l__enumext_item_starred_vii_bool
3564     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3565     \peek_meaning:NTF [
3566         { \__enumext_starred_item_vii_aux_i:w }
3567         { \__enumext_starred_item_vii_aux_ii:w }
3568     }
3569     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3570     {
3571         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3572         \__enumext_starred_item_vii_aux_ii:w
3573     }
3574     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3575     {
3576         \peek_meaning:NTF [
3577             { \__enumext_starred_item_vii_aux_iii:w }
3578             {
3579                 \dim_set_eq:NN
3580                 \l__enumext_item_symbol_sep_vii_dim
3581                 \l__enumext_labelsep_vii_dim
3582                 \legacy_if_set_true:n { @noitemarg }
3583                 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3584             }
3585         }
3586     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3587     {
3588         \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3589         \legacy_if_set_true:n { @noitemarg }
3590         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3591     }

```

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

11.36.3 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.

```

3592 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3593 {
3594     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3595     \legacy_if:nT { @noitemarg }
3596     {
3597         \legacy_if_set_false:n { @noitemarg }
3598         \legacy_if:nT { @nmbrrlist }
3599         {
3600             \bool_if:NT \l__enumext_hyperref_bool
3601             {
3602                 \legacy_if_set_true:n { @hyper@item }
3603             }
3604             \refstepcounter{enumXvii}
3605             \bool_if:NT \l__enumext_check_answers_bool
3606             {
3607                 \int_gincr:N \g__enumext_item_number_int

```

```

3608     }
3609   }
3610 }

```

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.

```

3611 \group_begin:
3612   \lrbox{ \l__enumext_item_text_vii_box }
3613   \bool_if:NF \l__enumext_footnotes_key_bool
3614   {
3615     \__enumext_renew_footnote:
3616   }
3617   \bool_if:NT \l__enumext_item_starred_vii_bool
3618   {
3619     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3620     {
3621       \tl_gset_eq:NN
3622         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3623     }
3624     \mode_leave_vertical:
3625     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3626     \makebox[ \opt ]{ r }{ \g__enumext_item_symbol_aux_vii_tl }
3627     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3628     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3629   }
3630   \group_begin:
3631     \tl_use:N \l__enumext_label_font_style_vii_tl
3632     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3633     {
3634       \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }
3635       { \__enumext_wrapper_label_vii:n {#1} }
3636     }
3637     {
3638       \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }{ #1 }
3639     }
3640   \group_end:
3641   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3642   \tl_use:N \l__enumext_after_list_args_vii_tl
3643   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3644     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3645     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3646     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3647 }

```

(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.

```

3648 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3649 {
3650   \__enumext_endminipage:
3651   \endlrbox
3652   \group_end:
3653   \box_set_wd:Nn \l__enumext_item_text_vii_box
3654   {
3655     \l__enumext_joined_width_vii_dim
3656     + \l__enumext_labelwidth_vii_dim
3657     + \l__enumext_labelsep_vii_dim
3658   }
3659   \int_set:Nn \hbadness { 10000 }
3660   \box_use:N \l__enumext_item_text_vii_box
3661   \bool_if:NF \l__enumext_footnotes_key_bool
3662   {
3663     \__enumext_print_footnote:
3664   }

```

```

3665 \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3666 {
3667   \par\noindent
3668   \int_zero:N \l__enumext_item_column_pos_vii_int
3669 }
3670 { \hspace{ \l__enumext_columns_sep_vii_dim } }
3671 }

```

(End of definition for `__enumext_stop_item_vii:`)

`__enumext_remove_extra_parsep_vii:`

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

```

3672 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3673 {
3674   \int_compare:nNnT
3675   {
3676     \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3677   }
3678   =
3679   { \c_zero_int }
3680   {
3681     \par
3682     \vspace{ -\l__enumext_itemsep_vii_skip }
3683     \int_gzero:N \g__enumext_item_count_all_vii_int
3684   }
3685 }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`)

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

```

3686 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

11.37 The environment `keyans*`

11.37.1 Functions for item box width

`__enumext_starred_columns_set_viii:`

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

```

3687 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3688 {
3689   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3690   {
3691     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3692     {
3693       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3694       / \l__enumext_columns_viii_int
3695     }
3696   }
3697   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3698   \dim_set:Nn \l__enumext_item_width_viii_dim
3699   {
3700     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3701     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3702     - \l__enumext_labelsep_viii_dim
3703   }
3704   \dim_zero_new:N \itemwidth
3705 }

```

(End of definition for `__enumext_starred_columns_set_viii:`)

`__enumext_starred_joined_item_viii:n`

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

```

3706 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3707 {
3708   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3709   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3710   {
3711     \msg_warning:nnee { enumext } { item-joined }
3712     { \int_use:N \l__enumext_joined_item_viii_int }
3713     { \int_use:N \l__enumext_columns_viii_int }
3714     \int_set:Nn \l__enumext_joined_item_viii_int

```

```

3715         {
3716             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3717         }
3718     }
3719     \int_compare:nNnT
3720     { \l__enumext_joined_item_viii_int }
3721     >
3722     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3723     {
3724         \msg_warning:nnee { enumext } { item-joined-columns }
3725         { \int_use:N \l__enumext_joined_item_viii_int }
3726         {
3727             \int_eval:n
3728             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3729         }
3730         \int_set:Nn \l__enumext_joined_item_viii_int
3731         {
3732             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3733         }
3734     }
3735     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3736     {
3737         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3738         \int_decr:N \l__enumext_joined_item_aux_viii_int
3739         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3740         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3741         \dim_set:Nn \l__enumext_joined_width_viii_dim
3742         {
3743             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3744             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3745               + \l__enumext_columns_sep_viii_dim
3746               ) * \l__enumext_joined_item_aux_viii_int
3747         }
3748         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3749     }
3750     {
3751         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3752         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3753     }
3754 }

```

(End of definition for `__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.

```

3755 \cs_new_protected:Nn \__enumext_start_mini_viii:
3756 {
3757     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3758     {
3759         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3760         {
3761             \linewidth
3762             - \l__enumext_minipage_right_viii_dim
3763             - \l__enumext_minipage_hsep_viii_dim
3764         }
3765         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3766         \dim_gset_eq:NN
3767         \g__enumext_minipage_right_viii_dim
3768         \l__enumext_minipage_right_viii_dim
3769         \__enumext_mini_addvspace_viii:
3770         \nointerlineskip\noindent
3771         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3772     }
3773 }
3774 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3775 {
3776     \bool_if:NT \l__enumext_minipage_active_viii_bool
3777     {
3778         \end{__enumext_mini_env*}
3779         \hfill
3780         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3781     }

```

```

3782     }
3783     \__enumext_after_env:nn {keyans*}
3784     {
3785         \bool_if:NT \g__enumext_minipage_active_viii_bool
3786         {
3787             \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3788             \par\addvspace { \g__enumext_minipage_right_skip }
3789             \bool_if:NF \g__enumext_minipage_center_viii_bool
3790             {
3791                 \centering
3792             }
3793             \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3794             \end{__enumext_mini_env*}
3795             \par\addvspace{ \g__enumext_minipage_after_skip }
3796         }
3797         \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3798         \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3799         \tl_gclear:N \g__enumext_miniright_code_viii_tl
3800         \dim_gzero:N \g__enumext_minipage_right_viii_dim
3801     }

```

(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.

```

3802 \NewDocumentEnvironment{keyans*}{ o }
3803 {
3804     \__enumext_safe_exec_viii:
3805     \__enumext_parse_keys_viii:n {#1}
3806     \__enumext_before_list_viii:
3807     \__enumext_start_list:nn { }
3808     {
3809         \__enumext_list_arg_two_viii:
3810         \__enumext_before_keys_exec_viii:
3811     }
3812     \__enumext_starred_columns_set_viii:
3813     \item[] \scan_stop:
3814     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3815     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3816 }
3817 {
3818     \__enumext_stop_item_tmp_viii:
3819     \__enumext_remove_extra_parsep_viii:
3820     \__enumext_check_starred_cmd:n { item }
3821     \__enumext_stop_list:
3822     \__enumext_after_list_viii:
3823 }

```

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

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

```

3824 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3825 {
3826     \int_incr:N \l__enumext_keyans_level_h_int
3827     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3828     {
3829         \msg_error:nn { enumext } { nested }
3830     }
3831     \__enumext_keyans_save_start_line:
3832     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3833     \bool_set_false:N \l__enumext_store_active_bool
3834     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3835     {
3836         \msg_error:nn { enumext } { keyans-wrong-level }
3837     }
3838 }

```

(End of definition for __enumext_safe_exec_viii:.)

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

```

3839 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3840 {
3841   \tl_if_novalue:nF {#1}
3842   {
3843     \keys_set:nn { enumext / keyans* } {#1}
3844   }
3845 }

```

(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*.

```

3846 \cs_new_protected:Nn \__enumext_before_list_viii:
3847 {
3848   \__enumext_vspace_above_viii:
3849   \__enumext_before_args_exec_viii:
3850   \__enumext_start_mini_viii:
3851 }

```

(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.

```

3852 \cs_new_protected:Nn \__enumext_after_list_viii:
3853 {
3854   \__enumext_stop_mini_viii:
3855   \__enumext_after_stop_list_viii:
3856   \__enumext_vspace_below_viii:
3857 }

```

(End of definition for `__enumext_after_list_viii:`.)

11.37.2 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`.

```

3858 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3859 {
3860   \__enumext_stop_item_tmp_viii:
3861   \int_incr:N \l__enumext_item_column_pos_viii_int
3862   \int_gincr:N \g__enumext_item_count_all_viii_int
3863   \__enumext_item_peek_args_viii:
3864 }

```

(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).

```

3865 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3866 {
3867   \peek_meaning:NTF (
3868     { \__enumext_joined_item_viii:w }
3869     { \__enumext_joined_item_viii:w (1) }
3870   }

```

(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`.

```

3871 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3872 {
3873   \__enumext_starred_joined_item_viii:n {#1}
3874   \peek_meaning_remove:NTF *
3875   { \__enumext_starred_item_viii:w }
3876   { \__enumext_standar_item_viii:w }
3877 }

```

(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]`.

```

3878 \cs_new_protected:Npn \__enumext_standar_item_viii:w
3879 {
3880   \bool_set_false:N \l__enumext_item_starred_viii_bool
3881   \peek_meaning:NTF [
3882   {
3883     \bool_set_eq:NN
3884     \l__enumext_wrap_label_viii_bool
3885     \l__enumext_wrap_label_opt_viii_bool
3886     \__enumext_start_item_viii:w
3887   }
3888   {
3889     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3890     \legacy_if_set_true:n { @noitemarg }
3891     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3892   }
3893 }

```

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

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

```

\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
3894 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3895 {
3896   \bool_set_true:N \l__enumext_item_starred_viii_bool
3897   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3898   \peek_meaning:NTF [
3899   { \__enumext_starred_item_viii_aux_i:w }
3900   { \__enumext_starred_item_viii_aux_ii:w }
3901 }

```

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`.

```

3902 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3903 {
3904   \tl_clear:N \l__enumext_store_keyans_label_tl
3905   \tl_if_no_value:nF { #1 }
3906   {
3907     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3908     {
3909       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3910       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3911     }
3912     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3913   }
3914   \__enumext_starred_item_viii_aux_ii:w
3915 }

```

```

3916 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3917 {
3918   \legacy_if_set_true:n { @noitemarg }
3919   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3920 }

```

(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.

```

3921 \cs_new_protected:Nn \__enumext_starred_item_exec:
3922 {
3923   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3924   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3925   \__enumext_keyans_store_ref:
3926   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3927   \__enumext_keyans_addto_seq_link:
3928   \int_gincr:N \g__enumext_check_starred_cmd_int
3929   \bool_if:NT \__enumext_show_answer_bool
3930   {
3931     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3932   }
3933   \bool_if:NT \__enumext_show_position_bool
3934   {
3935     \tl_set:Ne \__enumext_mark_answer_sym_tl
3936     {
3937       \group_begin:
3938       \exp_not:N \normalfont
3939       \exp_not:N \footnotesize [ \int_eval:n
3940         {
3941           \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3942         }
3943       ]
3944       \group_end:
3945     }
3946     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3947   }
3948 }

```

(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.

```

3949 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3950 {
3951   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3952   \legacy_if:nT { @noitemarg }
3953   {
3954     \legacy_if_set_false:n { @noitemarg }
3955     \legacy_if:nT { @nmbrlist }
3956     {
3957       \bool_if:NT \__enumext_hyperref_bool
3958       {
3959         \legacy_if_set_true:n { @hyper@item }
3960       }
3961       \refstepcounter{enumXviii}
3962     }
3963   }

```

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

```

3964   \group_begin:
3965   \lrbox{ \__enumext_item_text_viii_box }
3966   \bool_if:NF \__enumext_footnotes_key_bool
3967   {
3968     \__enumext_renew_footnote:
3969   }
3970   \bool_if:NT \__enumext_item_starred_viii_bool
3971   {

```

```

3972         \__enumext_starred_item_exec:
3973     }
3974     \group_begin:
3975     \tl_use:N \__enumext_label_font_style_viii_tl
3976     \bool_if:NTF \__enumext_wrap_label_viii_bool
3977     {
3978         \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]
3979         { \__enumext_wrapper_label_viii:n {#1} }
3980     }
3981     {
3982         \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]{ #1
3983     }
3984     \group_end:
3985     \skip_horizontal:N \__enumext_labelsep_viii_dim
3986     \tl_use:N \__enumext_after_list_args_viii_tl
3987     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_viii_dim }
3988     \skip_set_eq:NN \parindent \__enumext_listparindent_viii_dim
3989     \skip_set_eq:NN \parskip \__enumext_parsep_viii_skip
3990     \bool_if:NT \__enumext_item_starred_viii_bool
3991     {
3992         \tl_use:N \__enumext_fake_item_indent_viii_tl
3993         \__enumext_keyans_show_item_opt:
3994         \skip_horizontal:n { -\__enumext_fake_item_indent_viii_dim - \__enumext_labelsep_viii_dim }
3995     }
3996     {
3997         \tl_use:N \__enumext_fake_item_indent_viii_tl
3998     }
3999 }

```

(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.

```

4000 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4001 {
4002     \__enumext_endminipage:
4003     \endlrbox
4004     \group_end:
4005     \box_set_wd:Nn \__enumext_item_text_viii_box
4006     {
4007         \__enumext_joined_width_viii_dim
4008         + \__enumext_labelwidth_viii_dim
4009         + \__enumext_labelsep_viii_dim
4010     }
4011     \int_set:Nn \hbadness { 10000 }
4012     \box_use:N \__enumext_item_text_viii_box
4013     \bool_if:NF \__enumext_footnotes_key_bool
4014     {
4015         \__enumext_print_footnote:
4016     }
4017     \int_compare:nNnTF
4018     { \__enumext_item_column_pos_viii_int } = { \__enumext_columns_viii_int }
4019     {
4020         \par\noindent
4021         \int_zero:N \__enumext_item_column_pos_viii_int
4022     }
4023     { \hspace{ \__enumext_columns_sep_viii_dim } }
4024 }

```

(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.

```

4025 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4026 {
4027     \int_compare:nNnTF
4028     {
4029         \int_mod:nn
4030         { \__enumext_item_count_all_viii_int }
4031         { \__enumext_columns_viii_int }

```

```

4032     }
4033     =
4034     { \c_zero_int }
4035     {
4036         \par
4037         \vspace{ -\l__enumext_itemsep_viii_skip }
4038         \int_gzero:N \g__enumext_item_count_all_viii_int
4039     }
4040 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`.)

11.38 The command `\getkeyans`

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

```

4041 \NewDocumentCommand \getkeyans { m }
4042 {
4043     \exp_args:Ne \__enumext_getkeyans_aux:n
4044     { \tl_to_str:e { \text_expand:n {#1} } }
4045 }

```

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

`__enumext_getkeyans_aux:n`

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the `⟨argument⟩` using “:”. If “:” is omitted it will return an error.

```

4046 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4047 {
4048     \str_if_in:nnTF {#1} { : }
4049     {
4050         \use:e
4051         {
4052             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4053             { {##1} {##2} }
4054         }
4055         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4056     }
4057     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4058 }

```

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

`__enumext_getkeyans:nn`

The internal function `__enumext_getkeyans:nn` will check for the existence of the `⟨prop list⟩`, if it does not exist it will return an error message, then it will fetch the content specified by the second `⟨argument⟩` from `⟨prop list⟩`.

```

4059 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4060 {
4061     \prop_if_exist:cF { g__enumext_#1_prop }
4062     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4063     \group_begin:
4064     \prop_item:cn { g__enumext_#1_prop }{#2}
4065     \group_end:
4066 }

```

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

11.39 The command `\printkeyans`

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

The variable `\l__enumext_print_keyans_starred_tl` will have the default `⟨keys⟩` for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the `⟨sequence⟩` and will be set by `\setenumext[⟨print , *⟩]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[⟨print , level⟩]`

```

4067 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4068 \keys_define:nn { enumext / print }
4069 {
4070     print* .code:n = \keys_precompile:neN { enumext / enumext* }

```

```

4071         { \__enumext_filter_save_key:n {#1} }
4072         \__enumext_print_keyans_starred_tl, % starred cmd
4073     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4074     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4075         { \__enumext_filter_save_key:n {#1} }
4076         \__enumext_print_keyans_i_tl,
4077     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4078     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4079         { \__enumext_filter_save_key:n {#1} }
4080         \__enumext_print_keyans_ii_tl,
4081     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4082     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4083         { \__enumext_filter_save_key:n {#1} }
4084         \__enumext_print_keyans_iii_tl,
4085     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4086     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4087         { \__enumext_filter_save_key:n {#1} }
4088         \__enumext_print_keyans_iv_tl,
4089     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4090     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4091         { \__enumext_filter_save_key:n {#1} }
4092         \__enumext_print_keyans_vii_tl, % starred nested
4093     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4094 }

```

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

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

```

4095 \NewDocumentCommand \printkeyans { s O{} m }
4096 {
4097     \group_begin:
4098         \tl_use:N \__enumext_print_keyans_i_tl
4099         \tl_use:N \__enumext_print_keyans_ii_tl
4100         \tl_use:N \__enumext_print_keyans_iii_tl
4101         \tl_use:N \__enumext_print_keyans_iv_tl
4102         \tl_use:N \__enumext_print_keyans_vii_tl
4103         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4104     \group_end:
4105 }

```

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

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

```

4106 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4107 {
4108     \seq_if_exist:cTF { g__enumext_#3_seq }
4109     {
4110         \seq_if_empty:cF { g__enumext_#3_seq }
4111         {
4112             %%\seq_show:c { g__enumext_#3_seq }

```

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

```

4113         \bool_if:nTF {#1}
4114         {
4115             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4116             {
4117                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4118             }
4119         }
4120         \tl_use:N \__enumext_print_keyans_starred_tl
4121         \begin{enumext*}[#2]
4122             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4123         \end{enumext*}

```

```

4124         }
4125     }

```

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

```

4126     {
4127         \begin{enumext}[#2]
4128         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4129         \end{enumext}
4130     }
4131 }
4132 }
4133 {
4134     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4135 }
4136 }

```

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

11.40 The command `\setenumext`

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

```

4137 \keys_define:nn { enumext / meta-families }
4138 {
4139     enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
4140     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
4141     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
4142     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
4143     keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} },
4144     enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} },
4145     keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} },
4146     print*    .code:n = { \keys_set:nn { enumext / print   } { print* = {#1} } },
4147     print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } },
4148     print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } },
4149     print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } },
4150     print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } },
4151     print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } },
4152     unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
4153 }

```

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

```

4154 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4155 {
4156     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4157     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4158 }

```

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

```

4159 \NewDocumentCommand \setenumext { O{enumext,1} +m }
4160 {
4161     \tl_if_novalue:nTF {#1}
4162     {
4163         \seq_map_inline:Nn \c__enumext_all_families_seq
4164     }
4165     {
4166         \seq_clear:N \l__enumext_setkey_tmpa_seq
4167         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4168         \int_set:Nn \l__enumext_setkey_tmpa_int
4169         {
4170             \seq_count:N \l__enumext_setkey_tmpb_seq
4171         }
4172         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4173         {
4174             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4175             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4176             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4177             {
4178                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4179             }
4180         }
4181         {
4182             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }

```

```

4183     }
4184     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4185     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4186     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4187   }
4188   {
4189     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4190   }
4191 }

```

(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.

```

4192 \cs_new_protected:Npn \__enumext_set_parse:n #1
4193 {
4194   \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4195   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4196   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4197   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4198   {
4199     \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
4200     { \tl_trim_spaces:n {#1} }
4201   }
4202   { \__enumext_set_error:nn {#1} { } }
4203 }
4204 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4205 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.41 Messages

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

```

4206 \msg_new:nnn { enumext } { package-load }
4207 {
4208   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4209 }
4210 \msg_new:nnn { enumext } { package-not-load }
4211 {
4212   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4213 }
4214 \msg_new:nnn { enumext } { package-load-foot }
4215 {
4216   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4217 }

```

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

```

4218 \msg_new:nnn { enumext } { counters }
4219 {
4220   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4221   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4222 }

```

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

```

4223 \msg_new:nnn { enumext } { store-prop }
4224 {
4225   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4226 }
4227 \msg_new:nnn { enumext } { store-seq }
4228 {
4229   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4230 }
4231 \msg_new:nnn { enumext } { store-int }
4232 {
4233   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4234 }
4235 \msg_new:nnn { enumext } { prop-seq-int-hook }
4236 {
4237   * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4238   * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4239   * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4240 }
4241 \msg_new:nnn { enumext } { item-answer-hook }

```



```

4242 {
4243     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
4244     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
4245     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4246 }

```

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

```

4247 \msg_new:nnn { enumext } { invalid-key }
4248 {
4249     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4250 }
4251 \msg_new:nnn { enumext } { unknown-key-family }
4252 {
4253     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4254 }

```

Messages used in length calculation.

```

4255 \msg_new:nnn { enumext } { width-negative }
4256 {
4257     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4258     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4259 }
4260 \msg_new:nnn { enumext } { width-zero }
4261 {
4262     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4263     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4264 }

```

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

```

4265 \msg_new:nnn { enumext } { list-lengths }
4266 {
4267     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4268     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4269     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4270     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4271     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4272     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4273     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4274     \__enumext_show_length:nnn { skip } { topsep } {#1}
4275     \__enumext_show_length:nnn { skip } { parsep } {#1}
4276     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4277     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4278     ****
4279 }

```

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

```

4280 \msg_new:nnn { enumext } { list-lengths-not-nested }
4281 {
4282     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4283     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4284     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4285     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4286     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4287     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4288     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4289     \__enumext_show_length:nnn { skip } { topsep } {#1}
4290     \__enumext_show_length:nnn { skip } { parsep } {#1}
4291     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4292     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4293     ****
4294 }

```

Messages used by `ref` key.

```

4295 \msg_new:nnn { enumext } { key-ref-empty }
4296 {
4297     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4298 }

```

Messages used by `save-ans` key.

```

4299 \msg_new:nnn { enumext } { save-ans-empty }
4300 {
4301     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4302 }
4303 \msg_new:nnn { enumext } { save-ans-log }

```

```

4304 {
4305     * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
ans=#2 ~ \msg_line_context:.
4306 }
4307 \msg_new:nnn { enumext } { save-ans-log-hook }
4308 {
4309     * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
ans=#2 ~ \msg_line_context:.
4310 }
4311 \msg_new:nnn { enumext } { save-ans-hook }
4312 {
4313     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4314 }

```

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

```

4315 \msg_new:nnn { enumext } { need-save-ans }
4316 {
4317     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4318 }
4319 \msg_new:nnn { enumext } { items-same-answer }
4320 {
4321     *****\\
4322     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_
4323     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\\
4324     *****
4325 }
4326 \msg_new:nnn { enumext } { item-greater-answer }
4327 {
4328     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4329     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4330     Items ~ > ~ Answers.
4331 }
4332 \msg_new:nnn { enumext } { item-less-answer }
4333 {
4334     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4335     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4336     Items ~ < ~ Answers.
4337 }

```

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

```

4338 \msg_new:nnn { enumext } { missing-starred }
4339 {
4340     Missing ~ '\c_backslash_str #1*' ~ #2.
4341 }
4342 \msg_new:nnn { enumext } { many-starred }
4343 {
4344     Many ~ '\c_backslash_str #1*' ~ #2.
4345 }

```

Messages used by `\printkeyans*` command.

```

4346 \msg_new:nnn { enumext } { print-starred }
4347 {
4348     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4349     #2 ~ environment ~ \msg_line_context:.
4350 }

```

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

```

4351 \msg_new:nnn { enumext } { list-too-deep }
4352 {
4353     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4354     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4355 }

```

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

```

4356 \msg_new:nnn { enumext } { anskey-empty-arg }
4357 {
4358     Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4359 }
4360 \msg_new:nnn { enumext } { anskey-wrong-place }
4361 {
4362     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4363     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4364 }

```

```

4365 \msg_new:nnn { enumext } { anskey-nested }
4366 {
4367   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4368 }
4369 \msg_new:nnn { enumext } { anskey-nested-env }
4370 {
4371   The ~ environment ~ anskey* ~ can't ~ be ~ nested ~ \msg_line_context:.
4372 }
4373 \msg_new:nnn { enumext } { ansPIC-wrong-place }
4374 {
4375   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4376   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4377 }
4378 \msg_new:nnn { enumext } { command-wrong-place }
4379 {
4380   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4381   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4382 }

```

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

```

4383 \msg_new:nnn { enumext } { keyans-nested }
4384 {
4385   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4386 }
4387 \msg_new:nnn { enumext } { keyans-wrong-level }
4388 {
4389   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4390   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4391 }
4392 \msg_new:nnn { enumext } { wrong-place }
4393 {
4394   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4395   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4396 }
4397 \msg_new:nnn { enumext } { keyansPIC-nested }
4398 {
4399   The ~ environment ~ 'keyansPIC' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4400 }
4401 \msg_new:nnn { enumext } { keyansPIC-wrong-level }
4402 {
4403   Wrong ~ level ~ position ~ for ~ 'keyansPIC' ~ \msg_line_context:~ \\
4404   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4405 }

```

Messages used by **\getkeyans** command.

```

4406 \msg_new:nnn { enumext } { undefined-storage-anskey }
4407 {
4408   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4409 }

```

Messages used by **\miniright** command.

```

4410 \msg_new:nnn { enumext } { missing-miniright }
4411 {
4412   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4413   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4414 }
4415 \msg_new:nnn { enumext } { wrong-miniright-place }
4416 {
4417   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4418   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4419 }
4420 \msg_new:nnn { enumext } { wrong-miniright-use }
4421 {
4422   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4423   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4424 }

```

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

```

4425 \msg_new:nnn { enumext } { nested }
4426 {
4427   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4428 }
4429 \msg_new:nnn { enumext } { item-joined }

```

```
4430 {  
4431   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.  
4432 }  
4433 \msg_new:nnn { enumext } { item-joined-columns }  
4434 {  
4435   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4436 }
```

11.42 Finish package

Finish package implementation.

```
4437 \file_input_stop:  
4438 </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, 3232, 4220, 4237, 4238, 4243, 4244, 4257, 4262, 4267, 4282, 4321, 4322, 4323, 4328, 4329, 4334, 4335, 4353, 4362, 4375, 4380, 4389, 4394, 4403, 4412, 4417, 4422
A	
<code>above</code>	<u>1371</u>
<code>above*</code>	<u>1371</u>
<code>\addvspace</code>	1025, 1053, 1169, 1248, 1311, 1317, 1345, 1362, 3071, 3086, 3188, 3203, 3429, 3436, 3788, 3795
<code>after</code>	<u>864</u>
<code>align</code>	<u>481</u>
<code>\Alph</code>	35, 39, <u>40</u>
<code>\Alph</code>	433, 542, 587, 655, 4089
<code>\alph</code>	35, 39, <u>40</u>
<code>\alph</code>	434, 540, 4081
<code>\anskey</code>	12, 70, <u>2200</u>
<code>anskey*</code>	13, <u>2416</u>
<code>\anspic</code>	15, 92, <u>3210</u>
<code>\anspic*</code>	65
<code>\arabic</code>	29, 35
<code>\arabic</code>	432, 539, 586, 4073, 4077, 4093
B	
<code>\baselineskip</code>	47
<code>\baselineskip</code>	2145, 2153
<code>before</code>	<u>864</u>
<code>before*</code>	<u>864</u>
<code>below</code>	<u>1371</u>
<code>below*</code>	<u>1371</u>
bool commands:	
<code>\bool_gset_false:N</code>	308, 309, 310, 3438, 3442, 3797
<code>\bool_gset_true:N</code>	222, 231, 967, 1862, 1868, 3421, 3439, 3780, 3798
<code>\bool_if:NTF</code>	373, 385, 402, 1393, 1407, 1420, 1431, 1442, 1453, 1464, 1475, 1528, 1545, 1550, 1558, 1585, 1623, 1628, 1635, 1639, 1661, 1666, 1674, 1681, 1712, 1720, 1812, 1936, 2018, 2028, 2107, 2131, 2138, 2166, 2204, 2214, 2242, 2268, 2385, 2396, 2400, 2419, 2429, 2477, 2492, 2567, 2578, 2582, 2695, 2725, 2799, 2815, 2877, 2887, 2917, 2922, 3005, 3055, 3069, 3077, 3116, 3173, 3186, 3194, 3212, 3417, 3426, 3430, 3507, 3517, 3600, 3605, 3613, 3617, 3632, 3661, 3776, 3785, 3789, 3929, 3933, 3957, 3966, 3970, 3976, 3990, 4013
<code>\bool_if:nTF</code>	1346, 1363, 2736, 2771, 2835, 3233, 4113
<code>\bool_if_p:N</code>	240, 254, 1692, 1693, 1701, 1702, 1825, 1847, 1859, 1860, 1865, 1866, 2253, 2294, 2295, 2319, 2328, 2329, 2341, 2357, 2554, 2555, 2592, 2593, 2978, 2991, 2993, 3240, 3241
<code>\bool_lazy_all:nTF</code>	238, 252, 1823, 1845, 2317, 2326, 2339, 2355, 2976, 2989
<code>\bool_lazy_and:nnTF</code>	218, 227, 1691, 1700, 1858, 1864, 2252, 2259, 2293, 2553
<code>\bool_lazy_or:nnTF</code>	1753, 1760, 2591, 3239
<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, 2254, 2260, 2344, 2359, 2979, 2980, 2992
<code>\bool_set_eq:NN</code>	2703, 2751, 3550, 3883
<code>\bool_set_false:N</code>	382, 1797, 1798, 3092, 3124, 3206, 3271, 3289, 3502, 3547, 3833, 3880
<code>\bool_set_true:N</code>	245, 259, 364, 368, 474, 792, 1377, 1382, 1648, 1770, 1771, 2050, 2058, 2699, 2729, 2747, 2759, 2954, 2985, 2998, 3024, 3121, 3148, 3406, 3476, 3556, 3563, 3564, 3765, 3889, 3896, 3897
box commands:	
<code>\box_dp:N</code>	1065, 1069, 1073, 1084, 1088, 1099, 1108, 1114, 1124, 1137, 1143, 1149, 1180, 1181, 1182, 1185, 1195, 1199, 1208, 1215, 1220, 1228, 1257, 1258, 1261, 1268, 1281, 1289, 1295, 1303, 3301
<code>\box_new:N</code>	59, 163
<code>\box_set_wd:Nn</code>	3653, 4005
<code>\box_use:N</code>	3660, 4012
<code>\box_wd:N</code>	440
C	
<code>\c</code>	202, 203, 692, 694, 706, 708
<code>\cB</code>	203
<code>\cE</code>	203
<code>\centering</code>	1348, 1365, 3322, 3432, 3791
<code>check-ans</code>	<u>1789</u>
Document class:	
<code>article</code>	41
clist commands:	
<code>\clist_const:Nn</code>	175
<code>\clist_map_function:nN</code>	3309
<code>\clist_map_inline:Nn</code>	480, 734, 797, 863, 878, 959, 1387
<code>\clist_map_inline:nn</code>	37, 48, 67, 73, 85, 97, 121, 151, 174, 505, 522, 802, 973, 1493, 1737, 1803, 1997, 2015, 2047, 2314, 2486, 2653, 2864, 2867, 2894, 2904, 2907, 2927, 4195
<code>\columnbreak</code>	71
<code>\columnbreak</code>	2256
<code>columns</code>	<u>943</u>
<code>columns-sep</code>	<u>943</u>
<code>\columnsep</code>	88, 91
<code>\columnsep</code>	3049, 3170
<code>\columnseprule</code>	88, 91
<code>\columnseprule</code>	3053, 3172
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, 68, 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	3162, 3314, 3332, 3339, 3382, 3400, 3588, 3691, 3698, 3741, 3759
enumXi	25, 34	\dim_set_eq:NN 530, 577, 648, 652, 2718, 2866, 2906, 3049, 3170, 3389, 3392, 3393, 3579, 3748, 3751, 3752
enumXviii	25, 34	\dim_use:N 806, 814, 1338, 1344, 2174, 2177, 2182, 2788, 2790, 3016, 3021, 3022, 3029, 3039, 3043, 3044, 3046
enumXvii	25, 34, 100	\dim_zero:N 3053, 3172, 3293, 3294, 3295
enumXvi	25, 34	\dim_zero_new:N 3345, 3704
enumXv	25, 34	\c_zero_dim 808, 822, 834, 846, 1338, 1356, 2280, 2825, 2830, 2836, 2843, 3016, 3039, 3142, 3160, 3330, 3398, 3689, 3757
cs commands:		E
\cs_generate_variant:Nn	442, 458, 698, 714, 2099, 2104, 2184, 2854, 3311, 4067	\end .. 1341, 1359, 2133, 2168, 3068, 3085, 3185, 3202, 3419, 3435, 3778, 3794, 4115, 4123, 4129
\cs_if_exist:NTF	412	\endlist .. 32
\cs_new:Nn	188	\endlist .. 336
\cs_new:Npn	206, 1494, 1503, 1512, 2062, 2071, 2079	\endlrbox .. 3651, 4003
\cs_new_eq:NN	335, 336, 337, 341, 342, 387, 388, 391, 392	\endminipage .. 32
\cs_new_protected:Nn	198, 212, 236, 267, 294, 300, 306, 312, 318, 326, 344, 359, 563, 626, 678, 879, 883, 887, 891, 895, 899, 903, 907, 911, 915, 919, 923, 927, 931, 935, 939, 974, 986, 1010, 1027, 1038, 1055, 1130, 1154, 1171, 1233, 1250, 1272, 1307, 1313, 1388, 1402, 1416, 1427, 1438, 1449, 1460, 1471, 1556, 1659, 1672, 1689, 1710, 1738, 1743, 1768, 1808, 1818, 1856, 1871, 1878, 1887, 1892, 1897, 1902, 1911, 1916, 1921, 1926, 2105, 2129, 2136, 2164, 2171, 2212, 2305, 2427, 2475, 2490, 2518, 2551, 2587, 2599, 2607, 2658, 2662, 2681, 2732, 2767, 2783, 2793, 2809, 2947, 2974, 3003, 3010, 3033, 3063, 3075, 3114, 3138, 3156, 3181, 3192, 3229, 3273, 3287, 3307, 3312, 3328, 3396, 3415, 3467, 3489, 3496, 3505, 3515, 3532, 3672, 3687, 3755, 3774, 3824, 3846, 3852, 3865, 3921, 4025	\endminipage .. 342
\cs_new_protected:Npn	180, 184, 395, 410, 427, 437, 443, 543, 588, 660, 685, 699, 1335, 1354, 1524, 1543, 1613, 1646, 1748, 1945, 2016, 2026, 2048, 2056, 2091, 2100, 2227, 2239, 2382, 2394, 2442, 2454, 2528, 2572, 2691, 2709, 2743, 2755, 2823, 2857, 2897, 2957, 3134, 3282, 3347, 3479, 3538, 3545, 3561, 3569, 3574, 3586, 3706, 3839, 3871, 3878, 3894, 3902, 3916, 4046, 4059, 4106, 4192, 4204	enumext .. 5, 2928
\cs_new_protected_nopar:Nn	3525, 3648, 3858, 4000	enumext internal commands:
\cs_new_protected_nopar:Npn	3592, 3949	\l__enumext__check_start_line_env_tl .. 30
\cs_set:Nn	2387	\l__enumext__ref_the_count_tl .. 37
\cs_set:Npn	2315, 2353, 4052	\l__enumext__resume_name_tl .. 57
\cs_set_eq:NN	3457, 3458, 3594, 3814, 3815, 3951	__enumext_add_pre_parsep: .. 46, 984, 986, 986
\cs_set_protected:Nn	803, 819, 831, 843	__enumext_after_args_exec: .. 43, 879, 891, 2940
\cs_set_protected:Npn	33, 42, 60, 68, 80, 86, 113, 147, 155, 459, 481, 510, 523, 570, 715, 735, 779, 798, 855, 864, 943, 960, 1371, 1482, 1729, 1789, 1962, 1998, 2034, 2307, 2479, 2642, 2855, 2895	__enumext_after_args_exec_v: 44, 895, 907, 3107
\cs_to_str:N	429, 452	__enumext_after_args_exec_vii: .. 911, 935
D		__enumext_after_args_exec_viii: .. 939
\d .. 194		__enumext_after_env:nn 65, 89, 102, 184, 184, 3095, 3424, 3686, 3783
\DeclareDocumentEnvironment .. 348		__enumext_after_hyperref: .. 33, 357, 359, 359
dim commands:		__enumext_after_list: .. 89, 98, 105, 2945, 3075, 3075
\dim_abs:n .. 2828, 2833		\l__enumext_after_list_args_v_tl .. 909
\dim_add:Nn .. 3292		\l__enumext_after_list_args_vii_tl 937, 3642
\dim_compare:nNnTF .. 805, 821, 833, 845, 1337, 1356, 2825, 2830, 2836, 2842, 2844, 2846, 3015, 3038, 3142, 3160, 3284, 3330, 3398, 3689, 3757		\l__enumext_after_list_args_viii_tl 941, 3986
\dim_compare:nTF .. 2278		__enumext_after_list_v: .. 91, 3112, 3192, 3192
\dim_gset_eq:NN .. 3407, 3766		__enumext_after_list_vii: .. 3465, 3496, 3496
\dim_gzero:N .. 3441, 3800		__enumext_after_list_viii: .. 3822, 3852, 3852
\dim_new:N .. 56, 63, 64, 65, 82, 128, 164, 165, 171		__enumext_after_star_env:nn .. 96
\dim_set:Nn .. 440, 793, 2723, 2828, 2833, 2835, 2838, 2839, 2843, 2845, 2848, 2849, 2851, 3018, 3041, 3144,		__enumext_after_stop_list: .. 43, 44, 879, 887, 3090
		__enumext_after_stop_list_v: 44, 895, 903, 3207
		\l__enumext_after_stop_list_v_tl .. 905
		__enumext_after_stop_list_vii: 911, 927, 3499
		\l__enumext_after_stop_list_vii_tl .. 929
		__enumext_after_stop_list_viii: .. 931, 3855
		\l__enumext_after_stop_list_viii_tl .. 933
		\l__enumext_align_label_vii_str .. 3634, 3638
		\l__enumext_align_label_viii_str .. 3978, 3982
		\l__enumext_align_label_X_str .. 155
		\c__enumext_all_envs_clist .. 175, 480, 734, 797, 863, 878, 959, 1387
		\c__enumext_all_families_seq .. 111, 4154, 4163, 4185
		__enumext_anskey_env_safe_inner:n 75, 2416, 2422, 2442
		__enumext_anskey_env_safe_outer: .. 75, 2416, 2418, 2427
		\l__enumext_anskey_level_int .. 20, 2233, 2234, 2448, 2449
		__enumext_anskey_safe_inner:n .. 70, 71, 2207, 2212, 2227

__enumext_anskey_safe_outer: . 70, 2202, 2212, 2212
 __enumext_anskey_wrapper:n 1966, 2392
 __enumext_at_begin_document:n . . 32, 180, 180, 333, 339
 __enumext_before_args_exec: 43, 879, 879, 3013
 __enumext_before_args_exec_v: 43, 44, 895, 895, 3141
 __enumext_before_args_exec_vii: . . 911, 911, 3493
 __enumext_before_args_exec_viii: 915, 3849
 __enumext_before_keys_exec: 43, 879, 883, 2938
 __enumext_before_keys_exec_v: . . 44, 895, 899, 3105
 __enumext_before_keys_exec_vii 911
 __enumext_before_keys_exec_vii: 44, 919, 3453
 __enumext_before_keys_exec_viii: . . 44, 923, 3810
 __enumext_before_list: . . . 87, 2932, 3010, 3010
 __enumext_before_list_v: . 90, 3100, 3138, 3138
 __enumext_before_list_vii: 98, 3448, 3489, 3489
 __enumext_before_list_viii: . . 105, 3806, 3846, 3846
 \l__enumext_before_no_starred_key_v_tl 901
 \l__enumext_before_no_starred_key_vii_tl 921
 \l__enumext_before_no_starred_key_viii_tl 925
 \l__enumext_before_starred_key_v_tl . . . 897
 \l__enumext_before_starred_key_vii_tl . 913
 \l__enumext_before_starred_key_viii_tl 917
 __enumext_calc_hspace:NNNNNNN 84, 2823, 2823, 2854, 2859, 2899
 __enumext_check_ans_active: 62, 87, 1808, 1808, 3014, 3492
 \g__enumext_check_ans_item_tl 77
 \g__enumext_check_ans_key_bool 63, 64, 132, 308, 1862, 1868, 1936
 \l__enumext_check_ans_key_bool . . . 63, 80, 132, 1793, 1798, 1859, 1865
 __enumext_check_ans_key_hook: 63, 1856, 1856, 3089, 3500
 __enumext_check_ans_level: . 62, 63, 1808, 1814, 1818
 __enumext_check_ans_log: 64, 65, 1902, 1902, 1940
 __enumext_check_ans_log_msg_greater: 1902, 1908, 1921
 __enumext_check_ans_log_msg_less: 1902, 1906, 1911
 __enumext_check_ans_log_msg_same_ok: 1902, 1907, 1916
 __enumext_check_ans_msg_greater: 1878, 1884, 1897
 __enumext_check_ans_msg_less: 1878, 1882, 1887
 __enumext_check_ans_msg_same_ok: 1878, 1883, 1892
 __enumext_check_ans_show: . . 64, 65, 1878, 1878, 1938
 \g__enumext_check_ans_show_bool 89
 \l__enumext_check_answers_bool 61, 62, 70, 132, 1771, 1797, 1812, 2107, 2131, 2138, 2166, 2204, 2419, 2567, 2695, 2725, 3605
 __enumext_check_starred_cmd:n 30, 65, 77, 1945, 1945, 3110, 3268, 3820
 \g__enumext_check_starred_cmd_int 132, 1948, 1954, 1959, 2765, 3238, 3928
 \l__enumext_check_start_line_env_tl 132, 273, 280, 287, 1951, 1957, 1960
 \l__enumext_columns_sep_v_dim 3160, 3162, 3170
 \l__enumext_columns_sep_vii_dim . . 3330, 3332, 3341, 3386, 3670
 \l__enumext_columns_sep_viii_dim . 3689, 3691, 3700, 3745, 4023
 \l__enumext_columns_v_int 1176, 3158, 3166, 3178, 3183
 \l__enumext_columns_vii_int . . 3335, 3338, 3342, 3350, 3354, 3357, 3363, 3369, 3373, 3665, 3676
 \l__enumext_columns_viii_int . 3694, 3697, 3701, 3709, 3713, 3716, 3722, 3728, 3732, 4018, 4031
 \l__enumext_counter_i_tl 33, 419
 \l__enumext_counter_ii_tl 33, 420
 \l__enumext_counter_iii_tl 33, 421
 \l__enumext_counter_iv_tl 33, 422
 \c__enumext_counter_style_tl 29, 38, 200
 \g__enumext_counter_styles_tl . 25, 35, 56, 430, 448
 \l__enumext_counter_v_tl 33, 423, 668
 \l__enumext_counter_vi_tl 33, 424
 \l__enumext_counter_vii_tl 33, 425, 598
 \l__enumext_counter_viii_tl 33, 426, 615
 \l__enumext_current_widest_dim 25, 56, 454, 531, 578, 649, 653
 __enumext_default_item:n . . . 2691, 2691, 2740
 __enumext_define_counters:Nn 25, 410, 410, 419, 420, 421, 422, 423, 424, 425, 426
 __enumext_endminipage: . 32, 339, 342, 354, 3324, 3650, 4002
 \g__enumext_envir_name_tl 30, 137, 246, 260, 316, 1741, 1746, 1756, 1890, 1895, 1900, 1914, 1919, 1924
 __enumext_execute_after_env: 31, 32, 61, 64, 65, 1926, 1926, 3095, 3686
 __enumext_fake_item: 803, 803, 2886
 \l__enumext_fake_item_indent_v_dim 822, 827
 \l__enumext_fake_item_indent_v_tl 824, 2748, 2752, 2760
 \l__enumext_fake_item_indent_vii_dim 834, 839
 \l__enumext_fake_item_indent_vii_tl 836, 3646
 \l__enumext_fake_item_indent_viii_dim . 846, 851, 3994
 \l__enumext_fake_item_indent_viii_tl . . 848, 3992, 3997
 \l__enumext_fake_item_indent_X_tl 86
 __enumext_fake_item_vii: 803, 831, 2916
 __enumext_fake_item_viii: 803, 843, 2921
 __enumext_filter_save_key:n . . 68, 2023, 2031, 2054, 2060, 2062, 2062, 4071, 4075, 4079, 4083, 4087, 4091
 __enumext_filter_save_key_key:n . . 68, 2062, 2067, 2071
 __enumext_filter_save_key_pair:nn 68, 2062, 2068, 2079
 __enumext_filter_series:n 56, 1494, 1494, 1536, 1548, 1553
 __enumext_filter_series_key:n 56, 1494, 1499, 1503
 __enumext_filter_series_pair:nn . . 56, 1494, 1500, 1512
 \g__enumext_footnote_arg_seq . 152, 2664, 2677,

2687
 \g__enumext_footnote_int . 152, 2671, 2674, 2676, 2678
 \g__enumext_footnote_int_seq . 152, 2665, 2678, 2683, 2686
 __enumext_footnotes_key_bool 33
 \l__enumext_footnotes_key_bool 28, 33, 101, 142, 368, 373, 382, 3613, 3661, 3966, 4013
 __enumext_footnotetext:nn . . . 2658, 2658, 2688
 __enumext_getkeyans:nn . . 109, 4055, 4059, 4059
 __enumext_getkeyans_aux:n 109, 4043, 4046, 4046
 \l__enumext_hyperref_bool 28, 33, 142, 364, 385, 402, 2295, 2555, 3600, 3957
 __enumext_hypertarget:nn 33, 359, 387, 391, 407
 __enumext_if_is_int:n 192
 __enumext_if_is_int:nTF 192, 687, 701
 __enumext_internal_mini_page: . . 32, 344, 344, 2949, 3469
 __enumext_is_not_nested: 30, 86, 212, 212, 2950, 3470
 __enumext_is_on_first_level: . 30, 86, 212, 236, 2955, 3477
 \g__enumext_item_anskey_int 70, 77, 132, 303, 330, 331, 1875, 2206, 2421, 2569
 __enumext_item_answer_diff: 64, 65, 1871, 1871, 1933
 \g__enumext_item_answer_diff_int 64, 141, 304, 1873, 1880, 1904
 \l__enumext_item_column_pos_vii_int 99, 3357, 3363, 3369, 3373, 3380, 3528, 3665, 3668
 \l__enumext_item_column_pos_viii_int . . 105, 3716, 3722, 3728, 3732, 3739, 3861, 4018, 4021
 \l__enumext_item_column_pos_X_int 155
 \g__enumext_item_count_all_vii_int 99, 3381, 3529, 3676, 3683
 \g__enumext_item_count_all_viii_int 105, 3740, 3862, 4030, 4038
 \g__enumext_item_count_all_X_int 155
 \g__enumext_item_number_int . . 63, 132, 302, 329, 331, 1829, 1833, 1836, 1839, 1851, 1875, 2697, 2727, 3607
 __enumext_item_peek_args_vii: 99, 3530, 3532, 3532
 __enumext_item_peek_args_viii: . . 105, 3863, 3865, 3865
 __enumext_item_starred: . . 82, 2783, 2783, 2801
 \l__enumext_item_starred_vii_bool 3547, 3563, 3617
 \l__enumext_item_starred_viii_bool 3880, 3896, 3970, 3990
 \l__enumext_item_starred_X_bool 155
 __enumext_item_std:w 32, 80, 81, 94, 333, 337, 2700, 2706, 2730, 2748, 2752, 2760, 3305
 \g__enumext_item_symbol_aux_vii_tl 3571, 3619, 3622, 3626, 3628
 \g__enumext_item_symbol_aux_X_tl 155
 \l__enumext_item_symbol_sep_vii_dim . . 3580, 3588, 3625, 3627
 \g__enumext_item_symbol_tl 25, 80, 49, 2715, 2789, 2806
 \l__enumext_item_symbol_vii_tl 3622
 \l__enumext_item_text_vii_box 3612, 3653, 3660
 \l__enumext_item_text_viii_box 3965, 4005, 4012
 \l__enumext_item_text_X_box 155
 \l__enumext_item_width_vii_dim . . . 3339, 3384, 3392, 3393
 \l__enumext_item_width_viii_dim . . 3698, 3743, 3751, 3752
 \l__enumext_item_width_X_dim 155
 \l__enumext_itemindent_X_dim 60
 \l__enumext_itemsep_vii_skip 3682
 \l__enumext_itemsep_viii_skip 4037
 \l__enumext_joined_item_aux_vii_int . . 3378, 3379, 3380, 3381, 3387
 \l__enumext_joined_item_aux_viii_int . 3737, 3738, 3739, 3740, 3746
 \l__enumext_joined_item_aux_X_int 155
 __enumext_joined_item_vii:w . . 99, 3535, 3536, 3538, 3538
 \l__enumext_joined_item_vii_int . . 3349, 3350, 3353, 3355, 3361, 3366, 3371, 3376, 3378, 3384
 __enumext_joined_item_viii:w . 105, 106, 3868, 3869, 3871, 3871
 \l__enumext_joined_item_viii_int . 3708, 3709, 3712, 3714, 3720, 3725, 3730, 3735, 3737, 3743
 \l__enumext_joined_item_X_int 155
 \l__enumext_joined_width_vii_dim . 3382, 3389, 3392, 3643, 3655
 \l__enumext_joined_width_viii_dim 3741, 3748, 3751, 3987, 4007
 \l__enumext_joined_width_X_dim 155
 __enumext_keyans_addto_prop:n 75, 2454, 2454, 2762, 3235
 __enumext_keyans_addto_seq:n . 77, 2528, 2528, 2764, 3237
 __enumext_keyans_addto_seq_link: 2528, 2549, 2551, 3927
 __enumext_keyans_anspic_code:nnn 92, 93, 3226, 3229, 3229
 __enumext_keyans_default_item:n . . 81, 2743, 2743, 2779
 \l__enumext_keyans_env_bool 26, 2979, 2992, 3121, 3206
 __enumext_keyans_fake_item: . . 803, 819, 2876
 \l__enumext_keyans_item_opt_tl . 106, 98, 2576, 2589, 2595, 3912
 \l__enumext_keyans_level_h_int . . 20, 608, 635, 2506, 3826, 3827
 \l__enumext_keyans_level_int . . 20, 1329, 2218, 2433, 2501, 3120, 3125, 3220
 __enumext_keyans_make_label: 35, 83, 2809, 2809, 2874
 __enumext_keyans_mini_addvspace: 50, 90, 1233, 1233, 3150
 __enumext_keyans_mini_right_cmd:n 53, 1331, 1354, 1354
 __enumext_keyans_mini_set_vskip: . 50, 1171, 1171, 1235
 __enumext_keyans_multi_addvspace: 91, 1027, 1038, 3175
 __enumext_keyans_multi_set_vskip: 47, 1027, 1027, 1040
 __enumext_keyans_multicols_start: . . 90, 91, 3154, 3156, 3156
 __enumext_keyans_multicols_stop: . 91, 1358, 3181, 3181, 3205
 __enumext_keyans_parse_keys:n 3099, 3134, 3134
 \l__enumext_keyans_pic_above_int . 127, 3315,

3316, 3318
 \l__enumext_keyans_pic_above_skip .. 94, 127,
 3259, 3299
 __enumext_keyans_pic_arg_two: 94, 3257, 3287,
 3287
 \l__enumext_keyans_pic_below_int . 127, 3315,
 3316, 3319
 \l__enumext_keyans_pic_body_seq .. 92-94, 127,
 3224, 3264, 3323
 __enumext_keyans_pic_do:n 94, 3264, 3266, 3307,
 3307, 3311
 \l__enumext_keyans_pic_level_int .. 20, 1321,
 2222, 2437, 2457, 2496, 2531, 2609, 3275, 3276
 __enumext_keyans_pic_row:n 94, 3309, 3312, 3312
 __enumext_keyans_pic_safe_exec: .. 93, 3253,
 3273, 3273
 __enumext_keyans_pic_skip_abs:N .. 94, 3282,
 3282, 3298
 \l__enumext_keyans_pic_width_dim . 127, 3314,
 3321
 __enumext_keyans_redefine_item: .. 82, 2767,
 2767, 2873
 __enumext_keyans_ref: 39, 660, 678, 2875
 __enumext_keyans_ref:n 39, 657, 660, 660
 __enumext_keyans_safe_exec: . 3098, 3114, 3114
 __enumext_keyans_save_start_line: . 30, 267,
 267, 3122, 3280, 3831
 __enumext_keyans_show_ans: .. 2572, 2580, 2599
 __enumext_keyans_show_item_opt: . 2572, 2587,
 2760, 3249, 3993
 __enumext_keyans_show_left:n . 81, 2572, 2572,
 2758, 3244
 __enumext_keyans_show_pos: .. 2572, 2584, 2607
 __enumext_keyans_starred_item:n .. 81, 2755,
 2755, 2775
 __enumext_keyans_store_ref: .. 76, 2475, 2475,
 2763, 3236, 3925
 __enumext_keyans_store_ref_aux_i: 76, 2475,
 2487, 2490
 __enumext_keyans_store_ref_aux_ii: 77, 2475,
 2516, 2518
 \l__enumext_keyans_tmpa_tl .. 26, 98, 2757, 2761
 __enumext_keyans_wrapper_opt:n .. 1969, 2595
 \l__enumext_label_copy_i_tl .. 2349, 2494, 2499,
 2504, 2509
 \l__enumext_label_copy_v_tl 2504
 \l__enumext_label_copy_vi_tl 2499
 \l__enumext_label_copy_vii_tl 2324, 2335, 2366,
 2494
 \l__enumext_label_copy_viii_tl 2509
 \l__enumext_label_copy_X_tl 144
 \l__enumext_label_fill_left_v_tl 2813
 \l__enumext_label_fill_left_X_tl 86
 \l__enumext_label_fill_right_v_tl 2820
 \l__enumext_label_fill_right_X_tl 86
 \l__enumext_label_font_style_v_tl 2814, 3248
 \l__enumext_label_font_style_vii_tl ... 3631
 \l__enumext_label_font_style_viii_tl .. 3975
 \l__enumext_label_i_tl 523
 \l__enumext_label_ii_tl 523
 \l__enumext_label_iii_tl 523
 \l__enumext_label_iv_tl 523
 __enumext_label_style:Nnn 25, 35, 443, 443, 458,
 528, 575, 646, 650
 \l__enumext_label_v_tl .. 75, 77, 643, 2462, 2536,
 2601, 2636, 2757, 2761, 3102, 3243, 3245
 \l__enumext_label_vi_tl . 75, 77, 643, 2459, 2533,
 3243, 3245, 3249
 \l__enumext_label_vii_tl . 570, 3558, 3583, 3590
 \l__enumext_label_viii_tl 570, 3891, 3919, 3923
 \l__enumext_label_width_by_box .. 56, 439, 440
 __enumext_label_width_by_box:Nn 35, 437, 437,
 442, 454, 711
 \l__enumext_labelsep_i_dim ... 2604, 2639, 3931,
 3946
 \l__enumext_labelsep_v_dim 3165
 \l__enumext_labelsep_vii_dim . 3334, 3343, 3385,
 3581, 3641, 3657
 \l__enumext_labelsep_viii_dim 3693, 3702, 3744,
 3985, 3994, 4009
 \l__enumext_labelwidth_i_dim . 2604, 2639, 3931,
 3946
 \l__enumext_labelwidth_v_dim 3165
 \l__enumext_labelwidth_vii_dim ... 3334, 3342,
 3385, 3634, 3638, 3656
 \l__enumext_labelwidth_viii_dim .. 3693, 3701,
 3744, 3978, 3982, 4008
 \l__enumext_leftmargin_tmp_v_bool . 94, 3289
 \l__enumext_leftmargin_tmp_X_bool 60
 \l__enumext_leftmargin_tmp_X_dim 60
 \l__enumext_leftmargin_X_dim 60
 __enumext_level: 188, 188, 552, 555, 556, 565, 567,
 806, 810, 814, 881, 885, 889, 893, 976, 978, 980, 982,
 1015, 1017, 1019, 1021, 1025, 1058, 1061, 1080, 1089,
 1095, 1100, 1104, 1115, 1119, 1120, 1125, 1161, 1165,
 1338, 1344, 1391, 1393, 1395, 1398, 1405, 1407, 1409,
 1412, 2018, 2020, 2022, 2050, 2051, 2053, 2109, 2117,
 2121, 2125, 2387, 2390, 2391, 2699, 2700, 2704, 2705,
 2706, 2713, 2715, 2719, 2720, 2723, 2729, 2730, 2785,
 2788, 2790, 2797, 2798, 2799, 2802, 2805, 2935, 2937,
 2985, 2998, 3005, 3016, 3018, 3021, 3022, 3024, 3029,
 3036, 3039, 3041, 3043, 3044, 3045, 3046, 3049, 3055,
 3060, 3066, 3069, 3071, 3077
 \l__enumext_level_h_int .. 20, 220, 242, 255, 591,
 628, 1826, 1842, 2343, 2360, 3471, 3472
 \l__enumext_level_int . 86, 20, 190, 229, 241, 256,
 346, 988, 1132, 1325, 1820, 1848, 1928, 2320, 2330,
 2336, 2342, 2350, 2358, 2365, 2889, 2951, 2952, 2962,
 2969, 2983, 2996, 3051, 3129, 3216, 3509, 3519, 3834
 __enumext_list_arg_two_i: 2855
 __enumext_list_arg_two_ii: 2855
 __enumext_list_arg_two_iii: 2855
 __enumext_list_arg_two_iv: 2855
 __enumext_list_arg_two_v: . 82, 2855, 3104, 3290
 __enumext_list_arg_two_vii: 2895, 3452
 __enumext_list_arg_two_viii: 2895, 3809
 \l__enumext_listoffset_v_dim 3167
 \l__enumext_listparindent_vii_dim 3644
 \l__enumext_listparindent_viii_dim ... 3988
 __enumext_log_answer_vars: . 32, 318, 326, 1935
 __enumext_log_global_vars: . 31, 318, 318, 1934
 __enumext_make_label: 35, 80, 82, 2793, 2793, 2884
 \l__enumext_mark_answer_sym_tl . 70, 122, 1975,
 2179, 2402, 2611, 2624, 3935
 \l__enumext_mark_position_str 122, 1979, 1980,
 2003, 2004, 2177
 \l__enumext_mark_ref_sym_tl .. 122, 1989, 2300,
 2563

__enumext_mini_addvspace: . . 49, 87, 1154, 1154, 3026
 __enumext_mini_addvspace_vii: 52, 1307, 1307, 3410
 __enumext_mini_addvspace_viii: 52, 1307, 1313, 3769
 __enumext_mini_env* 344
 __enumext_mini_right_cmd:n . 52, 53, 1333, 1335, 1335
 __enumext_mini_set_vskip: . 47, 1055, 1055, 1156
 __enumext_mini_set_vskip_vii: 51, 1250, 1250, 1309
 __enumext_mini_set_vskip_viii: 51, 1250, 1272, 1315
 __enumext_minipage:w 32, 339, 341, 350, 3321, 3643, 3987
 \l__enumext_minipage_active_v_bool . . 90, 91, 3148, 3173, 3186, 3194
 \g__enumext_minipage_active_vii_bool . . . 96, 3421, 3426, 3438
 \l__enumext_minipage_active_vii_bool . 3406, 3417
 \g__enumext_minipage_active_viii_bool 3780, 3785, 3797
 \l__enumext_minipage_active_viii_bool 3765, 3776
 \g__enumext_minipage_active_X_bool . . . 155
 \l__enumext_minipage_active_X_bool 74
 \g__enumext_minipage_after_skip 74, 1254, 1266, 3436, 3795
 \l__enumext_minipage_after_skip 48, 49, 89, 91, 74, 1071, 1086, 1106, 1122, 1137, 1143, 1149, 1163, 1173, 1182, 1185, 1197, 1215, 1226, 1242, 1274, 1287, 1301, 3086, 3203
 \g__enumext_minipage_center_vii_bool . 3430, 3439
 \g__enumext_minipage_center_viii_bool 3789, 3798
 \g__enumext_minipage_center_X_bool . . . 155
 \l__enumext_minipage_hsep_v_dim . . . 90, 3146
 \l__enumext_minipage_hsep_vii_dim 3404
 \l__enumext_minipage_hsep_viii_dim . . . 3763
 \l__enumext_minipage_left_skip 48, 90, 74, 1063, 1078, 1097, 1112, 1159, 1169, 1174, 1180, 1189, 1206, 1218, 1238, 1248, 1252, 1257, 1261, 1275, 1279, 1293, 1311, 1317
 \l__enumext_minipage_left_v_dim 90, 3144, 3152
 \l__enumext_minipage_left_vii_dim 3400, 3412
 \l__enumext_minipage_left_viii_dim 3759, 3771
 \l__enumext_minipage_left_X_dim 74
 \g__enumext_minipage_right_skip 74, 1253, 1258, 1262, 3429, 3788
 \l__enumext_minipage_right_skip . 48, 74, 1067, 1082, 1102, 1117, 1175, 1181, 1193, 1211, 1222, 1276, 1283, 1297, 1345, 1362
 \l__enumext_minipage_right_v_dim . . 90, 1356, 1361, 3142, 3146
 \g__enumext_minipage_right_vii_dim 96, 3408, 3428, 3441
 \l__enumext_minipage_right_vii_dim 96, 3398, 3403, 3409
 \g__enumext_minipage_right_viii_dim . . 3767, 3787, 3800
 \l__enumext_minipage_right_viii_dim . . 3757, 3762, 3768
 \g__enumext_minipage_right_X_dim 155
 \g__enumext_minipage_right_X_skip 155
 \g__enumext_minipage_stat_int . 87, 90, 74, 1350, 1367, 3025, 3079, 3084, 3149, 3196, 3201
 \g__enumext_miniright_code_vii_tl . 97, 3434, 3440
 \g__enumext_miniright_code_viii_tl 3793, 3799
 \g__enumext_miniright_code_X_tl 155
 __enumext_multi_addvspace: . 46, 88, 1010, 1010, 3057
 __enumext_multi_set_vskip: . 45, 974, 974, 1012
 \l__enumext_multicols_above_ii_skip . . . 993
 \l__enumext_multicols_above_iii_skip . . . 999
 \l__enumext_multicols_above_iv_skip . . . 1005
 \l__enumext_multicols_above_v_skip 1029, 1043, 1053
 \l__enumext_multicols_above_X_skip 68
 \l__enumext_multicols_below_v_skip 1033, 1047, 3188
 \l__enumext_multicols_below_X_skip 68
 __enumext_multicols_start: . 87, 88, 3031, 3033, 3033
 __enumext_multicols_stop: 88, 1340, 3063, 3063, 3088
 __enumext_newlabel:nn 28, 33, 74, 395, 395, 2376, 2522
 \l__enumext_newlabel_arg_one_tl 28, 33, 74, 76, 144, 2299, 2369, 2377, 2511, 2523, 2561
 \l__enumext_newlabel_arg_two_tl 28, 33, 73, 144, 2323, 2333, 2347, 2363, 2378, 2498, 2503, 2508, 2524
 __enumext_parse_keys:n . . . 56, 2931, 2957, 2957
 __enumext_parse_keys_vii:n 56, 3447, 3479, 3479
 __enumext_parse_keys_viii:n . 3805, 3839, 3839
 __enumext_parse_save_key:n 67, 2043, 2048, 2048
 __enumext_parse_save_key_vii:n 67, 2038, 2048, 2056
 __enumext_parse_serie:n 98
 __enumext_parse_series:n . . 56, 86, 1524, 1524, 2965, 3485
 __enumext_parse_store_keys:n 86
 \l__enumext_parsep_i_skip 991, 993, 1135, 1183
 \l__enumext_parsep_ii_skip 997, 999, 1141
 \l__enumext_parsep_iii_skip . . 1003, 1005, 1147
 \l__enumext_parsep_vii_skip 3645
 \l__enumext_parsep_viii_skip 3989
 \l__enumext_partopsep_v_skip . 1045, 1049, 1209, 1213, 1220, 1224, 1240, 1244
 \l__enumext_partopsep_viii_skip 1285
 __enumext_phantomsection: 33, 359, 388, 392, 408
 __enumext_print_footnote: . . . 2658, 2681, 3663, 4015
 __enumext_print_keyans_box:NN 70, 2171, 2171, 2184, 2389, 2603, 2638, 3931, 3946
 \l__enumext_print_keyans_i_tl 4076, 4098
 \l__enumext_print_keyans_ii_tl . . . 4080, 4099
 \l__enumext_print_keyans_iii_tl . . 4084, 4100
 \l__enumext_print_keyans_iv_tl . . . 4088, 4101
 \l__enumext_print_keyans_starred_tl 109, 110, 112, 4072, 4120
 \l__enumext_print_keyans_vii_tl 109, 4092, 4102
 \l__enumext_print_keyans_X_tl 112
 __enumext_printkeyans:nnn 110, 4103, 4106, 4106
 __enumext_redefine_item: . 81, 2732, 2732, 2883
 \l__enumext_ref_key_arg_tl 37, 38, 203, 545, 546,

559, 590, 593, 604, 610, 621, 662, 663, 674
 \l__enumext_ref_the_count_tl . 37, 38, 552, 555,
 558, 598, 600, 603, 615, 617, 620, 668, 670, 673
 __enumext_regex_counter_style: . . 29, 37, 198,
 198, 553, 599, 616, 669
 __enumext_register_counter_style:Nn . . 427,
 427, 432, 433, 434, 435, 436
 __enumext_remove_extra_parsep_vii: . . 3462,
3672, 3672
 __enumext_remove_extra_parsep_viii: . 3819,
4025, 4025
 __enumext_renew_footnote: . . . 2658, 2662, 3615,
 3968
 \l__enumext_renew_the_count_v_tl 671, 680, 682
 \l__enumext_renew_the_count_vii_tl 601, 630,
 632
 \l__enumext_renew_the_count_viii_tl 618, 637,
 639
 \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,
 1942
 \l__enumext_resume_active_bool 56, 59, 49, 1528,
 1648
 __enumext_resume_counter: . . 58, 59, 1646, 1652,
 1659
 __enumext_resume_counter:n . 56, 59, 1617, 1622,
1646, 1646, 1716, 1724
 __enumext_resume_counter_save_ans: . . 59, 60,
1646, 1657, 1689
 __enumext_resume_counter_series: . 59, 1646,
 1655, 1672
 \g__enumext_resume_int . . . 49, 1569, 1663, 1664
 __enumext_resume_last:n 56, 57, 1524, 1530, 1543
 \l__enumext_resume_name_tl 49, 1565, 1573, 1576,
 1592, 1600, 1603, 1649, 1650, 1678, 1685
 __enumext_resume_save_counter: 57, 1556, 1556,
 3093, 3503
 __enumext_resume_series:n . 58, 1488, 1613, 1613
 __enumext_resume_starred: . 60, 1489, 1710, 1710
 \g__enumext_resume_vii_int . . 98, 49, 1596, 1668,
 1669
 __enumext_safe_exec: . . 32, 86, 2930, 2947, 2947
 __enumext_safe_exec_vii: . 32, 3446, 3467, 3467
 __enumext_safe_exec_viii: . . . 3804, 3824, 3824
 \l__enumext_series_name_tl 59
 \l__enumext_series_str . 57, 86, 1486, 1526, 1534,
 1535, 1537, 1539, 1560, 1563, 1567, 1587, 1590, 1594,
 2961, 3483
 __enumext_set_error:nn 4192, 4202, 4204
 __enumext_set_parse:n 4175, 4192, 4192
 \l__enumext_setkey_tmpa_int . . . 107, 4168, 4172
 \l__enumext_setkey_tmpa_seq . . . 107, 4166, 4176,
 4182, 4184, 4186, 4199
 \l__enumext_setkey_tmpa_tl 107, 4174, 4178
 \l__enumext_setkey_tmpb_seq . . . 107, 4167, 4170,
 4174, 4175
 \l__enumext_setkey_tmpb_tl 107, 4194, 4196, 4197
 \l__enumext_show_answer_bool . 122, 1983, 2007,
 2396, 2578, 2592, 3240, 3929
 __enumext_show_length:nnn . . 42, 206, 206, 4268,
 4269, 4270, 4271, 4272, 4273, 4274, 4275, 4276, 4277,
 4283, 4284, 4285, 4286, 4287, 4288, 4289, 4290, 4291,
 4292
 \l__enumext_show_position_bool 122, 1986, 2010,
 2400, 2582, 2593, 3241, 3933
 \g__enumext_standar_bool 30, 86, 26, 219, 222, 240,
 309, 1558, 1623, 1635, 1661, 1674, 1712, 1847, 1860
 \l__enumext_standar_bool . 86, 89, 26, 2328, 2341,
 2357, 2954, 3092
 \l__enumext_standar_first_bool 30, 86, 26, 245,
 1545, 1692, 1754, 1761
 __enumext_standar_item_vii:w . 99, 3543, 3545,
 3545
 __enumext_standar_item_viii:w 106, 3876, 3878,
 3878
 __enumext_standar_ref: 37, 543, 563, 2885
 __enumext_standar_ref:n 37, 535, 543, 543
 \g__enumext_standar_series_tl . 49, 1547, 1548,
 1714, 1717
 \g__enumext_starred_bool 30, 97, 98, 26, 228, 231,
 254, 310, 1585, 1628, 1639, 1666, 1681, 1720, 1825,
 1866, 2319, 2329, 2359, 2492, 2980, 2993, 3442
 \l__enumext_starred_bool . 97, 98, 26, 2254, 2260,
 2344, 2385, 3476, 3502
 __enumext_starred_columns_set_vii: . . 3328,
 3328, 3455
 __enumext_starred_columns_set_viii: . 3687,
 3687, 3812
 \l__enumext_starred_first_bool . . 30, 26, 259,
 1550, 1701, 1754, 1761
 __enumext_starred_item:nn . . . 2709, 2709, 2738
 __enumext_starred_item_exec: . 107, 3921, 3921,
 3972
 __enumext_starred_item_vii:w . . 99, 100, 3542,
3561, 3561
 __enumext_starred_item_vii_aux_i:w . . 3561,
 3566, 3569
 __enumext_starred_item_vii_aux_ii:w . 3561,
 3567, 3572, 3574
 __enumext_starred_item_vii_aux_iii:w 3561,
 3577, 3586
 __enumext_starred_item_viii:w 106, 3875, 3894,
 3894
 __enumext_starred_item_viii_aux_i:w . . 106,
3894, 3899, 3902
 __enumext_starred_item_viii_aux_ii:w . 106,
3894, 3900, 3914, 3916
 __enumext_starred_joined_item_vii:n . 95, 99,
3347, 3347, 3540
 __enumext_starred_joined_item_viii:n . 102,
106, 3706, 3706, 3873
 __enumext_starred_ref: 38, 588, 626, 2913
 __enumext_starred_ref:n 38, 582, 588, 588
 \g__enumext_starred_series_tl . 49, 1552, 1553,
 1722, 1725
 __enumext_start_from:NNn 39, 685, 685, 698, 720
 \l__enumext_start_i_int 1664, 1676, 1695
 __enumext_start_item_tmp_vii: 97, 3458, 3525,
 3525
 __enumext_start_item_tmp_viii: . . 104, 3815,
3858, 3858
 __enumext_start_item_vii:w 99, 100, 3553, 3558,
 3583, 3590, 3592, 3592
 __enumext_start_item_viii:w . . 106, 3886, 3891,
 3919, 3949, 3949

`\g__enumext_start_line_tl` 30, [132](#), 247, 261, 315, 1890, 1895, 1900, 1914, 1919, 1924
`__enumext_start_list:nn` 32, 83, 94, [333](#), 335, 2934, 3101, 3254, 3450, 3807
`__enumext_start_mini_vii:` . 98, [3396](#), 3396, 3494
`__enumext_start_mini_viii:` . . . [105](#), [3755](#), 3755, 3850
`__enumext_start_save_ans_msg:` 61, [1738](#), 1738, 1763
`__enumext_start_store_level:` . 86, 2933, [2974](#), 2974
`__enumext_start_store_level_vii:` . 98, 3449, [3505](#), 3505
`\l__enumext_start_vii_int` . . . 1669, 1683, 1704
`\l__enumext_start_X_int` [86](#), [715](#)
`__enumext_stop_item_tmp_vii:` 97, 99, 100, 3457, 3461, 3527, 3594
`__enumext_stop_item_tmp_viii:` 104, 105, 3814, 3818, 3860, 3951
`__enumext_stop_item_vii:` 100, 101, 3594, [3648](#), 3648
`__enumext_stop_item_viii:` 108, 3951, 4000, 4000
`__enumext_stop_list:` . . 32, [333](#), 336, 2943, 3111, 3267, 3463, 3821
`__enumext_stop_mini_vii:` 96, 98, [3415](#), 3415, 3498
`__enumext_stop_mini_viii:` 105, [3755](#), 3774, 3854
`__enumext_stop_save_ans_msg:` . 61, [1738](#), 1743, 1932
`__enumext_stop_store_level:` . . 86, 2944, [2974](#), 3003
`__enumext_stop_store_level_vii:` . . 98, 3464, [3505](#), 3515
`\l__enumext_store_active_bool` 26, 61, 86, 98, 98, 1693, 1702, 1770, 2214, 2429, 2978, 2991, 3116, 3124, 3212, 3271, 3507, 3517, 3833
`__enumext_store_active_keys:n` 67, [2016](#), 2016, 2971
`__enumext_store_active_keys_vii:n` . . 67, 98, [2016](#), 2026, 3486
`__enumext_store_addto_prop:n` 68, 75, [2091](#), 2091, 2099, 2241, 2473, 3924
`__enumext_store_addto_seq:n` 68, 77, [2100](#), 2100, 2104, 2111, 2125, 2133, 2142, 2160, 2168, 2303, 2566
`\l__enumext_store_anskey_arg_tl` 26, 71, 72, 98, 2251, 2256, 2258, 2263, 2270, 2273, 2283, 2288, 2291, 2297, 2303
`__enumext_store_anskey_code:nn` . 70, 71, 2208, [2239](#), 2239, 2423
`__enumext_store_anskey_safe_outer:` 71
`__enumext_store_anskey_show_left:n` 74, 2246, [2394](#), 2394
`__enumext_store_anskey_show_wrap:n` 74, [2382](#), 2382, 2398, 2413
`\l__enumext_store_columns_break_bool` . 2187, 2253
`\l__enumext_store_columns_join_int` [98](#)
`__enumext_store_internal_ref:` . . 71, 72, 2244, [2305](#), 2305
`\l__enumext_store_item_join_int` . . 2190, 2261, 2265
`\l__enumext_store_item_star_bool` . 2192, 2268
`\l__enumext_store_item_symbol_sep_dim` 2197, 2280, 2285
`\l__enumext_store_item_symbol_tl` . 2195, 2271, 2275
`\l__enumext_store_keyans_item_opt_sep_tl` 1972, 2467, 2469, 2540, 2544, 3907, 3909
`\l__enumext_store_keyans_item_opt_tl` . . . [98](#)
`\l__enumext_store_keyans_label_tl` 26, 75, 77, 106, [98](#), 2456, 2459, 2462, 2469, 2471, 2473, 2530, 2533, 2536, 2542, 2547, 2557, 2566, 3904, 3909, 3910, 3923, 3924, 3926
`__enumext_store_level_close:` . 69, [2105](#), 2129, 3007
`__enumext_store_level_close_vii:` [2136](#), 2164, 3521
`__enumext_store_level_open:` . . 69, [2105](#), 2105, 2986, 2999
`__enumext_store_level_open_vii:` . [2136](#), 2136, 3511
`\g__enumext_store_name_tl` . 26, 89, [98](#), 314, 321, 322, 323, 324, 1746, 1772, 1889, 1894, 1899, 1913, 1918, 1923, 1930
`\l__enumext_store_name_tl` . 26, 61, 62, [98](#), 1579, 1582, 1606, 1609, 1697, 1706, 1741, 1750, 1751, 1772, 1773, 1775, 1776, 1778, 1780, 1781, 1783, 1785, 1786, 1810, 2093, 2095, 2102, 2371, 2372, 2408, 2513, 2514, 2617, 2630, 3941
`\l__enumext_store_ref_key_bool` 71, 1992, 2242, 2294, 2477, 2554
`\l__enumext_store_save_key_vii_bool` . . 2028, 2058
`\l__enumext_store_save_key_vii_tl` 2030, 2031, 2059, 2060, 2140, 2150, 2156, 2160
`\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](#), 2374, 2380, 2520, 2526
`__enumext_storing_exec:` . . 61, [1748](#), 1764, 1768
`__enumext_storing_set:n` . . 61, 1733, [1748](#), 1748
`\l__enumext_the_counter_v_tl` 670
`\l__enumext_the_counter_vii_tl` 600
`\l__enumext_the_counter_viii_tl` 617
`\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, 798, 802, 1482, 1493, 1729, 1737, 1789, 1807, 1962, 1997, 1998, 2015, 2034, 2047, 2307, 2314, 2315, 2336, 2350, 2353, 2365, 2479, 2486, 2855, 2894, 2895, 2927
`__enumext_tmp:nn` 459, 480, 481, 509, 510, 522, 715, 734, 779, 797, 855, 863, 864, 878, 943, 959, 960, 973, 1371, 1387, 2642, 2657
`__enumext_tmp:nnn` 523, 539, 540, 541, 542, 570, 586, 587
`__enumext_tmp:nnnnn` 735, 760, 763, 766, 768, 770, 773, 776
`__enumext_tmp:w` 4052, 4055
`\l__enumext_tmpa_vii_int` 3338, 3341
`\l__enumext_tmpa_viii_int` 3697, 3700
`\l__enumext_tmpa_X_int` [155](#)
`\l__enumext_topsep_v_skip` 1031, 1035, 1178, 1191, 1199, 1204, 1224, 1228, 3270, 3302
`\l__enumext_topsep_vii_skip` . . 1255, 1264, 1268
`\l__enumext_topsep_viii_skip` . 1277, 1299, 1303
`\l__enumext_vspace_a_star_v_bool` 1420
`\l__enumext_vspace_a_star_vii_bool` . . . 1442
`\l__enumext_vspace_a_star_viii_bool` . . . 1453
`\l__enumext_vspace_a_star_X_bool` [86](#)

`__enumext_vspace_above:` .. 54, [1388](#), 1388, 3012
`__enumext_vspace_above_v:` . 54, [1416](#), 1416, 3140
`\l_enumext_vspace_above_v_skip` .. 1418, 1422, 1424
`__enumext_vspace_above_vii:` .. 55, [1438](#), 1438, 3491
`\l_enumext_vspace_above_vii_skip` 1440, 1444, 1446
`__enumext_vspace_above_viii:` . 55, [1438](#), 1449, 3848
`\l_enumext_vspace_above_viii_skip` 1451, 1455, 1457
`\l_enumext_vspace_b_star_v_bool` 1431
`\l_enumext_vspace_b_star_vii_bool` ... 1464
`\l_enumext_vspace_b_star_viii_bool` ... 1475
`\l_enumext_vspace_b_star_X_bool` [86](#)
`__enumext_vspace_below:` .. 54, [1402](#), 1402, 3091
`__enumext_vspace_below_v:` . 54, [1427](#), 1427, 3208
`\l_enumext_vspace_below_v_skip` .. 1429, 1433, 1435
`__enumext_vspace_below_vii:` .. 55, [1460](#), 1460, 3501
`\l_enumext_vspace_below_vii_skip` 1462, 1466, 1468
`__enumext_vspace_below_viii:` . 55, [1460](#), 1471, 3856
`\l_enumext_vspace_below_viii_skip` 1473, 1477, 1479
`__enumext_widest_from:nNNn` .. 40, [699](#), 699, 714, 726
`\g_enumext_widest_label_tl` 25, 35, [56](#), 447, 451, 455
`\l_enumext_wrap_label_opt_v_bool` 2751
`\l_enumext_wrap_label_opt_vii_bool` 99, 3552
`\l_enumext_wrap_label_opt_viii_bool` .. 106, 3885
`\l_enumext_wrap_label_opt_X_bool` [86](#)
`\l_enumext_wrap_label_v_bool` 2747, 2751, 2759, 2815
`\l_enumext_wrap_label_vii_bool` 99, 3551, 3556, 3564, 3632
`\l_enumext_wrap_label_viii_bool` . 106, 3884, 3889, 3897, 3976
`\l_enumext_wrap_label_X_bool`..... [86](#)
`__enumext_wrapper_label_v:n` 2817, 3249
`__enumext_wrapper_label_vii:n` 3635
`__enumext_wrapper_label_viii:n` 3979
`__enumext_zero_parsep:` ... 49, [1075](#), [1130](#), 1130
`enumext*` 5, [3444](#)
`enumXi` [419](#)
`enumXii` [419](#)
`enumXiii` [419](#)
`enumXiv` [419](#)
`enumXv` [419](#)
`enumXvi` [419](#)
`enumXvii` [419](#)
`enumXviii` [419](#)
Environments provide by **enumext**:
 `enumext*` .. 24, 25, 27–30, 32, 34, 37, 38, 40–45, 51, 52, 55–58, 60–63, 65–73, 76, 79, 85, 86, 97, 98, 100, 102, 103, 105, 107, 109, 110, 113, 115
 `enumext` 24, 25, 27, 29, 30, 32, 34–43, 45–58, 60–63, 65–71, 73, 76, 79–86, 89, 93, 95, 96, 98, 109, 111, 113, 114
 `keyans*` 24–30, 34, 37–45, 51, 52, 55, 61, 62, 65, 66, 68, 76,

79, 85, 104, 105, 113, 115
`keyanspic` 24–27, 30, 34, 35, 38, 52, 61, 62, 65, 68, 75–78, 92–94, 115
`keyans` 24–27, 29, 30, 34, 35, 38, 40–45, 47, 50, 52–54, 61, 62, 65, 66, 68, 75–78, 82–84, 89, 90, 92–94, 96, 105, 113, 115
Environments:
 `list` 29, 32, 83, 85
 `lrbox` 95, 101, 107, 108
 `minipage` 29, 32, 45, 47, 92–95, 101, 108
 `multicols` 45–48, 52, 87–91
exp commands:
 `\exp_after:wN` 4055
 `\exp_args:Ne` 2968, 4043
 `\exp_not:N` . 46, 450, 558, 603, 620, 673, 812, 826, 827, 838, 839, 850, 851, 2299, 2405, 2406, 2559, 2614, 2615, 2627, 2628, 3938, 3939, 4052
 `\exp_not:n` 249, 263, 275, 282, 289, 558, 559, 603, 604, 620, 621, 673, 674, 813, 1510, 1522, 2077, 2089, 2265, 2275, 2285, 2299, 2300, 2377, 2523, 2561, 2563

F
`\fbox` 1967
file commands:
 `\file_input_stop:` 4437
`first` [864](#)
`font` [459](#)
`\footnote` 79
`\footnote` 79, 2666
`\footnotemark` 2676
`\footnotesize` 2406, 2615, 2628, 3939
`\footnotetext` 2660

G
`\getkeyans` 15, 109, [4041](#)
group commands:
 `\group_begin:` .. 2203, 2404, 2613, 2626, 3611, 3630, 3937, 3964, 3974, 4063, 4097
 `\group_end:` 2210, 2411, 2620, 2633, 3640, 3652, 3944, 3984, 4004, 4065, 4104

H
`\hbadness` 3659, 4011
hbox commands:
 `\hbox_set:Nn` 439
`\hfill` 489, 493, 498, 499, 1342, 1360, 2299, 2559, 3420, 3779
hook commands:
 `\hook_gput_code:nnn` 9, 182, 186, 357
 `\hook_gset_rule:nnnn` 358
`\hspace` 3670, 4023
`\hyperlink` 72, 77
`\hyperlink` 2299, 2559
`\hypertarget` 33
`\hypertarget` 387

I
`\IfHyperBoolean` 365
`\IfPackageLoadedTF` 11, 361, 375
`\ignorespaces` 815
`\inputlineno` 249, 263, 275, 282, 289
int commands:
 `\int_add:Nn` 3380, 3739
 `\int_case:nn` 988, 1132, 1820, 1842, 1880, 1904
 `\int_compare:nNnTF` .. 346, 591, 608, 628, 635, 1057, 1176, 1321, 1325, 1329, 1928, 1947, 1953, 2218, 2222, 2234, 2433, 2437, 2449, 2457, 2496, 2501, 2506, 2531,

2609, 2952, 2962, 2983, 2996, 3035, 3051, 3065, 3079, 3125, 3129, 3158, 3183, 3196, 3216, 3220, 3276, 3350, 3360, 3376, 3472, 3509, 3519, 3665, 3674, 3709, 3719, 3735, 3827, 3834, 4017, 4027, 4172	
\int_compare_p:nNn . . .	220, 229, 241, 242, 255, 256, 1826, 1848, 2261, 2320, 2330, 2342, 2343, 2358, 2360
\int_decr:N	3379, 3738
\int_eval:n	331, 2095, 2372, 2406, 2514, 2615, 2628, 2870, 2912, 3368, 3727, 3939
\int_from_alph:n	693, 707
\int_from_roman:n	695, 709
\int_gadd:Nn	3381, 3740
\int_gdecr:N	1829, 1833, 1836, 1839, 1851
\int_gincr:N	1663, 1668, 2206, 2421, 2569, 2697, 2727, 2765, 3025, 3149, 3238, 3529, 3607, 3862, 3928
\int_gset:Nn	1873, 2674
\int_gset_eq:NN	1562, 1569, 1575, 1581, 1589, 1596, 1602, 1608, 2671
\int_gzero:N	302, 303, 304, 1350, 1367, 1959, 3084, 3201, 3683, 4038
\int_if_exist:NTF	1537, 1573, 1579, 1600, 1606, 1783
\int_incr:N	2233, 2448, 2951, 3120, 3275, 3471, 3528, 3826, 3861
\int_mod:nn	3676, 4029
\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, 1539, 1786
\int_set:Nn	689, 693, 695, 1676, 1683, 1695, 1704, 3315, 3316, 3338, 3349, 3355, 3371, 3659, 3697, 3708, 3714, 3730, 4011, 4168
\int_set_eq:NN	1664, 1669, 3378, 3737
\int_sign:n	1875
\int_step_function:nnN	2336, 2350, 2365
\int_step_inline:nnn	3317
\int_to_roman:n	190, 2316, 2354
\int_use:N	324, 329, 330, 1058, 1678, 1685, 1697, 1706, 2870, 2889, 2912, 2969, 3036, 3045, 3060, 3066, 3353, 3354, 3366, 3712, 3713, 3725
\int_zero:N	3668, 4021
\c_one_int	3338, 3357, 3363, 3369, 3373, 3376, 3697, 3716, 3722, 3728, 3732, 3735
\c_zero_int	2320, 2330, 2342, 2343, 2358, 2360, 3509, 3519, 3679, 4034
\item	32, 43, 44, 69, 80, 92, 94, 95, 97, 104
\item	80, 81, 99, 100, 105, 107, 337, 2113, 2119, 2144, 2152, 2258, 2533, 2536, 2734, 2769, 3456, 3458, 3813, 3815, 3926
\item*	5, 14, 65, 2767
item-pos*	2642
item-sym*	2642
\itemindent	25, 84
\itemindent	83
itemindent	779
\itemsep	93, 94
\itemsep	3291, 3297
\itemwidth	3345, 3389, 3393, 3704, 3748, 3752

K

keyans	13, 3096
keyans*	13, 3802
keyanspic	14, 3251
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	43, 44, 89, 92, 98, 105
align	26, 36, 82, 101
before*	43, 44, 87, 98, 105
before	43, 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	45, 88, 91
columns	26, 45, 47, 53, 88, 91
first	43, 44, 101
font	35, 82, 101
item-pos*	79
item-sym*	25, 79, 80
item*-sep	80
itemindent	26, 41, 42, 82, 101
itemsep	40, 85
labelsep	35, 80, 84, 101
labelwidth	34–38, 40, 84
label	25, 34–36, 39, 40, 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, 32, 45, 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, 45, 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, 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 461, 483, 512, 525, 572, 643, 717, 737, 781, 800, 857, 866, 945, 962, 1373, 1484, 1731, 1791, 1964, 2000, 2036, 2041, 2185, 2644, 4068, 4137

\l_keys_key_str 4253

\keys_precompile:nnN . 110, 4067, 4070, 4074, 4078, 4082, 4086, 4090

\keys_set:nn . 475, 968, 1378, 1383, 1625, 1630, 1717, 1725, 2249, 2964, 2968, 3136, 3484, 3843, 4139, 4140, 4141, 4142, 4143, 4144, 4145, 4146, 4147, 4148, 4149, 4150, 4151, 4189

keyval commands:

\keyval_parse:NNn 1498, 2066

L

label 523, 570, 643

Labels provide by enumext:

\Alph* 34, 35

\Roman* 34, 35

\alph* 34, 35

\arabic* 29, 34, 35

\roman* 34, 35

\labelsep 94

\labelsep 3292, 3295

labelsep 459

\labelwidth 35, 94

\labelwidth 3292, 3293

labelwidth 459

\leftmargin 25, 84

\leftmargin 83, 3292

legacy commands:

\legacy_if:nTF 3595, 3598, 3952, 3955

\legacy_if_gset_false:n 351

\legacy_if_set_false:n 3597, 3954

\legacy_if_set_true:n 3557, 3582, 3589, 3602, 3890, 3918, 3959

\linewidth 87, 90

\linewidth 3020, 3146, 3314, 3341, 3402, 3700, 3761

\list 32

\list 335

list-indent 779

list-offset 779

\listparindent 3294

listparindent 779

\lrbox 3612, 3965

M

\makebox 95

\makebox .. 2175, 2177, 2789, 3626, 3634, 3638, 3978, 3982

\makelabel 80, 82, 83, 95

\makelabel 82, 83, 2795, 2811

\makesavenoteenv 381

mark-ans 1962

mark-pos 1962, 1998

mark-ref 1962

mini-env 943

mini-sep 943

\minipage 32

\minipage 341

\miniright 10, 52, 1319, 3082, 3199

\miniright* 10

mode commands:

\mode_if_vertical:TF 1013, 1041, 1157, 1236

\mode_leave_vertical: .. 812, 826, 838, 850, 2144, 2152, 2173, 2787, 3624

msg commands:

\msg_error:nn .. 2231, 2236, 2446, 2451, 3127, 3131, 3218, 3278, 3474, 3829, 3836, 4152

\msg_error:nnn 548, 595, 612, 665, 1323, 1327, 1352, 1369, 1637, 1641, 1756, 4057, 4062, 4134, 4205

\msg_error:nnnn 2216, 2220, 2224, 2431, 2435, 2439, 3118, 3214, 3222, 4117

\msg_fatal:nn 2953

\msg_fatal:nnn 413

\msg_info:nnn 13, 16, 363, 377

\msg_line_context: .. 4225, 4229, 4233, 4257, 4262, 4267, 4282, 4297, 4301, 4305, 4309, 4313, 4317, 4323, 4329, 4335, 4349, 4353, 4358, 4362, 4367, 4371, 4375, 4380, 4385, 4389, 4394, 4399, 4403, 4408, 4412, 4417, 4422, 4427, 4431, 4435

\msg_log:nnn 1775, 1780, 1785

\msg_log:nnnnn 328, 1913, 1918, 1923

\msg_log:nnnnnn 320

\msg_new:nnn 4206, 4210, 4214, 4218, 4223, 4227, 4231, 4235, 4241, 4247, 4251, 4255, 4260, 4265, 4280, 4295, 4299, 4303, 4307, 4311, 4315, 4319, 4326, 4332, 4338, 4342, 4346, 4351, 4356, 4360, 4365, 4369, 4373, 4378, 4383, 4387, 4392, 4397, 4401, 4406, 4410, 4415, 4420, 4425, 4429, 4433

\msg_term:nnnn . 1740, 1745, 2879, 2889, 2918, 2923

\msg_term:nnnnn 1894

\msg_warning:nn 3081, 3198

\msg_warning:nnnn 1950, 1956, 2827, 2832, 3352, 3365, 3711, 3724

\msg_warning:nnnnn 1889, 1899

\multicolsep 88, 91

\multicolsep 3050, 3171

N

\NeedsTeXFormat 3

\newcounter 416

\NewDocumentCommand 1319, 2200, 3210, 4041, 4095, 4159

\NewDocumentEnvironment . 2416, 2928, 3096, 3251, 3444, 3802

\newlabel 34

\newlabel 399

no-store 1789

\noindent 97, 104

\noindent . 3027, 3151, 3411, 3457, 3667, 3770, 3814, 4020

\nointerlineskip 3027, 3151, 3411, 3770

noitemsep 735

\nopagebreak 1024, 1052, 1168, 1247, 1310, 1316

\normalfont 2405, 2614, 2627, 3938

nosep 735

P

Packages:

enumext 24, 36, 60, 83, 92, 112

enumitem 34

expl3 95

footnotehyper 33

hyperref 28, 29, 33, 34, 72, 77, 100, 112

lua-visual-debug 47

multicol 24, 112

shortlst 95

\par .. 1024, 1052, 1168, 1247, 1310, 1316, 1345, 1362, 2384, 3071, 3086, 3188, 3203, 3326, 3429, 3436, 3667, 3681, 3788, 3795, 4020, 4036

\parindent 3644, 3988

\parsep	46, 49, 93, 94
\parsep	2145, 2153, 2909, 3291, 3298, 3303
parsep	735
\parskip	3645, 3989
\partopsep	94
\partopsep	2910, 3296
partopsep	735
peek commands:	
\peek_meaning:NTF	3534, 3548, 3565, 3576, 3867, 3881, 3898
\peek_meaning_remove:NTF	3541, 3874
\peek_remove_spaces:n	2773
\phantomsection	33
\phantomsection	388
prg commands:	
\prg_do_nothing:	392
\prg_new_protected_conditional:Npnn ...	192
\prg_replicate:nn	209
\prg_return_false:	196
\prg_return_true:	195
\printkeyans	15, 109, 4095
prop commands:	
\prop_count:N	322, 2095, 2372, 2408, 2514, 2617, 2630, 3941
\prop_gput_if_not_in:Nnn	2093
\prop_if_exist:NTF	1773, 4061
\prop_item:Nn	4064
\prop_new:N	1776
\ProvidesExplPackage	4
R	
\raggedcolumns	3059, 3177
\ref	72, 76
ref	523, 570, 643
\refstepcounter	3604, 3961
regex commands:	
\regex_match:nnTF	194, 692, 694, 706, 708
\regex_replace_once:nnN	202
\renewcommand	558, 603, 620, 673
\RenewDocumentCommand ...	2666, 2734, 2769, 2795, 2811
\RequirePackage	17
resume	1482
resume*	1482
rightmargin	779
\Roman	35, 39, 40
\Roman	435
\roman	35, 39, 40
\roman	436, 541, 4085
S	
save-ans	1729
save-key	2034
save-ref	1962
save-sep	1962
scan commands:	
\scan_stop:	94, 3305, 3456, 3813, 4052, 4055
seq commands:	
\seq_clear:N	4166
\seq_const_from_clist:Nn	4154
\seq_count:N	323, 3264, 4170
\seq_gclear:N	2664, 2665
\seq_gput_right:Nn	2102, 2677, 2678
\seq_if_empty:NTF	2683, 4110, 4184
\seq_if_exist:NTF	1778, 4108
\seq_if_in:NnTF	4115
\seq_item:Nn	3323
\seq_map_function:NN	4175
\seq_map_inline:Nn ..	4122, 4128, 4163, 4185, 4186
\seq_map_pairwise_function:NNN	2685
\seq_new:N	110, 111, 127, 153, 154, 1781
\seq_pop_left:NN	4174
\seq_put_right:Nn	3224, 4182, 4199
\seq_set_from_clist:Nn	4167
\seq_set_map_e:NNn	4176
\seq_show:N	4112
series	1482
\setcounter	703, 707, 709, 2870, 2912, 3269
\setenumext	6, 111, 4159
\setlength	2146, 2154
show-ans	1962, 1998
show-length	855
show-pos	1998
skip commands:	
\skip_add:Nn ..	993, 999, 1005, 1015, 1019, 1043, 1047, 1137, 1143, 1149, 1159, 1163, 1185, 1238, 1242, 3291
\skip_eval:n	2145, 2153
\skip_gset:Nn	1258, 1262, 1266
\skip_gzero_new:N	1253, 1254
\skip_horizontal:N ..	827, 839, 851, 3627, 3641, 3985
\skip_horizontal:n ...	813, 2174, 2182, 2788, 2790, 3625, 3994
\skip_if_eq:nnTF ..	991, 997, 1003, 1060, 1094, 1135, 1141, 1147, 1178, 1183, 1204, 1255, 1277, 1390, 1404, 1418, 1429, 1440, 1451, 1462, 1473
\skip_new:N	70, 71, 75, 76, 77, 78, 79, 131, 172
\skip_set:Nn ..	976, 980, 1029, 1033, 1063, 1067, 1071, 1078, 1082, 1086, 1097, 1102, 1106, 1112, 1117, 1122, 1180, 1181, 1182, 1189, 1193, 1197, 1206, 1211, 1215, 1218, 1222, 1226, 1257, 1261, 1279, 1283, 1287, 1293, 1297, 1301, 3285, 3299
\skip_set_eq:NN ..	2868, 2908, 2909, 3644, 3645, 3988, 3989
\skip_use:N ..	978, 982, 1017, 1021, 1025, 1045, 1049, 1061, 1080, 1089, 1095, 1100, 1104, 1115, 1119, 1120, 1125, 1161, 1165, 1191, 1391, 1395, 1398, 1405, 1409, 1412, 3071
\skip_zero:N	2910, 3050, 3171, 3296, 3297
\skip_zero_new:N ..	1173, 1174, 1175, 1252, 1274, 1275, 1276
\c_zero_skip ..	991, 997, 1003, 1061, 1095, 1135, 1141, 1147, 1178, 1183, 1204, 1255, 1277, 1391, 1405, 1418, 1429, 1440, 1451, 1462, 1473
\small	4073, 4077, 4081, 4085, 4089, 4093
\star	2648
start	715
\stepcounter	2670, 3231
str commands:	
\c_backslash_str ..	4225, 4229, 4233, 4237, 4238, 4239, 4243, 4244, 4340, 4344, 4348, 4362, 4363, 4367, 4375, 4376, 4380, 4381, 4412, 4413, 4417, 4422, 4423
\c_colon_str	2371, 2513, 4052
\c_left_brace_str ..	4305, 4309, 4322, 4328, 4334
\c_right_brace_str ..	4305, 4309, 4322, 4328, 4334
\str_case:nn	214, 269
\str_case:nnTF	1505, 1514, 2073, 2081
\str_clear:N	2961, 3483
\str_count:n	209
\str_if_empty:NTF	1526, 1567, 1594
\str_if_eq:nnTF	2871, 2914
\str_if_in:nnTF	4048

<code>\str_new:N</code>	126, 167
<code>\str_set:Nn</code>	515, 516, 517, 1979, 1980, 2003, 2004
<code>\string</code>	381
<code>\strutbox</code>	1065, 1069, 1073, 1084, 1088, 1099, 1108, 1114, 1124, 1137, 1143, 1149, 1180, 1181, 1182, 1185, 1195, 1199, 1208, 1215, 1220, 1228, 1257, 1258, 1261, 1268, 1281, 1289, 1295, 1303, 3301
T	
T _E X and L ^A T _E X 2 _ε commands:	
<code>\@auxout</code>	397
<code>\@currenvir</code>	214, 269
<code>\protected@write</code>	397
text commands:	
<code>\text_expand:n</code>	4044
<code>\textasteriskcentered</code>	1976, 1990
<code>\thepage</code>	403
tl commands:	
<code>\c_space_tl</code>	2595, 4267, 4282, 4305, 4309
<code>\tl_clear:N</code>	488, 494, 1960, 2020, 2030, 2051, 2059, 2251, 2456, 2530, 3904
<code>\tl_clear_new:N</code>	445
<code>\tl_const:Nn</code>	38, 429
<code>\tl_gclear:N</code>	314, 315, 316, 1547, 1552, 2806, 3440, 3628, 3799
<code>\tl_gclear_new:N</code>	1534
<code>\tl_gput_right:Nn</code>	430
<code>\tl_greplace_all:Nnn</code>	451
<code>\tl_gset:Nn</code>	246, 247, 260, 261, 1535, 1548, 1553, 1772, 3571
<code>\tl_gset_eq:NN</code>	447, 2715, 3621
<code>\tl_if_blank:nTF</code>	3619
<code>\tl_if_empty:NTF</code>	546, 565, 593, 610, 630, 637, 663, 680, 1560, 1565, 1587, 1592, 1650, 1714, 1722, 1751, 1810, 1930, 2109, 2140, 2271, 2467, 2540, 2589, 2785, 3907, 4197
<code>\tl_if_empty:nTF</code>	1615, 2229, 2444
<code>\tl_if_exist:NTF</code>	1620
<code>\tl_if_novalue:nTF</code>	2247, 2464, 2538, 2574, 2668, 2693, 2711, 2716, 2745, 2959, 3262, 3481, 3841, 3905, 4161
<code>\tl_map_inline:Nn</code>	200, 448
<code>\tl_new:N</code>	35, 40, 41, 44, 45, 51, 53, 54, 55, 57, 58, 91, 92, 93, 99, 100, 101, 103, 104, 105, 106, 107, 108, 112, 115, 117, 118, 124, 125, 135, 136, 137, 144, 145, 146, 149, 166, 169
<code>\tl_put_left:Nn</code>	2117, 2150, 2256, 2601, 2636, 3923, 3926
<code>\tl_put_right:Nn</code>	446, 556, 601, 618, 671, 2121, 2156,

	2258, 2263, 2270, 2273, 2283, 2288, 2291, 2297, 2323, 2333, 2347, 2363, 2369, 2374, 2459, 2462, 2469, 2471, 2498, 2503, 2508, 2511, 2520, 2533, 2536, 2542, 2547, 2557, 3909, 3910
<code>\tl_remove_all:Nn</code>	4196
<code>\tl_remove_once:Nn</code>	2311, 2483
<code>\tl_replace_all:Nnn</code>	450
<code>\tl_reverse:N</code>	2310, 2312, 2482, 2484
<code>\tl_set:Nn</code>	46, 273, 280, 287, 415, 489, 493, 498, 499, 545, 590, 662, 810, 824, 836, 848, 1649, 1750, 2021, 2031, 2052, 2060, 2402, 2576, 2611, 2624, 2713, 3912, 3935, 4194
<code>\tl_set_eq:NN</code>	456, 551, 554, 598, 600, 615, 617, 668, 670, 2309, 2481, 2494, 2757, 2761, 3243, 3245
<code>\tl_to_str:n</code>	1620, 1626, 1631, 4044
<code>\tl_trim_spaces:n</code>	446, 4182, 4194, 4200
<code>\tl_use:N</code>	452, 455, 567, 632, 639, 682, 881, 885, 889, 893, 897, 901, 905, 909, 913, 917, 921, 925, 929, 933, 937, 941, 2179, 2316, 2324, 2335, 2349, 2354, 2366, 2700, 2706, 2730, 2748, 2752, 2760, 2797, 2798, 2805, 2813, 2814, 2820, 2935, 3102, 3248, 3434, 3631, 3642, 3646, 3793, 3975, 3986, 3992, 3997, 4098, 4099, 4100, 4101, 4102, 4120, 4178
token commands:	
<code>\token_to_str:N</code>	399
<code>\topsep</code>	2146, 2154
<code>topsep</code>	735
<code>\typeout</code>	367, 370, 380, 381
U	
<code>\u</code>	203
use commands:	
<code>\use:N</code>	210, 2802, 2937
<code>\use:n</code>	1496, 2064, 4050
<code>\use_none:nn</code>	391
<code>\usecounter</code>	2869, 2911
V	
<code>\value</code>	1563, 1569, 1576, 1582, 1590, 1596, 1603, 1609
<code>\vspace</code>	352, 1395, 1398, 1409, 1412, 1422, 1424, 1433, 1435, 1444, 1446, 1455, 1457, 1466, 1468, 1477, 1479, 2145, 2153, 3259, 3270, 3682, 4037
W	
<code>widest</code>	715
<code>wrap-ans</code>	1962
<code>wrap-label</code>	459
<code>wrap-label*</code>	459
<code>wrap-opt</code>	1962