

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-24^{*}

©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	3	4	The storage system	10
1.1	Description and usage	4	4.1	Keys for storage	10
1.2	The concept of left margin	4	4.2	Keys for internal label and ref	11
1.3	User interface	4	4.3	Keys for debugging and checking	11
1.3.1	Internal counters	4	4.4	The command <code>\anskey</code>	11
1.3.2	Support for <code>multicol</code>	5	4.5	The environment <code>keyans</code>	12
1.3.3	Support for <code>minipage</code>	5	4.5.1	The <code>\item*</code> in <code>keyans</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	5	4.6	The environment <code>keyanspic</code>	13
1.3.5	Support for <code>\footnote</code>	5	4.6.1	The command <code>\anspic</code>	13
2	The environment <code>enumext</code>	5	4.7	Printing stored content	14
2.1	The <code>\item*</code> in <code>enumext</code>	6	4.7.1	The command <code>\getkeyans</code>	14
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	6	4.7.2	The command <code>\printkeyans</code>	14
3	The command <code>\setenumext</code>	6	5	Full examples	15
3.1	Keys for <code>label</code> and <code>ref</code>	7	6	The way of non-enumerated lists	17
3.2	Keys for spaces	7	7	References	19
3.2.1	Vertical spaces	8	8	Change history	19
3.2.2	Horizontal spaces	8	9	Index of Documentation	20
3.3	Keys for <code>add code</code>	9	10	Implementation	22
3.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	9	11	Index of Implementation	112
3.5	Keys for <code>multicols</code>	10			
3.6	Keys for <code>minipage</code>	10			
3.6.1	The command <code>\miniright</code>	10			
3.6.2	The key <code>miniright</code>	10			

Motivation and acknowledgments

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

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

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

1. Answer given by Alan Munn in `\topsep`, `\itemsep`, `\partopsep`, `\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 multicols, vertical alignment

^{*}This file describes a documentation for v1.0, last revised 2024-05-24.

[†]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 (l^{pp}l), version 1.3 or later (<https://www.latex-project.org/l^{pp}l.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_TE_X. It has been tested with the standard classes provided by L^AT_EX: `book`, `report`, `article` and `letter` on 10pt, 11pt and 12pt.

1 Introduction

In the \LaTeX 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) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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) \LaTeX 2e 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

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:

1. (B), $x = 5$

2. (D)

3. (C), some note
4. (B)

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 `mktxlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

<code>enumext.sty</code>	<code>>> TDS:tex/latex/enumext/</code>
<code>enumext.pdf</code>	<code>>> TDS:doc/latex/enumext/</code>
<code>README.md</code>	<code>>> TDS:doc/latex/enumext/</code>
<code>enumext.dtx</code>	<code>>> TDS:source/latex/enumext/</code>

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.

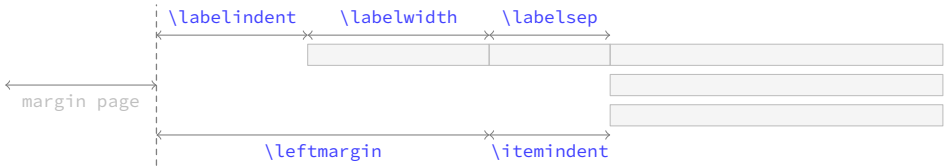


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

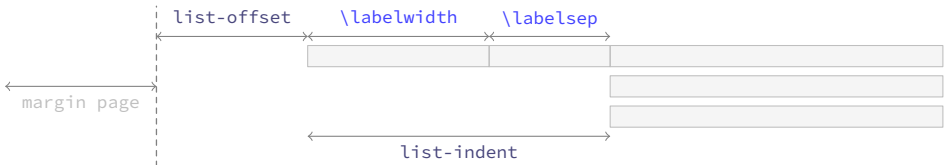


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

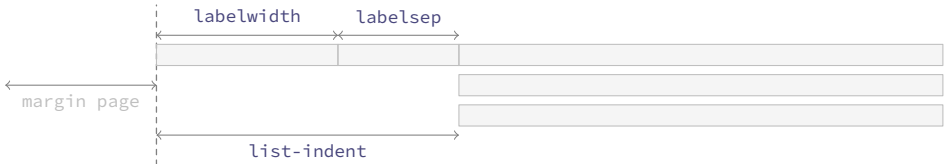


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

The user interface consists in `enumext`, `enumext*`, `keyans`, `keyans*` and `keyanspic` environments, `\anskey`, `\item*` and `\anspic*` commands to \langle stored content \rangle , `\getkeyans` command to get the individual \langle stored content \rangle , `\printkeyans` to print all \langle stored content \rangle , `\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.

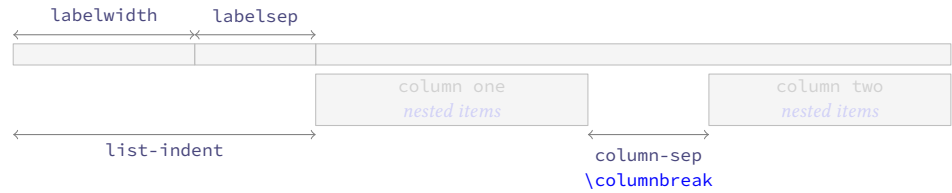


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §3.5).

- 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 §3.6).

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 (§3.1), the standard `\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` (§4.2) when the key `save-ans` (§4.1) 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, but, it will not produce the expected links, and when using the `mini-env` key or the starred environments `enumext*` and `keyans*` the output will look like the classic way they are displayed in the `minipage` environment.

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 environment enumext

<code>enumext</code>	<code>\begin{enumext} [⟨keyval list⟩]</code>	<code>\begin{enumext*} [⟨keyval list⟩]</code>
<code>enumext*</code>	<code>\item ⟨item content⟩</code>	<code>\item ⟨item content⟩</code>
	<code>\item [⟨custom⟩] ⟨item content⟩</code>	<code>\item [⟨custom⟩] ⟨item content⟩</code>
	<code>\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩</code>	<code>\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩</code>
	<code>\end{enumext}</code>	<code>\end{enumext*}</code>

The `enumext` is an “*enumerated list*” environment that works in the same way as the standard `enumerate` environment provided by L^AT_EX, `\item` and `\item[⟨custom⟩]` 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 `[⟨key = val⟩]` in the environment.

Example

- 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.

```
\begin{enumext}
  \item This text is in the first level.
  \begin{enumext}
    \item This text is in the second level.
    \begin{enumext}
      \item This text is in the third level.
      \begin{enumext}
        \item This text is in the fourth level.
      \end{enumext}
    \end{enumext}
  \end{enumext}
  \item[X] This text is in the first level.
  \item* This text is in the first level.
\end{enumext}
```

2.1 The \item* in enumext

```
\item* \item*
\item*[⟨symbol⟩]
\item*[⟨symbol⟩][⟨offset⟩]
```

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. The default values for `⟨symbol⟩` and `⟨offset⟩` are `\star ‘★’` and the value set by `labelsep` key.

The *starred version* ‘★’ 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` environment is NOT the same as in the `keyans` environment.

2.1.1 Keys for \item* in enumext

- `item-sym* = {⟨symbol⟩}` default: `\star`
Sets the `symbol` to be displayed in the “*left*” of the box containing the current `⟨label⟩` set by `labelwidth` key for `\item*` in `enumext`. The `symbol` can be in text or math mode, for example `item-sym*={\ast$}`.
- `item-pos* = {⟨rigid length | dim expression⟩}` default: `by levels`
Sets the `offset` between the box containing the current `⟨label⟩` defined by `labelwidth` key and the `⟨symbol⟩` 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*.

3 The command \setenumext

```
\setenumext \setenumext[⟨enumext, level⟩][⟨key = val⟩] \setenumext[⟨enumext*⟩][⟨key = val⟩]
\setenumext[⟨print, level⟩][⟨key = val⟩] \setenumext[⟨keyans*⟩][⟨key = val⟩]
\setenumext[⟨keyans⟩][⟨key = val⟩] \setenumext[⟨print*⟩][⟨key = val⟩]
```

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

The `⟨keys⟩` 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.

- It should be kept in mind that using any *key* 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* and *keyanspic* environments. All *keys* 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.

3.1 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 first level are `\arabic*`, for second level are `(\alph*)`, for third level are `\roman*`, and for fourth level are `\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., `\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⟩]`.

3.2 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`.

3.2.1 Vertical spaces

`topsep` = {*rubber length* | *rigid length*} 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 values for first level 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.

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

Set the *vertical space* between paragraphs within an item. Internally sets the value of `\parsep` for the current level. The default values for first level 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.

`partopsep` = {*rubber length* | *rigid length*} 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 are 2.0pt plus 1.0pt minus 1.0pt, for third and fourth level are 1.0pt minus 1.0pt.

- The value of this parameter also affects the *inner levels* and the `keyans` environment. 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 *vertical mode* and apply this value to the “top” and “bottom” the environment or nested level.

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

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

`noitemsep` *value forbidden* 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` *value forbidden* 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 *keys* 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 ‘*’ *keys* applies `\vspace*` so that T_EX does *not discard* this space at page break.

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

Set the *extra vertical 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*` = {*rubber length* | *rigid length*} default: *not used*

Set the *extra vertical 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”.

3.2.2 Horizontal spaces

`itemindent` = {*rigid length*} 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` = {*rigid length*} 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` 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.

3.3 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 L^AT_EX 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.

3.4 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*.

3.5 Keys for multicols

`columns = {⟨integer⟩}` default: 1

Set the *number of columns* to be used by the `multicols` 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 `multicols` 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 `multicols` 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.

3.6 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`.

3.6.1 The command `\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 version ‘*’ 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.

3.6.2 The key `miniright`

In the horizontal list environments `enumext*` and `keyans*` it is not possible to use the `\miniright` command and the `miniright` key must be used instead.

`miniright = {⟨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`.

`miniright* = {⟨code for drawing or tabular⟩}` default: *not used*

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

4 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.

<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \begin{keyans} ... \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \begin{keyanspic} ... \end{keyanspic} \end{enumext}</pre>
--	--

4.1 Keys for storage

`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..

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

Wraps the *current argument* passed `\anskey` command to referenced by `{#1}`. 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. 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 the double `{##1}`.

`save-sep = {<text symbol>}` default: {, }

Sets the *text symbol* that will separate the current *<label>* defined by the `label` key from the *<optional argument>* (if present), when storing them in the *<store name>* defined by the `save-ans` key for the `\item*` command in the `keyans` and `keyans*` environment and for the `\anspic` command in the `keyanspic` environment. 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.

`mark-ans = {<symbol>}` default: \textasteriskcentered

Sets the *symbol* to be displayed in the left margin of the “*stored content*” in *<store name>* set by `save-ans` key when using `show-ans` key.

`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.

4.2 Keys for internal 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.

4.3 Keys for debugging and checking

`show-ans = {<true | false>}` default: false

Displays the *current <argument>* passed to `\anskey` in `enumext` environment, the current *<label>* for `\item*` in `keyans` environment and the current *<label>* for `\anspic*` in `keyanspic` environment at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown in square brackets.

`show-pos = {<true | false>}` default: false

Displays the *position* occupied by the “*stored content*” by `\anskey` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment in *<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. This key works under the logic that each question will contain “*only one answer*”, it is intended to be used in conjunction with `no-store` key.

`no-store <value forbidden>` default: not used

This is a *meta-key* that does not receive an argument. This key is used in conjunction with `check-ans` and is designed to be used with nested levels of `enumext` in which the `\anskey` command will not be used.

4.4 The command \anskey

`\anskey <anskey>{<content>}`

The `\anskey` command takes a mandatory argument and is triggered by `save-ans` key. The “*content*” are “*stored*” in *<store name>* set by `save-ans` key. The command does “*not support*” verbatim content and must NOT be nested. By design it is assumed that each `\item` or `\item*` will have a “*single*” occurrence of the command unless a nested level is opened or the `no-store` key is used. 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.

Example

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

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

4.5 The environment keyans

```
keyans \begin{keyans}[\key = val] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}
keyans* \begin{keyans*}[\key = val] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans*}
```

The `keyans` is an “*enumerated list*” environment designed for “*multiple choice*” questions activated by the `save-ans` key. This environment 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.

```
\begin{enumext}[save-ans=test]
  \item \langle item content \rangle
    \begin{keyans}[\key = val]
      \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}
```

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

4.5.1 The `\item*` in `keyans`

```
\item* \item*
\item* [\langle content \rangle]
```

The `\item*` and `\item*[\langle content \rangle]` command store the current `\label` set by `label` key next to the `\content` (if it is present) in `\store name` set by `save-ans` key in the “*first level*” of the `enumext` environment.

The *starred version* ‘`*`’ 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 *starred version* ‘`*`’ will only appear “*once*” within the environment.

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

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.
    \begin{keyans}[nosep]
      \item Choice
      \item* Correct 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
2. Text containing a question and image.

(A) Choice

(B) Choice

(C) Choice

(D) Choice

* (E) [note] Correct choice



Some text

4.6 The environment keyanspic

keyanspic

`\begin{keyanspic}[\langle number above, number below \rangle]\anspic{\langle drawing \rangle}\anspic*[\langle content \rangle]{\langle drawing \rangle}`

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.

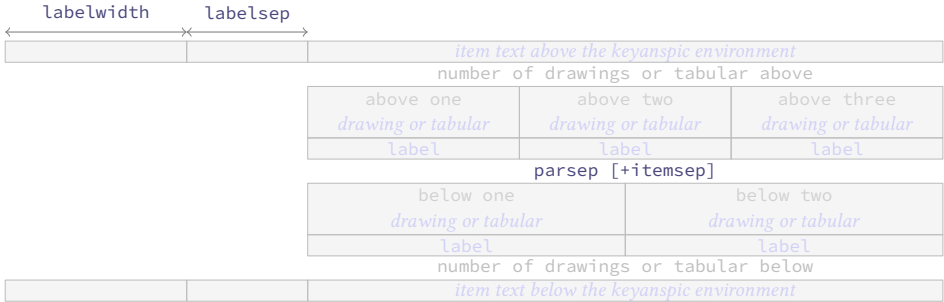


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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.

4.6.1 The command \anspic

\anspic

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

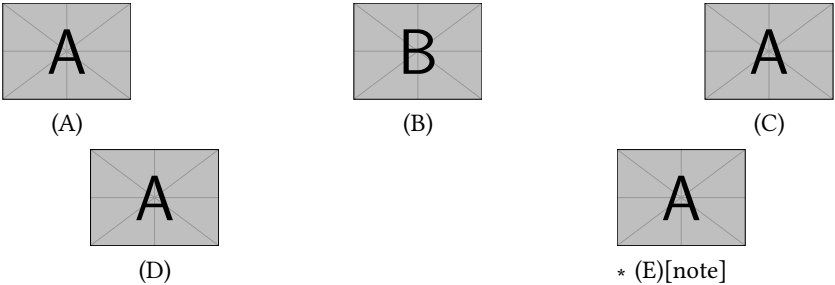
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

The *starred version* “*” 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 version* “*” 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.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

The command `\getkeyans` prints the “only stored content” in `<store name>` defined by `save-ans` key in the `<position>` returned by the `show-pos` key.

The “content” can only be accessed “after” it is stored, if the `<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.

4.7.2 The command \printkeyans

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

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

Internally it places the “stored content” inside the `enumext` environment with default values for `label` key are the same as those of the `enumext` environment along with the keys: `nosep`, `first=\small`, `font=\small` for all levels, except for the first one that adds the `columns=2` key.

The optional argument allows to handle the `<keys>` “on the first level” of the `enumext` environment encapsulated by the command. If need to pass options for nested levels use `\setenumext[<print> , <level>]{<store name>}`.

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1] $3(x + y + z)$
2. True False

(a) ~~LaTeXe~~ is cool?

[2] Very True!
3. Related to Linux

(a) You use linux?
- [3] Yes

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

5 Full examples

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

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

and then you can use the excellent [arara](#)¹ tool to compile them.

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

✓

 A

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

B

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

C

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

D

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

2. A

3. B

4. A

*

*

*

*

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

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

 value
2. Second type of questions
- I. $2\alpha + 2\delta = 90^\circ$

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

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

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

A

 value

B

 value

C

 value

D

 value

E

 value
4. Question with image and label below:



A



B



C



D



E

5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

1. B, $x = 5$
2. D
3. C, some note
4. E, A duck
5. D, other note

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

 True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?
- 4

 Related to Linux
- (a)

 You use linux?
- (b)

 Usually uses the package manager?
- (c)

 Rate the following package and class
- i.

 xsim-exam
- ii.

 xsim
- iii.

 exsheets

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

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

Example 5

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

1	Which choice best describes what happens in the passage? A) One character argues with another character who intrudes on her home. B) One character receives a surprising request from another character. C) One character reminisces about choices she has made over the years. D) One character criticizes another character for pursuing an unexpected course of action.	3	Which choice best describes what happens in the passage? A) One character argues with another character who intrudes on her home. B) One character receives a surprising request from another character. C) One character reminisces about choices she has made over the years. D) One character criticizes another character for pursuing an unexpected course of action.
2	Which choice best describes what happens in the passage? A) One character argues with another character who intrudes on her home. B) One character receives a surprising request from another character. C) One character reminisces about choices she has made over the years. D) One character criticizes another character for pursuing an unexpected course of action.	4	Which choice best describes what happens in the passage? A) One character argues with another character who intrudes on her home. B) One character receives a surprising request from another character. C) One character reminisces about choices she has made over the years. D) One character criticizes another character for pursuing an unexpected course of action.

1. A) 2. C) 3. B) 4. D)

6 The way of non-enumerated lists

It is possible to use (or abuse) the enumext environment to mimic non-enumerated list environments such as itemize and description, clearly the <keys> to “store answers”, the keyans and keyanspic environments lose their sense and it is not the focus of the main of this package, but, why not to do it?. Here I leave as an example other uses of the enumext environment that can be helpful for specific purposes. The “trick” to generate these fake environments is set label={} or label={<some>} and play with the list-indent, list-offset, font and wrap-label keys.

Fake itemize environment

Here we set the label key using the default settings in L^AT_EX for the four levels \textbullet, \textendash, \textasteriskcentered and \textperiodcentered together with the nosepe key to reduce the vertical spaces in the left side example and set the label key in mathematical mode for the right side as \ast, \diamond, \circ and \star for the four levels together with the nosepe key

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

Fake description environment

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

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

Something A short one-line description text.

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

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

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

Something A short one-line description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

Something 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.
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.

7 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.

8 Change history

v1.0 2024-05-24 – First public release.

9 Index of Documentation

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

C

Document class:

article 2

book 2

exam 3

letter 2

report 2

\columnbreak 5

\columnsep 10

Commands provide by enumext:

\anskey 4, 10–12

\anspic* 4, 10, 11, 13

\anspic 11, 13

\getkeyans 4, 11, 14

\item* 4–7, 10–12

\item 6, 7, 9–12

\miniright 4, 5, 10

\printkeyans 4, 6, 11, 14

\setenumext 4, 6, 7, 10–12, 14

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

enumext* 4, 5, 9, 10

enumext 4–6, 9–12, 14, 17

keyans* 4, 5, 10, 11

keyanspic 4, 7, 10, 11, 13, 17

keyans 4–13, 17

Environments:

enumerate 1, 3, 4, 6, 19

list 4, 9, 19

minipage 3–5, 10, 19

multicols 3, 5, 10

I

\item 4, 5

\itemsep 8

K

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 18

before* 9

before 9

below* 8

below 8

check-ans 11

columns-sep 5, 10

columns 5, 8, 10

first 9

font 7

item-pos* 6

item-sym* 6

itemindent 8

itemsep 8, 13

labelsep 4, 6–11, 18

labelwidth 4, 6, 7, 9–11, 18

label 7, 9, 11, 12, 14, 17, 18

list-indent 4, 8, 9

list-offset 4, 8, 18

listparindent 8

mark-ans 11

mark-pos 11

mark-ref 11

mini-env 5, 8, 10

mini-sep 5, 10

miniright* 10

miniright 10

no-store 11

noitemsep 8

nosep 8, 17

parsep 8, 13

partopsep 8

ref 5, 7

resume* 9

resume* 9

resume 9

rightmargin 8

save-ans 5, 9–14

save-key 9

save-ref 5, 7, 11, 14

save-sep 11

series 9

show-ans 11

show-length 7

show-pos 11, 14

start 9

topsep 8

widest 7

wrap-ans 10

wrap-label* 7, 18

wrap-label 7

wrap-opt 10

L

\label 5

Labels provide by enumext:

\Alph* 7, 12

\Roman* 7

\alph* 7

\arabic* 7

\roman* 7

\labelsep 4, 7

\labelwidth 4, 7

\linewidth 10

\listparindent 8

P

Packages:

enumerate 18

enumext 1–4, 13, 18, 19

enumitem 4, 5, 9, 18, 19

©2024 by Pablo González L

20 / 124

footnotehyper	5	R	
hyperref	5, 11, 19	\raggedcolumns	5
l3prop	1, 19	\ref	5
l3seq	1, 19	\rightmargin	8
multicol	1, 2, 5, 19		
xsim	3	T	
\parsep	8		
\partopsep	8	\topsep	8

10 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.

10.1 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.

10.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

Identify the internal prefix (L^AT_EX3 DocStrip convention) for l3doc class.

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

10.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L^AT_EX to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-24}
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 }
```

10.4 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 boolean variables will be used to determine if they are present (nested) in each other. 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.

```
20 \int_new:N \__enumext_level_int
21 \int_new:N \__enumext_level_h_int
22 \int_new:N \__enumext_keyans_level_int
23 \int_new:N \__enumext_keyans_level_h_int
24 \int_new:N \__enumext_keyans_pic_level_int
25 \bool_new:N \__enumext_starred_bool
26 \bool_new:N \g__enumext_starred_bool
```

```

27 \bool_new:N \__enumext_starred_first_level_bool
28 \bool_new:N \__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_first_level_bool
31 \bool_new:N \l__enumext_keyans_env_bool

```

(End of definition for `\l__enumext_level_int` 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` (§10.8) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§10.11).

```

32 \cs_set_protected:Npn \__enumext_tmp:n #1
33 {
34   \tl_new:c { l__enumext_counter_#1_tl }
35 }
36 \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 (§10.11).

```

37 \tl_const:Nn \c__enumext_counter_style_tl
38 { { arabic } { roman } { Roman } { alph } { Alph } }
39 \tl_new:N \l__enumext_ref_key_arg_tl
40 \tl_new:N \l__enumext_ref_the_count_tl
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { l__enumext_renew_the_count_#1_tl }
44   \tl_new:c { l__enumext_the_counter_#1_tl }
45   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
46 }
47 \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 (§10.27).

```

48 \int_new:N \g__enumext_resume_int
49 \int_new:N \g__enumext_resume_vii_int
50 \tl_new:N \l__enumext_resume_name_tl
51 \bool_new:N \l__enumext_resume_active_bool
52 \tl_new:N \g__enumext_item_symbol_tl
53 \tl_new:N \g__enumext_standar_series_tl
54 \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` (§10.12) and `label` (§10.10) keys.

```

55 \dim_new:N \l__enumext_current_widest_dim
56 \tl_new:N \g__enumext_counter_styles_tl
57 \tl_new:N \g__enumext_widest_label_tl
58 \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 (§10.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§10.31.1) which determines the internal values for `\leftmargin` and `\itemindent`.

```

59 \cs_set_protected:Npn \__enumext_tmp:n #1
60 {
61   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
62   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
63   \dim_new:c { l__enumext_leftmargin_#1_dim }
64   \dim_new:c { l__enumext_itemindent_#1_dim }
65 }
66 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
```

Internal variables used by `columns` key §10.18).

```
67 \cs_set_protected:Npn \__enumext_tmp:n #1
68 {
69   \skip_new:c { \l__enumext_multicols_above_#1_skip }
70   \skip_new:c { \l__enumext_multicols_below_#1_skip }
71 }
72 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by `\miniright` command (§10.19.4) and the keys `miniright`, `miniright*`, `mini-env` and `mini-sep` (§10.17, §10.19).

```
73 \int_new:N \g__enumext_minipage_stat_int
74 \skip_new:N \l__enumext_minipage_left_skip
75 \skip_new:N \l__enumext_minipage_right_skip
76 \skip_new:N \l__enumext_minipage_after_skip
77 \skip_new:N \g__enumext_minipage_right_skip
78 \skip_new:N \g__enumext_minipage_after_skip
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \dim_new:c { \l__enumext_minipage_left_#1_dim }
82   \bool_new:c { \l__enumext_minipage_active_#1_bool }
83 }
84 \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 (§10.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§10.10). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```
85 \cs_set_protected:Npn \__enumext_tmp:n #1
86 {
87   \bool_new:c { \l__enumext_wrap_label_#1_bool }
88   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
89   \int_new:c { \l__enumext_start_#1_int }
90   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
91   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
92   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
93   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
94   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
95 }
96 \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` (§10.25) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.29.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§10.34.1) for the `keyanspic` environment.

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

```
97 \bool_new:N \l__enumext_store_active_bool
98 \tl_new:N \l__enumext_store_name_tl
99 \tl_new:N \g__enumext_store_name_tl
100 \tl_new:N \l__enumext_store_anskey_arg_tl
101 \int_new:N \l__enumext_store_columns_join_int
102 \tl_new:N \l__enumext_store_keyans_label_tl
103 \tl_new:N \l__enumext_store_keyans_item_opt_tl
104 \tl_new:N \l__enumext_keyans_item_opt_tl
105 \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` (§10.39).

```
106 \tl_new:N \l__enumext_setkey_tmpa_tl
107 \tl_new:N \l__enumext_setkey_tmpb_tl
108 \int_new:N \l__enumext_setkey_tmpa_int
109 \seq_new:N \l__enumext_setkey_tmpa_seq
110 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

```
\l__enumext_store_opt_X_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_columns_X_bool
\l__enumext_store_columns_X_int
\l__enumext_store_columns_sep_X_bool
\l__enumext_store_columns_sep_X_dim
\l__enumext_store_upper_level_X_bool
```

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.38) and the keys `columns*` and `columns-sep*`.

```
111 \cs_set_protected:Npn \l__enumext_tmp:n #1
112 {
113   \tl_new:c { \l__enumext_store_opt_#1_tl }
114   \tl_new:c { \l__enumext_print_keyans_#1_tl }
115   \bool_new:c { \l__enumext_store_columns_#1_bool }
116   \int_new:c { \l__enumext_store_columns_#1_int }
117   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
118   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
119   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
120 }
121 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_store_opt_X_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` (§10.25), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.24).

```
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 (§10.34.2).

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

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

```
\l__enumext_store_ans_bool
\l__enumext_check_ans_bool
\g__enumext_check_ans_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 (§10.23) 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_store_ans_bool
133 \bool_new:N \l__enumext_check_ans_bool
134 \bool_new:N \g__enumext_check_ans_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_store_ans_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 (§10.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with `key hyperfootnotes=true`.

```
142 \bool_new:N \l__enumext_hyperref_bool
143 \bool_new:N \l__enumext_footnotes_key_bool
```

(End of definition for `\l__enumext_hyperref_bool` and `\l__enumext_footnotes_key_bool`.)

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_store_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

Internal variables are used when executing the `save-ref` key. The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the labels defined by level on which operations will be performed.

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

```

144 \tl_new:N \l__enumext_newlabel_arg_one_tl
145 \tl_new:N \l__enumext_newlabel_arg_two_tl
146 \tl_new:N \l__enumext_store_write_aux_file_tl
147 \cs_set_protected:Npn \__enumext_tmp:n #1
148 {
149   \tl_new:c { l__enumext_label_copy_#1_tl }
150 }
151 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

Internal variables used for redefinition of `\footnote`.

```

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

```

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

```

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

```

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

```

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

```

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

```

\c__enumext_all_envs_clist

```

An internal `clist-var` variable to run with `__enumext_tmp:n`.

```

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

```

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 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 `minirigth` and `minirigth*` 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 { \l__enumext_level_int }
191 }
```

(End of definition for `__enumext_level:`.)

`__enumext_if_is_int:nT` 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?](#)

```
\__enumext_if_is_int:nF
\__enumext_if_is_int:nTF
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{\l__enumext_ref_the_count_tl}\cE} } \l__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 } { \l__enumext_#2_#3_#1 } \\
211 }
```

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

`__enumext_is_not_nested:` 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 { \@currenvir }
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 }
```

```

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_standard_first_level_bool` and `\l__enumext_standard_first_level_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` to use in messages related to the `check-ans` key.

```

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_level_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_level_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 { \@currentenvir }
270     {
271         {keyans}
272         {
273             \tl_set:Nn \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:Nn \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:.)

__enumext_reset_global_vars:

```

294 \cs_new_protected:Nn \__enumext_reset_global_vars:
295 {
296     \int_gzero:N \g__enumext_item_number_int
297     \int_gzero:N \g__enumext_item_anskey_int
298     \int_gzero:N \g__enumext_item_answer_diff_int
299     \bool_gset_false:N \g__enumext_check_ans_bool
300     \bool_gset_false:N \g__enumext_standar_bool
301     \bool_gset_false:N \g__enumext_starred_bool
302     \tl_gclear:N \g__enumext_store_name_tl
303     \tl_gclear:N \g__enumext_start_line_tl
304     \tl_gclear:N \g__enumext_envir_name_tl
305 }

```

(End of definition for __enumext_reset_global_vars:.)

__enumext_log_global_vars:

```

306 \cs_new_protected:Nn \__enumext_log_global_vars:
307 {
308     \msg_log:nneeee { enumext } { prop-seq-int-hook }
309     { \g__enumext_store_name_tl } { \prop_count:c { \g__enumext_ \g__enumext_store_name_tl _prop
310     { \seq_count:c { \g__enumext_ \g__enumext_store_name_tl _seq } }
311     { \int_use:c { \g__enumext_resume_ \g__enumext_store_name_tl _int } }
312 }

```

(End of definition for __enumext_log_global_vars:.)

__enumext_log_answer_vars:

```

313 \cs_new_protected:Nn \__enumext_log_answer_vars:
314 {
315     \msg_log:nneee { enumext } { item-answer-hook }
316     { \int_use:N \g__enumext_item_number_int }
317     { \int_use:N \g__enumext_item_anskey_int }
318     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
319 }

```

(End of definition for __enumext_log_answer_vars:.)

__enumext_execute_after_env:

The function __enumext_execute_after_env: will perform the comparison between the \item in the environments and the \item's with answers and return the appropriate message. As this function is passed to the function __enumext_after_env:nn for the environments enumext and enumext* we must make sure that we are not nested at any level and finally reset our global variables.

```

320 \cs_new_protected:Nn \__enumext_item_answer_diff:
321 {
322     \int_gset:Nn \g__enumext_item_answer_diff_int
323     {
324         %\int_eval:n
325         % {
326             \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
327         % }
328     }
329 }
330 \cs_new_protected:Nn \__enumext_execute_after_env:
331 {
332     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
333     {
334         \tl_if_empty:NF \g__enumext_store_name_tl

```

```

335         {
336             \__enumext_item_answer_diff:
337             \msg_term:nnVV
338             { enumext } { save-ans-log-hook }
339             \g__enumext_envir_name_tl \g__enumext_store_name_tl
340             \__enumext_log_global_vars:
341             \__enumext_log_answer_vars:
342             \bool_if:NT \g__enumext_check_ans_bool
343             {
344                 \__enumext_check_ans_show:
345             }
346         }
347     \__enumext_reset_global_vars:
348 }
349 }

```

(End of definition for __enumext_execute_after_env:.)

10.6 Copying list and minipage environments

The `list` environment provided by \LaTeX 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.

```

350 \__enumext_at_begin_document:n
351 {
352     \cs_new_eq:NN \__enumext_start_list:nn \list
353     \cs_new_eq:NN \__enumext_stop_list: \endlist
354     \cs_new_eq:NN \__enumext_item_std:w \item
355 }

```

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

The `minipage` environment provided by \LaTeX 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.

```

356 \__enumext_at_begin_document:n
357 {
358     \cs_new_eq:NN \__enumext_minipage:w \minipage
359     \cs_new_eq:NN \__enumext_endminipage: \endminipage
360 }

```

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

10.7 Compatibility with hyperref and footnotehyper

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

```

361 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
362 \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”.

```

363 \cs_new_protected:Nn \__enumext_after_hyperref:
364 {
365     \IfPackageLoadedTF { hyperref }
366     {
367         \msg_info:nnn { enumext } { package-load } { hyperref }

```

```

368     \bool_set_true:N \l__enumext_hyperref_bool
369     \IfHyperBoolean{hyperfootnotes}
370     {
371         \typeout{hyperfootnotes=true}
372         \bool_set_true:N \l__enumext_footnotes_key_bool
373     }
374     { \typeout{hyperfootnotes=false} }
375 }
376 { }

```

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`.

```

377     \bool_if:NT \l__enumext_footnotes_key_bool
378     {
379         \IfPackageLoadedTF { footnotehyper }
380         {
381             \msg_info:nnn { enumext } { package-load } { footnotehyper }
382         }
383         {
384             \typeout{No ~ footnotehyper ~ load}
385             \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
386             \bool_set_false:N \l__enumext_footnotes_key_bool
387         }
388     }

```

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.

```

389     \bool_if:NTF \l__enumext_hyperref_bool
390     {
391         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
392         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
393     }
394     {
395         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
396         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
397     }
398 }

```

(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.

```

399 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
400 {
401     \protected@write \@auxout { }
402     {
403         \token_to_str:N \newlabel {#1}
404         {
405             {#2}
406             \bool_if:NT \l__enumext_hyperref_bool
407             { { \thepage } {#2} {#1} }
408             { }
409         }
410     }
411     \__enumext_hypertarget:nn {#1} { }
412     \__enumext_phantomsection:
413 }

```

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

10.8 Definition of counters

```
\__enumext_define_counters:Nn
\__enumext_define_counters:cn
```

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

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

```
414 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
415 {
416   \cs_if_exist:cTF { c@ #2 }
417   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
418   {
419     \tl_set:Nn #1 { #2 }
420     \newcounter { #2 }
421   }
422 }
```

(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    423 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii   424 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii  425 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv   426 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXvii  427 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXviii 428 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
           429 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
           430 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
```

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

10.9 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```
431 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
432 {
433   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
434   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
435 }
436 \__enumext_register_counter_style:Nn \arabic { 0 }
437 \__enumext_register_counter_style:Nn \Alph { M }
438 \__enumext_register_counter_style:Nn \alph { m }
439 \__enumext_register_counter_style:Nn \Roman { VIII }
440 \__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.

```
441 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
442 {
443   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
444   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
445 }
446 \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.

```
447 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
```



```

448 {
449   \tl_clear_new:N #1
450   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
451   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
452   \tl_map_inline:Nn \g__enumext_counter_styles_tl
453   {
454     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
455     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
456     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
457   }
458   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
459   { \tl_use:N \g__enumext_widest_label_tl }
460   \tl_set_eq:cN { the #2 } #1
461 }
462 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

10.10 Setting keys associated with label

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

```

463 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
464 {
465   \keys_define:nn { enumext / #1 }
466   {
467     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
468     font .value_required:n = true,
469     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
470     labelsep .initial:n = {0.3333em},
471     labelsep .value_required:n = true,
472     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
473     labelwidth .value_required:n = true,
474     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
475     wrap-label .initial:n = {##1},
476     wrap-label .value_required:n = true,
477     wrap-label* .code:n = {
478       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
479       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
480     },
481     wrap-label* .value_required:n = true,
482   }
483 }
484 \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.

```

485 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
486 {
487   \keys_define:nn { enumext / #1 }
488   {
489     align .choice:,
490     align / left .code:n =
491     {
492       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
493       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
494     },
495     align / right .code:n =
496     {
497       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
498       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
499     },
500     align / center .code:n =
501     {
502       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
503       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
504     },

```

```

505     align .initial:n = left,
506     align .value_required:n = true,
507   }
508 }
509 \clist_map_inline:nn
510 {
511   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
512 }
513 { \__enumext_tmp:nn #1 }

514 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
515 {
516   \keys_define:nn { enumext / #1 }
517   {
518     align .choice:,
519     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
520     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
521     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
522     align .initial:n = left,
523     align .value_required:n = true,
524   }
525 }
526 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for *align*.)

10.11 Setting label and ref keys

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

10.11.1 Define and set label and ref keys for enumext environment

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

```

label
ref
\l__enumext_label_i_tl 527 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 528 {
\l__enumext_label_iii_tl 529   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 530   {
531     label .code:n = {
532       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
533       { l__enumext_counter_#2_tl } {##1}
534       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
535       \l__enumext_current_widest_dim
536     },
537     label .initial:n = #3,
538     label .value_required:n = true,
539     ref .code:n = \__enumext_standar_ref:n {##1},
540     ref .value_required:n = true,
541   }
542 }
543 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
544 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
545 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
546 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

```

547 \cs_new_protected:Npn \__enumext_standar_ref:n #1
548 {
549   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
550   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
551   {
552     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
553   }

```

```

554     {
555       \tl_set_eq:Nc
556       \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
557       \__enumext_regex_counter_style:
558       \tl_set_eq:Nc
559       \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
560       \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
561       {
562         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
563         { \exp_not:V \l__enumext_ref_key_arg_tl }
564       }
565     }
566   }

```

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

```

567 \cs_new_protected:Nn \__enumext_standar_ref:
568 {
569   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
570   {
571     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
572   }
573 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
574 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
575 {
576   \keys_define:nn { enumext / #1 }
577   {
578     label .code:n = {
579       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
580       { \l__enumext_counter_#2_tl } {##1}
581       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
582       \l__enumext_current_widest_dim
583     },
584     label .initial:n = #3,
585     label .value_required:n = true,
586     ref .code:n = \__enumext_starred_ref:n {##1},
587     ref .value_required:n = true,
588   }
589 }
590 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
591 \__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`.

```

592 \cs_new_protected:Npn \__enumext_starred_ref:n #1
593 {
594   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
595   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
596   {
597     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
598     {
599       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
600     }
601     {
602       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
603       \__enumext_regex_counter_style:
604       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
605       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
606       {
607         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
608         { \exp_not:V \l__enumext_ref_key_arg_tl }
609       }
610     }
611   }

```

```

611     }
612     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
613     {
614         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
615         {
616             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
617         }
618         {
619             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
620             \__enumext_regex_counter_style:
621             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
622             \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
623             {
624                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
625                 { \exp_not:V \l__enumext_ref_key_arg_tl }
626             }
627         }
628     }
629 }

```

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.

```

630 \cs_new_protected:Nn \__enumext_starred_ref:
631 {
632     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
633     {
634         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
635         {
636             \tl_use:N \l__enumext_renew_the_count_vii_tl
637         }
638     }
639     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
640     {
641         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
642         {
643             \tl_use:N \l__enumext_renew_the_count_viii_tl
644         }
645     }
646 }

```

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

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

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

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
647 \keys_define:nn { enumext / keyans }
648 {
649     label .code:n = {
650         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
651         { \l__enumext_counter_v_tl } {#1}
652         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
653         \l__enumext_current_widest_dim
654         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
655         { \l__enumext_counter_vi_tl } {#1}
656         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
657         \l__enumext_current_widest_dim
658     },
659     label .initial:n = (\Alph*),
660     label .value_required:n = true,
661     ref .code:n = \__enumext_keyans_ref:n {#1},
662     ref .value_required:n = true,
663 }

```

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

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

```

664 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
665 {
666     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
667     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl

```

```

668     {
669         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
670     }
671     {
672         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
673         \__enumext_regex_counter_style:
674         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
675         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
676         {
677             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
678             { \exp_not:V \l__enumext_ref_key_arg_tl }
679         }
680     }
681 }

```

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

```

682 \cs_new_protected:Nn \__enumext_keyans_ref:
683 {
684     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
685     {
686         \tl_use:N \l__enumext_renew_the_count_v_tl
687     }
688 }

```

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

10.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

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

```

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

```

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

```

689 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
690 {
691     \__enumext_if_is_int:nTF { #3 }
692     {
693         \int_set:Nn #2 { #3 }
694     }
695     {
696         \regex_match:nVT { \c{Alph} | \c{alph} } { #1 }
697         { \int_set:Nn #2 { \int_from_alph:n { #3 } } }
698         \regex_match:nVT { \c{Roman} | \c{roman} } { #1 }
699         { \int_set:Nn #2 { \int_from_roman:n { #3 } } }
700     }
701 }
702 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

(End of definition for `__enumext_start_from:NNn`.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

The function `__enumext_widest_from:nNNn` used by the `widest` key take four arguments:

```

#1: The counter associated with the environment level
#2: \l__enumext_label_X_tl
#3: \l__enumext_labelwidth_X_dim
#4: <integer or string>

```

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

```

703 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
704 {
705     \__enumext_if_is_int:nTF { #4 }
706     {
707         \setcounter{enumX#1} { #4 }
708     }
709     {
710         \regex_match:nVT { \c{Alph} | \c{alph} } { #2 }
711         { \setcounter{enumX#1} { \int_from_alph:n { #4 } } }

```

```

712         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
713         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
714     }
715     \__enumext_label_width_by_box:cv
716     { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
717 }
718 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for __enumext_widest_from:nNNn.)

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

```

start
widest
\__enumext_start_X_int
719 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
720 {
721     \keys_define:nn { enumext / #1 }
722     {
723         start .code:n = {
724             \__enumext_start_from:ccn
725             { \__enumext_label_#2_tl }
726             { \__enumext_start_#2_int } {##1}
727         },
728         start .initial:n = 1,
729         widest .code:n = {
730             \__enumext_widest_from:nccn {#2}
731             { \__enumext_label_#2_tl }
732             { \__enumext_labelwidth_#2_dim } {##1}
733         },
734         widest .value_required:n = true,
735         start .value_required:n = true,
736     }
737 }
738 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for start, widest, and __enumext_start_X_int.)

10.13 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
739 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
740 {
741     \keys_define:nn { enumext / #1 }
742     {
743         topsep .skip_set:c = { \__enumext_topsep_#2_skip },
744         topsep .initial:n = {#3},
745         topsep .value_required:n = true,
746         partopsep .skip_set:c = { \__enumext_partopsep_#2_skip },
747         partopsep .initial:n = {#4},
748         partopsep .value_required:n = true,
749         parsep .skip_set:c = { \__enumext_parsep_#2_skip },
750         parsep .initial:n = {#5},
751         parsep .value_required:n = true,
752         itemsep .skip_set:c = { \__enumext_itemsep_#2_skip },
753         itemsep .initial:n = {#6},
754         itemsep .value_required:n = true,
755         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
756         noitemsep .value_forbidden:n = true,
757         nosep .meta:n = {
758             itemsep = 0pt, parsep = 0pt,
759             topsep = 0pt, partopsep = 0pt,
760         },
761         nosep .value_forbidden:n = true,
762     }
763 }

```

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

```

764 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
765 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
766 { 4.0pt plus 2.0pt minus 1.0pt }
767 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
768 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
769 { 2.0pt plus 1.0pt minus 1.0pt }

```

```

770 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
771 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
772 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
773 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
774 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
775 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
776 { 2.0pt plus 1.0pt minus 1.0pt }
777 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
778 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
779 { 4.0pt plus 2.0pt minus 1.0pt }
780 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
781 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
782 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

10.14 Setting keys for horizontal spaces

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

```

783 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
784 {
785   \keys_define:nn { enumext / #1 }
786   {
787     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
788     itemindent .value_required:n = true,
789     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
790     rightmargin .value_required:n = true,
791     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
792     listparindent .value_required:n = true,
793     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
794     list-offset .value_required:n = true,
795     list-indent .code:n =
796       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
797       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
798     list-indent .value_required:n = true,
799   }
800 }
801 \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.

```

802 \cs_set_protected:Npn \__enumext_tmp:n #1
803 {
804   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
805 }
806 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

10.14.1 Functions for setting the fake `itemindent`

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

```

807 \cs_set_protected:Nn \__enumext_fake_item:
808 {
809   \dim_compare:nNnT
810     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
811     >
812     { \c_zero_dim }
813     {
814       \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
815       {
816         \exp_not:N \mode_leave_vertical:
817         \exp_not:n { \skip_horizontal:n }
818         { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
819         \ignorespaces
820       }
821     }
822 }
823 \cs_set_protected:Nn \__enumext_keyans_fake_item:

```



```

824 {
825   \dim_compare:nNt
826   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
827   {
828     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
829     {
830       \exp_not:N \mode_leave_vertical:
831       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
832     }
833   }
834 }
835 \cs_set_protected:Nn \__enumext_fake_item_vii:
836 {
837   \dim_compare:nNt
838   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
839   {
840     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
841     {
842       \exp_not:N \mode_leave_vertical:
843       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
844     }
845   }
846 }
847 \cs_set_protected:Nn \__enumext_fake_item_viii:
848 {
849   \dim_compare:nNt
850   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
851   {
852     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
853     {
854       \exp_not:N \mode_leave_vertical:
855       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
856     }
857   }
858 }

```

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

10.15 Setting show-length key

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

```

859 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
860 {
861   \keys_define:nn { enumext / #1 }
862   {
863     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
864     show-length .initial:n = false,
865   }
866 }
867 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

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

```

before* 868 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    869 {
first    870   \keys_define:nn { enumext / #1 }
          {
871     before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
872     before .value_required:n = true,
873     before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
874     before* .value_required:n = true,
875     after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
876     after .value_required:n = true,
877     first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
878     first .value_required:n = true,
879   }
880 }

```

```

881     }
882     \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16.1 Functions for before, after and first keys in enumext

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

```

883 \cs_new_protected:Nn \__enumext_before_args_exec:
884 {
885     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
886 }

```

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*.

```

887 \cs_new_protected:Nn \__enumext_before_keys_exec:
888 {
889     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
890 }

```

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

```

891 \cs_new_protected:Nn \__enumext_after_stop_list:
892 {
893     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
894 }

```

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*.

```

895 \cs_new_protected:Nn \__enumext_after_args_exec:
896 {
897     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
898 }

```

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

10.16.2 Functions for before, after and first keys in keyans

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

```

899 \cs_new_protected:Nn \__enumext_before_args_exec_v:
900 {
901     \tl_use:N \l__enumext_before_starred_key_v_tl
902 }

```

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*.

```

903 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
904 {
905     \tl_use:N \l__enumext_before_no_starred_key_v_tl
906 }

```

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

```

907 \cs_new_protected:Nn \__enumext_after_stop_list_v:
908 {
909     \tl_use:N \l__enumext_after_stop_list_v_tl
910 }

```

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*.

```

911 \cs_new_protected:Nn \__enumext_after_args_exec_v:
912 {
913     \tl_use:N \l__enumext_after_list_args_v_tl
914 }

```

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

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

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

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

```
915 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
916 {
917   \tl_use:N \l__enumext_before_starred_key_vii_tl
918 }
919 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
920 {
921   \tl_use:N \l__enumext_before_starred_key_viii_tl
922 }
```

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⟩`.

```
923 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
924 {
925   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
926 }
927 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
928 {
929   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
930 }
```

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

```
931 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
932 {
933   \tl_use:N \l__enumext_after_stop_list_vii_tl
934 }
935 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
936 {
937   \tl_use:N \l__enumext_after_stop_list_viii_tl
938 }
```

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`.

```
939 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
940 {
941   \tl_use:N \l__enumext_after_list_args_vii_tl
942 }
943 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
944 {
945   \tl_use:N \l__enumext_after_list_args_viii_tl
946 }
```

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

10.17 Setting keys for multicol and minipage

```
mini-env
mini-sep
columns-sep
columns
```

The default value of the `columns-sep` key is handled by the state of the boolean variable `\l__enumext_columns_sep_X_bool` which is handled in the internal definition of the `enumext` and `keyans` environments.

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

```
947 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
948 {
949   \keys_define:nn { enumext / #1 }
950   {
951     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
952     mini-env .value_required:n = true,
953     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
954     mini-sep .initial:n = 0.3333em,
955     mini-sep .value_required:n = true,
956     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
957     columns-sep .value_required:n = true,
958     columns .int_set:c = { l__enumext_columns_#2_int },
959     columns .initial:n = 1,
960     columns .value_required:n = true,
961   }
```

```

962 }
963 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

For `enumext*` and `keyans*` environments the situation is a bit different, the default value for `columns` key are `2` and the command `\miniright` is not available, so we will add the keys `miniright` and `miniright*` to implement support for `minipage`.

```

964 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
965 {
966   \keys_define:nn { enumext / #1 }
967   {
968     columns      .initial:n = 2,
969     miniright    .tl_gset:c = { g__enumext_miniright_code_#2_tl },
970     miniright    .value_required:n = true,
971     miniright*   .code:n     = {
972                           \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
973                           \keys_set:nn { enumext / #1 } { miniright = {##1} }
974                           },
975     miniright*   .value_required:n = true,
976   }
977 }
978 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for multicols

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

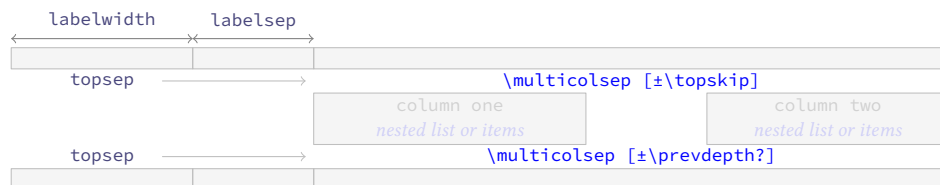


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

10.18.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

979 \cs_new_protected:Nn \__enumext_multi_set_vskip:
980 {
981   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
982   {
983     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
984   }
985   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
986   {
987     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
988   }
989   \__enumext_add_pre_parsep:
990 }

```

(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*.

```

991 \cs_new_protected:Nn \__enumext_add_pre_parsep:
992 {
993   \int_case:nn { \l__enumext_level_int }
994   {
995     { 2 }{
996       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
997       {
998         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
999       }
1000     }
1001     { 3 }{
1002       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1003       {
1004         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1005       }
1006     }
1007     { 4 }{
1008       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1009       {
1010         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1011       }
1012     }
1013   }
1014 }

```

(End of definition for `__enumext_add_pre_parsep:`.)

`__enumext_multi_addvspace:` The function `__enumext_multi_addvspace:` will apply the spaces set using `\addvspace` “*above*” the `multicols` environment in `enumext`, taking into account whether \TeX is in $\langle\textit{horizontal mode}\rangle$ or $\langle\textit{vertical mode}\rangle$.

```

1015 \cs_new_protected:Nn \__enumext_multi_addvspace:
1016 {
1017   \__enumext_multi_set_vskip:
1018   \mode_if_vertical:T
1019   {
1020     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1021     {
1022       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1023     }
1024     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1025     {
1026       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1027     }
1028   }
1029   \par\nopagebreak
1030   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1031 }

```

(End of definition for `__enumext_multi_addvspace:`.)

10.18.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1032 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1033 {
1034   \skip_set:Nn \l__enumext_multicols_above_v_skip
1035   {
1036     \l__enumext_topsep_v_skip
1037   }
1038   \skip_set:Nn \l__enumext_multicols_below_v_skip
1039   {
1040     \l__enumext_topsep_v_skip
1041   }
1042 }
1043 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1044 {

```

```

1045 \__enumext_keyans_multi_set_vskip:
1046 \mode_if_vertical:T
1047 {
1048   \skip_add:Nn \l__enumext_multicols_above_v_skip
1049   {
1050     \skip_use:N \l__enumext_partopsep_v_skip
1051   }
1052   \skip_add:Nn \l__enumext_multicols_below_v_skip
1053   {
1054     \skip_use:N \l__enumext_partopsep_v_skip
1055   }
1056 }
1057 \par\nopagebreak
1058 \addvspace{ \l__enumext_multicols_above_v_skip }
1059 }

```

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

10.19 Adjustment of vertical spaces for minipage

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

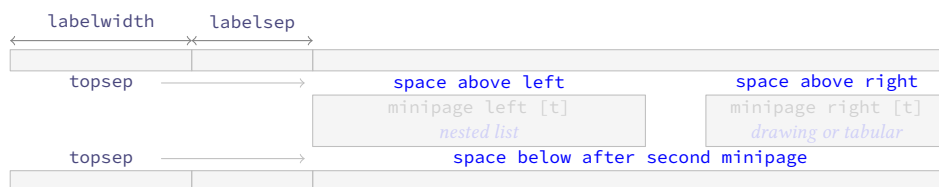


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

Here there are several complications that we must circumvent, the `minipage` environment eliminates the “top” spaces, the `multicols` environment can be nested in the `minipage` environment, the “top” and “bottom” spaces are affected when `topsep=0pt` and to this is added the `\partopsep` parameter that comes into action according to whether \TeX is in *horizontal mode* or *vertical mode*. Depending on these cases, small adjustments must be made using `\vspace` and `\addvspace` to obtain the “desired vertical spacing”.

Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (`glue`) produced by the `minipage` environment is quite complicated, even more if `multicols` it is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug`[12] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

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

```

1060 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
1061 {
1062   \__enumext_minipage:w [ t ] { #1 }
1063   \legacy_if_gset_false:n { @minipage }
1064   \vspace { 0pt }
1065 }
1066 { \__enumext_endminipage: }

```

(End of definition for `__enumext_mini_env*`.)

10.19.1 Adjustment of vertical spaces for minipage in enumext

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

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

First determine if the `multicols` environment is active by comparing the value of the `\l__enumext_columns_X_int` variable handled by the `columns` key, according to this comparison we set the adjusted values for `\l__enumext_minipage_left_skip`, `\l__enumext_minipage_right_skip` and `\l__enumext_minipage_after_skip`.

```

1067 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1068 {

```

```

1069 \int_compare:nNnTF
1070 { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1071 {

```

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

```

1072 \skip_if_eq:nnTF
1073 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1074 {
1075   \skip_set:Nn \l__enumext_minipage_left_skip
1076   {
1077     -0.150\box_dp:N \strutbox
1078   }
1079   \skip_set:Nn \l__enumext_minipage_right_skip
1080   {
1081     0.695\box_dp:N \strutbox
1082   }
1083   \skip_set:Nn \l__enumext_minipage_after_skip
1084   {
1085     \box_dp:N \strutbox
1086   }
1087   \__enumext_zero_parsep:
1088 }
1089 {
1090   \skip_set:Nn \l__enumext_minipage_left_skip
1091   {
1092     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1093   }
1094   \skip_set:Nn \l__enumext_minipage_right_skip
1095   {
1096     0.695\box_dp:N \strutbox
1097   }
1098   \skip_set:Nn \l__enumext_minipage_after_skip
1099   {
1100     1.85\box_dp:N \strutbox
1101     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1102   }
1103 }
1104 }
1105 {

```

If only `\enumext` environment is nested in `__enumext_mini_env*` environment, we will apply a correction factor to the *vertical spaces* taking into account the value of `\topsep`, if this is zero we will use `\strutbox` as the basis for the calculations.

```

1106 \skip_if_eq:nnTF
1107 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1108 {
1109   \skip_set:Nn \l__enumext_minipage_left_skip
1110   {
1111     0.5\box_dp:N \strutbox
1112     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1113   }
1114   \skip_set:Nn \l__enumext_minipage_right_skip
1115   {
1116     \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1117   }
1118   \skip_set:Nn \l__enumext_minipage_after_skip
1119   {
1120     1.6\box_dp:N \strutbox
1121   }
1122 }
1123 {
1124   \skip_set:Nn \l__enumext_minipage_left_skip
1125   {
1126     0.5875\box_dp:N \strutbox
1127     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1128   }
1129   \skip_set:Nn \l__enumext_minipage_right_skip
1130   {
1131     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1132     + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }

```



```

1133         }
1134         \skip_set:Nn \l__enumext_minipage_after_skip
1135         {
1136             0.325\box_dp:N \strutbox
1137             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1138         }
1139     }
1140 }
1141 }

```

(End of definition for \l__enumext_mini_set_vskip:.)

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

```

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

```

(End of definition for \l__enumext_zero_parsep:.)

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

```

1166 \cs_new_protected:Nn \l__enumext_mini_addvspace:
1167 {
1168     \l__enumext_mini_set_vskip:
1169     \mode_if_vertical:T
1170     {
1171         \skip_add:Nn \l__enumext_minipage_left_skip
1172         {
1173             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1174         }
1175         \skip_add:Nn \l__enumext_minipage_after_skip
1176         {
1177             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1178         }
1179     }
1180     \par\nopagebreak
1181     \addvspace { \l__enumext_minipage_left_skip }
1182 }

```

(End of definition for \l__enumext_mini_addvspace:.)

10.19.2 Adjustment of vertical spaces for minipage in keyans

`__enumext_keyans_mini_set_vskip:`

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

```

1183 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1184 {
1185   \skip_zero_new:N \__enumext_minipage_after_skip
1186   \skip_zero_new:N \__enumext_minipage_left_skip
1187   \skip_zero_new:N \__enumext_minipage_right_skip
1188   \int_compare:nNnTF { \__enumext_columns_v_int } > { 1 }
1189   {
1190     \skip_if_eq:nnTF { \__enumext_topsep_v_skip } { \c_zero_skip }
1191     {
1192       \skip_set:Nn \__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1193       \skip_set:Nn \__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1194       \skip_set:Nn \__enumext_minipage_after_skip { \box_dp:N \strutbox }
1195       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
1196       {
1197         \skip_add:Nn \__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1198       }
1199     }
1200     {
1201       \skip_set:Nn \__enumext_minipage_left_skip
1202       {
1203         \skip_use:N \__enumext_topsep_v_skip
1204       }
1205       \skip_set:Nn \__enumext_minipage_right_skip
1206       {
1207         0.705\box_dp:N \strutbox
1208       }
1209       \skip_set:Nn \__enumext_minipage_after_skip
1210       {
1211         1.85\box_dp:N \strutbox + \__enumext_topsep_v_skip
1212       }
1213     }
1214   }
1215   {
1216     \skip_if_eq:nnTF { \__enumext_topsep_v_skip } { \c_zero_skip }
1217     {
1218       \skip_set:Nn \__enumext_minipage_left_skip
1219       {
1220         0.5\box_dp:N \strutbox
1221         + \__enumext_partopsep_v_skip
1222       }
1223       \skip_set:Nn \__enumext_minipage_right_skip
1224       {
1225         \__enumext_partopsep_v_skip
1226       }
1227       \skip_set:Nn \__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1228     }
1229     {
1230       \skip_set:Nn \__enumext_minipage_left_skip
1231       {
1232         0.5875\box_dp:N \strutbox - \__enumext_partopsep_v_skip
1233       }
1234       \skip_set:Nn \__enumext_minipage_right_skip
1235       {
1236         \__enumext_topsep_v_skip + \__enumext_partopsep_v_skip
1237       }
1238       \skip_set:Nn \__enumext_minipage_after_skip
1239       {
1240         0.325\box_dp:N \strutbox + \__enumext_topsep_v_skip
1241       }
1242     }
1243   }
1244 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

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

```

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

```

(End of definition for `__enumext_keyans_mini_addvspace:.`)

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1262 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1263 {
1264   \skip_zero_new:N \l__enumext_minipage_left_skip
1265   \skip_gzero_new:N \g__enumext_minipage_right_skip
1266   \skip_gzero_new:N \g__enumext_minipage_after_skip
1267   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1268   {
1269     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1270     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1271   }
1272   {
1273     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1274     \skip_gset:Nn \g__enumext_minipage_right_skip
1275     {
1276       \l__enumext_topsep_vii_skip
1277     }
1278     \skip_gset:Nn \g__enumext_minipage_after_skip
1279     {
1280       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1281     }
1282   }
1283 }
1284 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1285 {
1286   \skip_zero_new:N \l__enumext_minipage_after_skip
1287   \skip_zero_new:N \l__enumext_minipage_left_skip
1288   \skip_zero_new:N \l__enumext_minipage_right_skip
1289   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1290   {
1291     \skip_set:Nn \l__enumext_minipage_left_skip
1292     {
1293       0.5\box_dp:N \strutbox
1294     }
1295     \skip_set:Nn \l__enumext_minipage_right_skip
1296     {
1297       \l__enumext_partopsep_viii_skip
1298     }
1299     \skip_set:Nn \l__enumext_minipage_after_skip
1300     {
1301       1.6\box_dp:N \strutbox
1302     }
1303   }
1304 }

```

```

1305     \skip_set:Nn \l__enumext_minipage_left_skip
1306     {
1307         0.5875\box_dp:N \strutbox
1308     }
1309     \skip_set:Nn \l__enumext_minipage_right_skip
1310     {
1311         \l__enumext_topsep_viii_skip
1312     }
1313     \skip_set:Nn \l__enumext_minipage_after_skip
1314     {
1315         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1316     }
1317 }
1318 }

```

(End of definition for `\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 `\miniright` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether \TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1319 \cs_new_protected:Nn \l__enumext_mini_addvspace_vii:
1320 {
1321     \l__enumext_mini_set_vskip_vii:
1322     \par\nopagebreak
1323     \addvspace { \l__enumext_minipage_left_skip }
1324 }
1325 \cs_new_protected:Nn \l__enumext_mini_addvspace_viii:
1326 {
1327     \l__enumext_mini_set_vskip_viii:
1328     \par\nopagebreak
1329     \addvspace { \l__enumext_minipage_left_skip }
1330 }

```

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

10.19.4 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 version* ‘`*`’ 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.

```

1331 \NewDocumentCommand \miniright { s }
1332 {
1333     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1334     {
1335         \msg_error:nnn { enumext } { wrong-miniright-place }
1336     }
1337     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1338     {
1339         \msg_error:nnn { enumext } { wrong-miniright-place }
1340     }
1341     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1342     {
1343         \l__enumext_keyans_mini_right_cmd:n {#1}
1344     }
1345     { \l__enumext_mini_right_cmd:n {#1} }
1346 }

```

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

`\l__enumext_mini_right_cmd:n`

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

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

```

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

```

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

`__enumext_keyans_mini_right_cmd:n`

The function `__enumext_keyans_mini_right_cmd:n` takes as argument the *starred version* ‘***’ 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.

```

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

```

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

10.20 Setting above and below keys

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

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

```

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

```

```

1397     }
1398 }
1399 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

10.20.1 Functions for above and below keys in enumext

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

```

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

```

(End of definition for `__enumext_vspace_above:`.)

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

```

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

```

(End of definition for `__enumext_vspace_below:`.)

10.20.2 Functions for above and below keys in keyans

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

```

1428 \cs_new_protected:Nn \__enumext_vspace_above_v:
1429 {
1430     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1431     {
1432         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1433         {
1434             \vspace*{ \l__enumext_vspace_above_v_skip }
1435         }
1436         { \vspace { \l__enumext_vspace_above_v_skip } }
1437     }
1438 }

```

(End of definition for `__enumext_vspace_above_v:`.)

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

```

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

```

```

1446     }
1447     { \vspace { \l__enumext_vspace_below_v_skip } }
1448   }
1449 }

```

(End of definition for `__enumext_vspace_below_v:`)

10.20.3 Functions for above and below keys in enumext* keyans*

The functions `__enumext_vspace_above_vii:` and `__enumext_vspace_above_viii:` apply the *vertical space above* the `enumext*` and `keyans*` environments set by the `above` and `above*` keys.

```

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

```

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

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.

```

1472 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1473 {
1474   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1475   {
1476     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1477     {
1478       \vspace*{ \l__enumext_vspace_below_vii_skip }
1479     }
1480     { \vspace { \l__enumext_vspace_below_vii_skip } }
1481   }
1482 }
1483 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1484 {
1485   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1486   {
1487     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1488     {
1489       \vspace*{ \l__enumext_vspace_below_viii_skip }
1490     }
1491     { \vspace { \l__enumext_vspace_below_viii_skip } }
1492   }
1493 }

```

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

10.21 Setting series, resume and resume* keys

The `series` key is responsible for the whole process of the `resume` and `resume*` keys. The idea behind this is to be able to absorb the $\langle keys \rangle$ passed to the optional argument of the “*first level*” of the environments `enumext` and `enumext*`, but, discarding some specific $\langle keys \rangle$.

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

```

series
resume
resume*
1494 \cs_set_protected:Npn \__enumext_tmp:n #1
1495 {
1496   \keys_define:nn { enumext / #1 }

```



```

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

```

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

10.21.1 Internal functions for series key

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\__enumext_parse_series:n
  \__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§10.32) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§10.35).

```

1536 \cs_new_protected:Npn \__enumext_parse_series:n #1
1537 {
1538     \str_if_empty:NTF \l__enumext_series_str
1539     {
1540         \bool_if:NF \l__enumext_resume_active_bool
1541         {

```

```

1542         \__enumext_resume_last:n {#1}
1543     }
1544 }
1545 {
1546     \tl_gclear_new:c { g__enumext_series_ \__enumext_series_str_tl }
1547     \tl_gset:ce { g__enumext_series_ \__enumext_series_str_tl }
1548         { \__enumext_filter_series:n {#1} }
1549     \int_if_exist:cF { g__enumext_series_ \__enumext_series_str_int }
1550     {
1551         \int_new:c { g__enumext_series_ \__enumext_series_str_int }
1552     }
1553 }
1554 }

```

The function `__enumext_resume_last:n` will be in charge of saving the filtering *⟨keys⟩* when the `series` key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the `enumext` environment and in the variable `\g__enumext_starred_series_tl` for the `enumext*` environment. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the `series` key is not being used.

```

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

```

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

10.21.2 Internal function to save counter value

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same *⟨series name⟩* but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§10.32) and the `enumext*` environment definition (§10.35).

```

1568 \cs_new_protected:Nn \__enumext_resume_save_counter:
1569 {
1570     \bool_if:NT \g__enumext_standar_bool
1571     {
1572         \tl_if_empty:NF \l__enumext_series_str
1573         {
1574             \int_gset_eq:cN
1575                 { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1576         }
1577         \tl_if_empty:NTF \l__enumext_resume_name_tl
1578         {
1579             \str_if_empty:NT \l__enumext_series_str
1580             {
1581                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1582             }
1583         }
1584         {
1585             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1586             {
1587                 \int_gset_eq:cN
1588                     { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1589             }
1590         }
1591         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }

```

```

1592     {
1593         \int_gset_eq:cN
1594         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1595     }
1596 }
1597 \bool_if:NT \g__enumext_starred_bool
1598 {
1599     \tl_if_empty:NF \l__enumext_series_str
1600     {
1601         \int_gset_eq:cN
1602         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1603     }
1604     \tl_if_empty:NTF \l__enumext_resume_name_tl
1605     {
1606         \str_if_empty:NT \l__enumext_series_str
1607         {
1608             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1609         }
1610     }
1611     {
1612         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1613         {
1614             \int_gset_eq:cN
1615             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1616         }
1617     }
1618     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1619     {
1620         \int_gset_eq:cN
1621         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1622     }
1623 }
1624 }

```

(End of definition for __enumext_resume_save_counter:.)

10.21.3 Internal functions for resume key

__enumext_resume_series:n

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

```

1625 \cs_new_protected:Npn \__enumext_resume_series:n #1
1626 {
1627     \tl_if_empty:NTF {#1}
1628     {
1629         \__enumext_resume_counter:n { }
1630     }
1631     {
1632         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1633         {
1634             \__enumext_resume_counter:n {#1}
1635             \bool_if:NT \g__enumext_standar_bool
1636             {
1637                 \keys_set:nv { enumext / level-1 }
1638                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1639             }
1640             \bool_if:NT \g__enumext_starred_bool
1641             {
1642                 \keys_set:nv { enumext / enumext* }
1643                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1644             }
1645         }
1646         {
1647             \bool_if:NT \g__enumext_standar_bool
1648             {
1649                 \msg_error:nnn { enumext } { unknown-series } {#1}
1650             }
1651             \bool_if:NT \g__enumext_starred_bool

```

```

1652         {
1653             \msg_error:nnn { enumext } { unknown-series } {#1}
1654         }
1655     }
1656 }
1657 }

```

(End of definition for __enumext_resume_series:n.)

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 $\langle \textit{series name} \rangle$. 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 \textit{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`.

```

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

```

The __enumext_resume_counter: function is executed when the `resume` key is used *without value*, only the counters for the “*first level*” of the environments will be set.

```

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

```

The function __enumext_resume_counter_series: will be executed when the `resume={\langle \textit{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.

```

1684 \cs_new_protected:Nn \__enumext_resume_counter_series:
1685 {
1686     \bool_if:NT \g__enumext_standar_bool
1687     {
1688         \int_set:Nn \l__enumext_start_i_int
1689         {
1690             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1691         }
1692     }
1693     \bool_if:NT \g__enumext_starred_bool
1694     {
1695         \int_set:Nn \l__enumext_start_vii_int
1696         {
1697             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1698         }
1699     }
1700 }

```

The function __enumext_resume_counter_save_ans: will be executed when the `save-ans` key is active along with the `resume` key, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the `save-ans` key.

```

1701 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:

```

```

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

```

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

10.21.4 Internal function for resume* key

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

```

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

```

(End of definition for `__enumext_resume_starred:`)

10.22 Setting save-ans key

The key `save-ans` is directly associated with the keys `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

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

```

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

```

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

10.22.1 Internal functions for save-ans key

__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 we proceed to execute the function __enumext_storing_exec: for `enumext` and `enumext*` environments.

```

1750 \cs_new_protected:Npn \__enumext_storing_set:n #1
1751 {
1752   \tl_set:Nx \l__enumext_store_name_tl {#1}
1753   \tl_if_empty:NTF \l__enumext_store_name_tl
1754   {
1755     \bool_lazy_or:nnT
1756     { \l__enumext_standar_first_level_bool }
1757     { \l__enumext_starred_first_level_bool }
1758     {
1759       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1760     }
1761   }
1762   {
1763     \bool_lazy_or:nnT
1764     { \l__enumext_standar_first_level_bool }
1765     { \l__enumext_starred_first_level_bool }
1766     {
1767       \msg_term:nnVV
1768       { enumext } { save-ans-log } \g__enumext_envir_name_tl \l__enumext_store_name_tl
1769       \__enumext_storing_exec:
1770     }
1771   }
1772 }

```

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_store_ans_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*`.

```

1773 \cs_new_protected:Nn \__enumext_storing_exec:
1774 {
1775   \bool_set_true:N \l__enumext_store_active_bool
1776   \bool_set_true:N \l__enumext_store_ans_bool
1777   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1778   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1779   {
1780     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1781     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1782   }
1783   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1784   {
1785     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1786     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1787   }
1788   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1789   {
1790     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1791     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1792   }
1793 }

```

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

10.23 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.

10.23.1 Setting check-ans key

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

check-ans

no-store

```

1794 \cs_set_protected:Npn \__enumext_tmp:n #1
1795 {
1796   \keys_define:nn { enumext / #1 }
1797   {
1798     check-ans .bool_set:N = \__enumext_check_ans_bool,
1799     check-ans .initial:n = false,
1800     check-ans .value_required:n = true,
1801     no-store .code:n = {
1802       \bool_set_false:N \__enumext_store_ans_bool
1803       \bool_set_false:N \__enumext_check_ans_bool
1804     },
1805     no-store .value_forbidden:n = true,
1806   }
1807 }
1808 \clist_map_inline:nn
1809 {
1810   level-1, level-2, level-3, level-4, enumext*
1811 }
1812 { \__enumext_tmp:n {#1} }
```

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

10.23.2 Set-up check answer mechanism

__enumext_check_ans:
__enumext_check_ans_level:

The function `__enumext_check_ans:` will check the state of the variable `\l__enumext_check_ans_bool` activated by the key `check-ans`, if this is “true” it will check the variable `\l__enumext_store_name_tl` is not empty, that is, the key `save-ans` is activated, if so it will execute the function `__enumext_check_ans_level:` and otherwise it will return an appropriate error message.

```

1813 \cs_new_protected:Nn \__enumext_check_ans:
1814 {
1815   \bool_if:NT \l__enumext_check_ans_bool
1816   {
1817     \tl_if_empty:NTF \l__enumext_store_name_tl
1818     {
1819       \bool_if:NT \l__enumext_standar_first_level_bool
1820       {
1821         \msg_error:nnnn { enumext } { need-save-ans }{ check-ans } { enumext }
1822       }
1823       \bool_if:NT \l__enumext_starred_first_level_bool
1824       {
1825         \msg_error:nnnn { enumext } { need-save-ans }{ check-ans } { enumext* }
1826       }
1827     }
1828     {
1829       \__enumext_check_ans_level:
1830     }
1831   }
```



```
1832 }
```

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.

```
1833 \cs_new_protected:Nn \__enumext_check_ans_level:
1834 {
1835   \int_case:nn { \l__enumext_level_int }
1836   {
1837     { 1 }{
1838       \bool_lazy_all:nT
1839       {
1840         { \bool_if_p:N \g__enumext_starred_bool }
1841         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1842       }
1843       {
1844         \int_gdecr:N \g__enumext_item_number_int
1845       }
1846     }
1847     { 2 }{
1848       \int_gdecr:N \g__enumext_item_number_int
1849     }
1850     { 3 }{
1851       \int_gdecr:N \g__enumext_item_number_int
1852     }
1853     { 4 }{
1854       \int_gdecr:N \g__enumext_item_number_int
1855     }
1856   }
1857   \int_case:nn { \l__enumext_level_h_int }
1858   {
1859     { 1 }{
1860       \bool_if:NT \g__enumext_standar_bool
1861       {
1862         \int_gdecr:N \g__enumext_item_number_int
1863       }
1864     }
1865   }
1866 }
```

(End of definition for `__enumext_check_ans:` and `__enumext_check_ans_level:`)

```
\__enumext_check_ans_to_hook:
```

The function `__enumext_check_ans_to_hook:` will *export* the status of the local variable `\l__enumext_check_ans_bool` to the global variable `\g__enumext_check_ans_bool` only if the key `check-ans` is active.

```
1867 \cs_new_protected:Nn \__enumext_check_ans_to_hook:
1868 {
1869   \bool_lazy_and:nnT
1870   { \bool_if_p:N \l__enumext_check_ans_bool }
1871   { \bool_if_p:N \g__enumext_standar_bool }
1872   {
1873     \bool_gset_true:N \g__enumext_check_ans_bool
1874   }
1875   \bool_lazy_and:nnT
1876   { \bool_if_p:N \l__enumext_check_ans_bool }
1877   { \bool_if_p:N \g__enumext_starred_bool }
1878   {
1879     \bool_gset_true:N \g__enumext_check_ans_bool
1880   }
1881 }
```

(End of definition for `__enumext_check_ans_to_hook:`)

```
\__enumext_check_ans_show:
```

The function `__enumext_check_ans_show:` will perform the comparison between the `\item` in the environments and the `\item`’s with answers and return the appropriate message. As this function is passed to the function `__enumext_after_env:nn` for the environments `enumext` and `enumext*` we must make sure that we are not nested at any level and finally reset our global variables.

```
1882 \cs_new_protected:Nn \__enumext_check_ans_show:
1883 {
1884   \int_case:nn { \g__enumext_item_answer_diff_int }
```

```

1885     {
1886       { -1 }{ \__enumext_check_ans_msg_less: }
1887       { 0 }{ \__enumext_check_ans_msg_same_ok: }
1888       { 1 }{ \__enumext_check_ans_msg_greater: }
1889     }
1890   }
1891   \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1892   {
1893     \msg_warning:nneee { enumext } { item-less-answer }
1894     { \g__enumext_store_name_tl } { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1895   }
1896   \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1897   {
1898     \msg_term:nneee { enumext } { items-same-answer }
1899     { \g__enumext_store_name_tl } { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1900   }
1901   \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1902   {
1903     \msg_warning:nneee { enumext } { item-greater-answer }
1904     { \g__enumext_store_name_tl } { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1905   }

```

(End of definition for __enumext_check_ans_show:.)

10.23.3 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.

```

1906 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1907 {
1908   \int_compare:nNnT
1909   { \g__enumext_check_starred_cmd_int } = { 0 }
1910   {
1911     \msg_warning:nnnV
1912     { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1913   }
1914   \int_compare:nNnT
1915   { \g__enumext_check_starred_cmd_int } > { 1 }
1916   {
1917     \msg_warning:nnnV
1918     { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1919   }
1920   \int_gzero:N \g__enumext_check_starred_cmd_int
1921   \tl_clear:N \l__enumext_check_start_line_env_tl
1922 }

```

(End of definition for __enumext_check_starred_cmd:n.)

10.24 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*`.

```

1923 \cs_set_protected:Npn \__enumext_tmp:n #1
1924 {
1925   \keys_define:nn { enumext / #1 }
1926   {
1927     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1928     wrap-ans .initial:n = \fbox{##1},
1929     wrap-ans .value_required:n = true,
1930     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1931     wrap-opt .initial:n = [{##1}],
1932     wrap-opt .value_required:n = true,
1933     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1934     save-sep .initial:n = {, ~ },
1935     save-sep .value_required:n = true,
1936     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1937     mark-ans .initial:n = \textasteriskcentered,
1938     mark-ans .value_required:n = true,
1939     mark-pos .choice:,
1940     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },

```

```

1941     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1942     mark-pos .initial:n = right,
1943     mark-pos .value_required:n = true,
1944     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1945     show-ans .initial:n = false,
1946     show-ans .value_required:n = true,
1947     show-pos .bool_set:N = \l__enumext_show_position_bool,
1948     show-pos .initial:n = false,
1949     show-pos .value_required:n = true,
1950     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
1951     mark-ref .initial:n = \textasteriskcentered,
1952     mark-ref .value_required:n = true,
1953     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1954     save-ref .initial:n = false,
1955     save-ref .value_required:n = true,
1956   }
1957 }
1958 \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`.

```

1959 \cs_set_protected:Npn \__enumext_tmp:n #1
1960 {
1961   \keys_define:nn { enumext / #1 }
1962   {
1963     mark-pos .choice:,
1964     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1965     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1966     mark-pos .initial:n = right,
1967     mark-pos .value_required:n = true,
1968     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1969     show-ans .initial:n = false,
1970     show-ans .value_required:n = true,
1971     show-pos .bool_set:N = \l__enumext_show_position_bool,
1972     show-pos .initial:n = false,
1973     show-pos .value_required:n = true,
1974   }
1975 }
1976 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

For the `enumext` and `enumext*` environments we will only add the keys `columns*` and `columns-sep*`. The values set by these keys will be passed as optional arguments to the “inner levels” of the `enumext` and `enumext*` environments via the `__enumext_store_level_open:` function used by the “storage system” to preserve the structure and then used by the `\printkeyans` command.

```

1977 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1978 {
1979   \keys_define:nn { enumext / #1 }
1980   {
1981     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1982               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1983               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1984               {
1985                 columns = \exp_not:v { l__enumext_store_columns_#2_int },
1986               },
1987     columns* .value_required:n = true,
1988     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1989               \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1990               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1991               {
1992                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1993               },
1994     columns-sep* .value_required:n = true,
1995   }
1996 }
1997 \clist_map_inline:nn
1998 {
1999   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}

```

```

2000   }
2001   { \__enumext_tmp:n #1 }

```

(End of definition for `columns*` and `columns-sep*`.)

10.24.1 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

2002 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2003 {
2004   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2005   {
2006     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2007   }
2008   { #1 }
2009 }
2010 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

10.24.2 Function for storing content in sequence

```

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

```

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

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

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

```

2011 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2012 {
2013   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2014 }
2015 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

10.24.3 Functions for storing the list structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

The memorization structure of the list is handled by the functions `__enumext_store_level_open:` and `__enumext_store_level_close:` which are executed per level within the `enumext` environment. As this structure will be stored in the sequence set by the `save-ans` key, we will not be able to modify it locally, so it is better to take only two copies of the values set by the `columns` and `columns-sep` keys if they are present when changing levels within the `enumext` environment when executing `\anskey`. We will store these values in the variable `__enumext_store_columns_X_tl` if they are different from `0` and `0pt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```

2016 \cs_new_protected:Nn \__enumext_store_level_open:
2017 {
2018   \bool_if:NT \__enumext_store_ans_bool
2019   {
2020     \tl_if_empty:cTF { \__enumext_store_opt_ \__enumext_level: _tl }
2021     {
2022       \__enumext_store_addto_seq:n
2023       {
2024         \item \begin{enumext}
2025       }
2026     }
2027     {
2028       \tl_put_left:cn { \__enumext_store_opt_ \__enumext_level: _tl }
2029       {
2030         \item \begin{enumext} [
2031         ]
2032       }
2033       \tl_put_right:cn { \__enumext_store_opt_ \__enumext_level: _tl }
2034       {
2035         ]
2036       }
2037       \__enumext_store_addto_seq:v { \__enumext_store_opt_ \__enumext_level: _tl }
2038     }
2039   }

```

```

2040 \cs_new_protected:Nn \__enumext_store_level_close:
2041 {
2042   \bool_if:NT \l__enumext_store_ans_bool
2043   {
2044     \__enumext_store_addto_seq:n { \end{enumext} }
2045   }
2046 }

```

(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.

```

2047 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2048 {
2049   \bool_if:NT \l__enumext_store_ans_bool
2050   {
2051     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
2052     {
2053       \__enumext_store_addto_seq:n
2054       {
2055         \item \mode_leave_vertical:
2056         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2057         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2058       }
2059     }
2060     {
2061       \tl_put_left:Nn \l__enumext_store_opt_vii_tl
2062       {
2063         \item \mode_leave_vertical:
2064         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2065         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2066       ]
2067       \tl_put_right:Nn \l__enumext_store_opt_vii_tl
2068       {
2069         ]
2070       }
2071       \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
2072     }
2073   }
2074 }
2075 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2076 {
2077   \bool_if:NT \l__enumext_store_ans_bool
2078   {
2079     \__enumext_store_addto_seq:n { \end{enumext*} }
2080   }
2081 }

```

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

10.24.4 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`

```

2082 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2083 {
2084   \mode_leave_vertical:
2085   \skip_horizontal:n { -\dim_use:N #2 }
2086   \makebox[0pt][ r ]
2087   {
2088     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2089     {
2090       \tl_use:N \l__enumext_mark_answer_sym_tl
2091     }
2092   }

```

```

2093     \skip_horizontal:n { \dim_use:N #2 }
2094   }
2095   \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

10.25 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 very similar behaviour to that of \item in the enumext and enumext* environments.

\anskey We want the command to be executed as follows: `\anskey<(number)>*[<key = val>]{<content>}` so first we’ll add the keys `item-sym*`, `item-pos*` and `store-brk`.

```

2096 \keys_define:nn { enumext / anskey }
2097 {
2098   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2099   item-sym* .value_required:n = true,
2100   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2101   item-pos* .value_required:n = true,
2102   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2103   store-brk .default:n = true,
2104   store-brk .value_forbidden:n = true,
2105 }

```

This command \anskey will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error. If the `check-ans` key is active, increment `\g__enumext_count_item_with_ans_int`, then call internal function `__enumext_store_anskey_code:nnnn` will “*store content*” in the *(sequence)* and in the *(prop list)*.

```

2106 \NewDocumentCommand \anskey { d() s o +m }
2107 {
2108   \bool_if:NF \l__enumext_store_active_bool
2109   {
2110     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2111   }
2112   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2113   {
2114     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2115   }
2116   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2117   {
2118     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2119   }
2120   \group_begin:
2121     \bool_if:NT \l__enumext_store_ans_bool
2122     {
2123       \bool_if:NT \l__enumext_check_ans_bool
2124       {
2125         \int_gincr:N \g__enumext_item_anskey_int
2126       }
2127       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2128     }
2129   \group_end:
2130 }

```

(End of definition for \anskey. This function is documented on page 11.)

__enumext_store_anskey_code:nnnn

The internal function `__enumext_store_anskey_code:nnnn` first we pass the command *(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.

```

2131 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2132 {
2133   \__enumext_store_addto_prop:n {#4}
2134   \bool_if:NT \l__enumext_store_ref_key_bool
2135   {
2136     \__enumext_store_internal_ref:
2137   }
2138   \__enumext_store_anskey_show_left:n { #4 }

```

Now we start processing the optional arguments 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 `[(key = val)]`, if the `store-brk` key is present and the command is running under `enumext` (not in the starred version) we will add `\columnbreak` and then `\item`.

```

2139 \tl_clear:N \l__enumext_store_anskey_arg_tl
2140 \tl_if_novalue:nF {#3}
2141 {
2142   \keys_set:nn { enumext / anskey } {#3}
2143 }
2144 \bool_lazy_and:nnT
2145 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2146 { \bool_not_p:n { \l__enumext_starred_bool } }
2147 {
2148   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2149 }
2150 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

Now we will check the *(⟨number⟩)* argument and add it to `\l__enumext_store_anskey_arg_tl` if the command is running under `enumext*` (starred version).

```

2151 \tl_if_novalue:nF {#1}
2152 {
2153   \int_set:Nn \l__enumext_store_columns_join_int {#1}
2154   \bool_if:NT \l__enumext_starred_bool
2155   {
2156     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2157     {
2158       ( \exp_not:V \l__enumext_store_columns_join_int )
2159     }
2160   }
2161 }

```

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

```

2162 \bool_if:nTF {#2}
2163 {
2164   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2165   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2166   {
2167     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2168     {
2169       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2170     }
2171   }
2172   \dim_compare:nT
2173   {
2174     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2175   }
2176   {
2177     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2178     {
2179       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2180     }
2181   }
2182   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2183 }
2184 {
2185   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2186 }

```

Finally we check 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 in *(sequence)*.

```

2187 \bool_lazy_and:nnT
2188 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2189 { \bool_if_p:N \l__enumext_hyperref_bool }
2190 {
2191   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2192   {
2193     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2194     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2195   }

```



```

2196     }
2197     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2198 }

```

(End of definition for __enumext_store_anskey_code:nnnn.)

__enumext_store_internal_ref:

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

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

```

2199 \cs_new_protected:Nn \__enumext_store_internal_ref:
2200 {
2201     \cs_set_protected:Npn \__enumext_tmp:n ##1
2202     {
2203         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2204         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2205         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2206         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2207     }
2208     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2209     \cs_set:Npn \__enumext_tmp:n ##1
2210     { . \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

Here we need to analyse the cases where the environment is started with enumext* and if \anskey is running alone in it or if it is running in a nested enumext environment within the starting environment.

```

2211     \bool_lazy_all:nT
2212     {
2213         { \bool_if_p:N \g__enumext_starred_bool }
2214         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2215     }
2216     {
2217         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2218         { \tl_use:N \l__enumext_label_copy_vii_tl }
2219     }
2220     \bool_lazy_all:nT
2221     {
2222         { \bool_if_p:N \l__enumext_standar_bool }
2223         { \bool_if_p:N \g__enumext_starred_bool }
2224         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2225     }
2226     {
2227         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2228         {
2229             \tl_use:N \l__enumext_label_copy_vii_tl
2230             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2231         }
2232     }

```

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.

```

2233     \bool_lazy_all:nT
2234     {
2235         { \bool_if_p:N \l__enumext_standar_bool }
2236         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2237         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2238         { \bool_not_p:n { \l__enumext_starred_bool } }
2239     }
2240     {
2241         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2242         {
2243             \tl_use:N \l__enumext_label_copy_i_tl
2244             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2245         }
2246     }
2247     \cs_set:Npn \__enumext_tmp:n ##1
2248     { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2249     \bool_lazy_all:nT
2250     {
2251         { \bool_if_p:N \l__enumext_standar_bool }
2252         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }

```

```

2253     { \bool_not_p:n { \g__enumext_starred_bool } }
2254     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2255   }
2256   {
2257     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2258     {
2259       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2260       . \tl_use:N \l__enumext_label_copy_vii_tl
2261     }
2262   }

```

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

```

2263     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2264     {
2265       \l__enumext_store_name_tl \c_colon_str
2266       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2267     }

```

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.

```

2268     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2269     {
2270       \__enumext_newlabel:nn
2271       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2272       { \l__enumext_newlabel_arg_two_tl }
2273     }
2274     \l__enumext_store_write_aux_file_tl
2275   }

```

(End of definition for `__enumext_store_internal_ref:.`)

`__enumext_store_anskey_show_wrap:n`

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

```

2276 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2277 {
2278   \par
2279   \bool_if:NT \l__enumext_starred_bool
2280   {
2281     \cs_set:Nn \__enumext_level: { vii }
2282   }
2283   \__enumext_print_keyans_box:cc
2284   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2285   { \l__enumext_labelsep_ \__enumext_level: _dim }
2286   \__enumext_anskey_wrapper:n { #1 }
2287 }

```

(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 $\langle \textit{prop list} \rangle$ when using the `show-pos` key on the left margin next to the “wraps” $\langle \textit{argument} \rangle$ passed to `\anskey` on the right side when using the `show-ans` key.

```

2288 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2289 {
2290   \bool_if:NT \l__enumext_show_answer_bool
2291   {
2292     \__enumext_store_anskey_show_wrap:n { #1 }
2293   }
2294   \bool_if:NT \l__enumext_show_position_bool
2295   {
2296     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2297     {
2298       \group_begin:
2299       \exp_not:N \normalfont
2300       \exp_not:N \footnotesize [ \int_eval:n
2301       {
2302         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2303       }
2304       ]
2305       \group_end:

```

```

2306     }
2307     \__enumext_store_anskey_show_wrap:n { #1 }
2308 }
2309 }

```

(End of definition for __enumext_store_anskey_show_left:n.)

10.26 Common functions for keyans, keyans* and keyanspic

10.26.1 Storing content in prop list

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

```

2310 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2311 {
2312     \tl_clear:N \l__enumext_store_keyans_label_tl
2313     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2314     {
2315         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2316     }
2317     {
2318         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2319     }
2320     \tl_if_novalue:nF { #1 }
2321     {
2322         % Set save-sep
2323         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2324         {
2325             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2326         }
2327         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2328     }
2329     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2330 }

```

(End of definition for __enumext_keyans_addto_prop:n.)

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

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

__enumext_keyans_store_ref: 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.

```

2331 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2332 {
2333     \bool_if:NT \l__enumext_store_ref_key_bool
2334     {
2335         \cs_set_protected:Npn \__enumext_tmp:n ##1
2336         {
2337             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2338             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2339             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2340             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2341         }
2342         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2343         \__enumext_keyans_store_ref_aux_i:
2344     }
2345 }

```

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*.

```

2346 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2347 {

```

```

2348 \bool_if:NT \g__enumext_starred_bool
2349 {
2350   \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2351 }
2352 \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2353 {
2354   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2355   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2356 }
2357 \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2358 {
2359   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2360   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2361 }
2362 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2363 {
2364   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2365   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2366 }
2367 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2368 {
2369   \l__enumext_store_name_tl \c_colon_str
2370   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2371 }
2372 \__enumext_keyans_store_ref_aux_ii:
2373 }

```

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.

```

2374 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2375 {
2376   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2377   {
2378     \__enumext_newlabel:nn
2379     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2380     { \l__enumext_newlabel_arg_two_tl }
2381   }
2382   \l__enumext_store_write_aux_file_tl
2383 }

```

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

10.26.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

2384 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2385 {
2386   \tl_clear:N \l__enumext_store_keyans_label_tl
2387   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2388   {
2389     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2390   }
2391   {
2392     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2393   }
2394   \tl_if_novalue:nF { #1 }
2395   {
2396     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2397     {
2398       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2399       {
2400         \l__enumext_store_keyans_item_opt_sep_tl
2401       }
2402     }
2403     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2404   }
2405   \__enumext_keyans_addto_seq_link:
2406 }

```

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.

```

2407 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2408 {
2409   \bool_lazy_and:nnT
2410     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2411     { \bool_if_p:N \l__enumext_hyperref_bool }
2412   {
2413     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2414       {
2415         \hfill \exp_not:N \hyperlink
2416           {
2417             \exp_not:V \l__enumext_newlabel_arg_one_tl
2418           }
2419           { \exp_not:V \l__enumext_mark_ref_sym_tl }
2420       }
2421   }
2422   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2423   \bool_if:NT \l__enumext_check_ans_bool
2424   {
2425     \int_gincr:N \g__enumext_item_anskey_int
2426   }
2427 }

```

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

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

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

```

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

```

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

```

2428 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2429 {
2430   \tl_if_novalue:nF { #1 }
2431   {
2432     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2433   }
2434   \bool_if:NT \l__enumext_show_answer_bool
2435   {
2436     \__enumext_keyans_show_ans:
2437   }
2438   \bool_if:NT \l__enumext_show_position_bool
2439   {
2440     \__enumext_keyans_show_pos:
2441   }
2442 }
2443 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2444 {
2445   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2446   {
2447     \bool_lazy_or:nnT
2448       { \bool_if_p:N \l__enumext_show_answer_bool }
2449       { \bool_if_p:N \l__enumext_show_position_bool }
2450     {
2451       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2452     }
2453   }
2454 }
2455 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2456 {
2457   \tl_put_left:Nn \l__enumext_label_v_tl
2458   {
2459     \__enumext_print_keyans_box:NN
2460     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2461   }

```

```

2462 }
2463 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2464 {
2465   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2466   {
2467     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2468     {
2469       \group_begin:
2470       \exp_not:N \normalfont
2471       \exp_not:N \footnotesize [ \int_eval:n
2472       {
2473         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2474       }
2475       ]
2476       \group_end:
2477     }
2478   }
2479   {
2480     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2481     {
2482       \group_begin:
2483       \exp_not:N \normalfont
2484       \exp_not:N \footnotesize [ \int_eval:n
2485       {
2486         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2487       }
2488       ]
2489       \group_end:
2490     }
2491   }
2492   \tl_put_left:Nn \l__enumext_label_v_tl
2493   {
2494     \__enumext_print_keyans_box:NN
2495     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2496   }
2497 }

```

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

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

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

```

item-sym* Define and set item-sym* and item-pos* keys for enumext and enumext*.
item-pos*
2498 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2499 {
2500   \keys_define:nn { enumext / #1 }
2501   {
2502     item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2503     item-sym* .value_required:n = true,
2504     item-sym* .initial:n = { $\star$ },
2505     item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2506     item-pos* .value_required:n = true,
2507   }
2508 }
2509 \clist_map_inline:nn
2510 {
2511   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2512 }
2513 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining `\footnote` command

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

```

2514 \cs_new_protected:Nn \__enumext_footnotetext:nn
2515 {
2516   \footnotetext[#1]{#2}

```

```

2517 }
2518 \cs_new_protected:Nn \__enumext_renew_footnote:
2519 {
2520   \seq_gclear:N \g__enumext_footnote_arg_seq
2521   \seq_gclear:N \g__enumext_footnote_int_seq
2522   \RenewDocumentCommand \footnote { o +m }
2523   {
2524     \tl_if_novalue:nTF {##1}
2525     {
2526       \stepcounter{footnote}
2527       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2528     }
2529     {
2530       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2531     }
2532     \footnotemark [ \g__enumext_footnote_int ]
2533     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2534     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2535   }
2536 }
2537 \cs_new_protected:Nn \__enumext_print_footnote:
2538 {
2539   \seq_if_empty:NF \g__enumext_footnote_int_seq
2540   {
2541     \seq_map_pairwise_function:NNN
2542     \g__enumext_footnote_int_seq
2543     \g__enumext_footnote_arg_seq
2544     \__enumext_footnotetext:nn
2545   }
2546 }

```

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

10.29 Redefining \item command

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

10.29.1 The \item command in enumext

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

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `\l__enumext_check_ans_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: (§10.30)`.

```

2547 \cs_new_protected:Npn \__enumext_default_item:n #1
2548 {
2549   \tl_if_novalue:nTF {#1}
2550   {
2551     \bool_if:NT \l__enumext_check_ans_bool
2552     {
2553       \int_gincr:N \g__enumext_item_number_int
2554     }
2555     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2556     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2557   }
2558   {
2559     \bool_set_eq:cc
2560     { l__enumext_wrap_label_ \__enumext_level: _bool }
2561     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2562     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2563   }
2564 }

```

(End of definition for __enumext_default_item:n.)

`__enumext_starred_item:nn`

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

`#1: \l__enumext_item_symbol_X_tl`

`#2: \l__enumext_item_symbol_sep_X_dim`

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

Then we will see the state of the variable `\l__enumext_check_ans_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:` (§10.30).

```

2565 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2566 {
2567   \tl_if_novalue:nF {#1}
2568   {
2569     \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2570   }
2571   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2572   \tl_if_novalue:nTF {#2}
2573   {
2574     \dim_set_eq:cc
2575     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2576     { \l__enumext_labelsep_ \__enumext_level: _dim }
2577   }
2578   {
2579     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2580   }
2581   \bool_if:NT \l__enumext_check_ans_bool
2582   {
2583     \int_gincr:N \g__enumext_item_number_int
2584   }
2585   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2586   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2587 }

```

(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 (§10.31.2).

```

2588 \cs_new_protected:Nn \__enumext_redefine_item:
2589 {
2590   \RenewDocumentCommand \item { s o o }
2591   {
2592     \bool_if:nTF {##1}
2593     {
2594       \__enumext_starred_item:nn {##2} {##3}
2595     }
2596     { \__enumext_default_item:n {##2} }
2597   }
2598 }

```

(End of definition for `__enumext_redefine_item:`.)

10.29.2 The `\item` command in keyans

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

`__enumext_keyans_default_item:n`

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

```

2599 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2600 {
2601   \tl_if_novalue:nTF { #1 }
2602   {
2603     \bool_set_true:N \__enumext_wrap_label_v_bool
2604     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2605   }
2606   {
2607     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2608     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2609   }
2610 }

```

(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 $\langle label \rangle$.

```

2611 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2612 {
2613   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2614   \__enumext_keyans_show_left:n { #1 }
2615   \bool_set_true:N \__enumext_wrap_label_v_bool
2616   \__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 $\langle sequence \rangle$.

```

2617   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2618   \__enumext_keyans_addto_prop:n { #1 }
2619   \__enumext_keyans_store_ref:
2620   \__enumext_keyans_addto_seq:n { #1 }
2621   \int_gincr:N \g__enumext_check_starred_cmd_int
2622 }

```

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

`\item*`

`__enumext_keyans_redefine_item:`

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.31.2).

```

2623 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2624 {
2625   \RenewDocumentCommand \item { s o }
2626   {
2627     \bool_if:nTF {##1}
2628     {
2629       \peek_remove_spaces:n
2630       {
2631         \__enumext_keyans_starred_item:n {##2}
2632       }
2633     }
2634     {
2635       \__enumext_keyans_default_item:n {##2}
2636     }
2637   }
2638 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 12.)

10.30 Redefining \makeLabel command

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

10.30.1 Redefining \makeLabel for enumext

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

```
2639 \cs_new_protected:Nn \__enumext_item_starred:
2640 {
2641   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2642   {
2643     \mode_leave_vertical:
2644     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2645     \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
2646     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2647   }
2648 }
```

(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 (§10.31.2).

```
2649 \cs_new_protected:Nn \__enumext_make_label:
2650 {
2651   \RenewDocumentCommand \makeLabel { m }
2652   {
2653     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2654     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2655     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2656     {
2657       \__enumext_item_starred:
2658       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2659     }
2660     { ##1 }
2661     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2662     \tl_gclear:N \g__enumext_item_symbol_tl
2663   }
2664 }
```

(End of definition for `__enumext_make_label:`)

10.30.2 Redefining \makeLabel for keyans

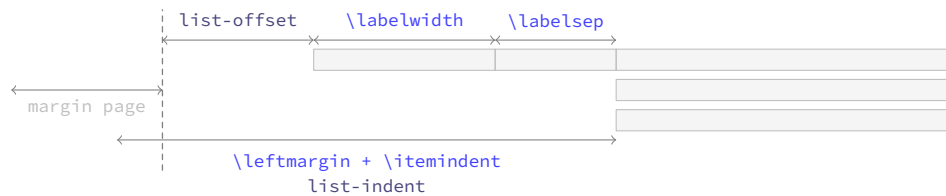
`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment. This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.31.2).

```
2665 \cs_new_protected:Nn \__enumext_keyans_make_label:
2666 {
2667   \RenewDocumentCommand \makeLabel { m }
2668   {
2669     \tl_use:N \l__enumext_label_fill_left_v_tl
2670     \tl_use:N \l__enumext_label_font_style_v_tl
2671     \bool_if:NTF \l__enumext_wrap_label_v_bool
2672     {
2673       \__enumext_wrapper_label_v:n { ##1 }
2674     }
2675     { ##1 }
2676     \tl_use:N \l__enumext_label_fill_right_v_tl
2677   }
2678 }
```

(End of definition for `__enumext_keyans_make_label:`)

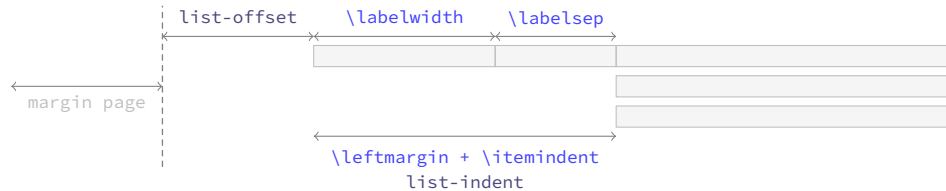
10.31 Second argument of the lists

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

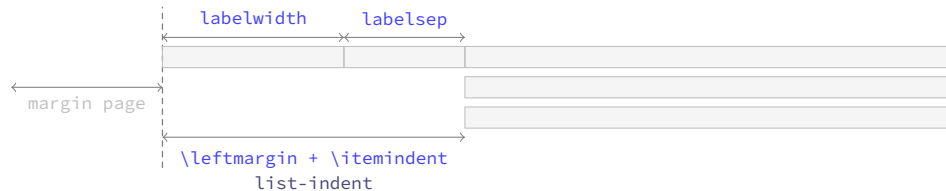
Figure 9: Representation of standard horizontal lengths in `list` environment.

10.31.1 Calculation of `\leftmargin` and `\itemindent`

Consider the figure 9 where the default margins (on the left) of a list are represented. 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 (§10.31.2).

```
2679 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2680 {
2681   \dim_compare:nNt { #1 } < { \c_zero_dim }
2682   {
2683     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2684     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2685   }
2686   \dim_compare:nNt { #2 } < { \c_zero_dim }
2687   {
2688     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2689     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2690   }
2691 }
```

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

```
2691 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2692 \dim_compare:nNtF { #4 } < { \c_zero_dim }
2693 {
2694   \dim_set:Nn #6 { #1 + #2 - #4 }
2695   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2696 }
```

```

2697     {
2698       \dim_compare:nNnT { #4 } = { #1 + #2 }
2699       { \dim_set:Nn #6 { \c_zero_dim } }
2700       \dim_compare:nNnT { #4 } < { #1 + #2 }
2701       { \dim_set:Nn #6 { #1 + #2 - #4 } }
2702       \dim_compare:nNnT { #4 } > { #1 + #2 }
2703       {
2704         \dim_set:Nn #6 { -#1 - #2 + #4 }
2705         \dim_set:Nn #6 { #6*-1 }
2706       }
2707       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2708     }
2709   }
2710   \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(End of definition for __enumext_calc_hspace:NNNNNNN.)

10.31.2 Setting second argument of the lists

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

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
2711   \cs_set_protected:Npn \__enumext_tmp:n #1
2712   {
2713     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2714     {
2715       \__enumext_calc_hspace:ccccc
2716       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2717       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2718       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2719       { \__enumext_leftmargin_tmp_#1_bool }
2720       \clist_map_inline:nn
2721       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2722       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2723       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2724       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2725       \usecounter { enumX#1 }
2726       \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2727       \str_if_eq:nnTF {#1} { v }
2728       {
2729         \__enumext_keyans_redefine_item:
2730         \__enumext_keyans_make_label:
2731         \__enumext_keyans_ref:
2732         \__enumext_keyans_fake_item:
2733         \bool_if:cT { \__enumext_show_length_#1_bool }
2734         {
2735           \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2736         }
2737       }
2738       {
2739         \__enumext_redefine_item:
2740         \__enumext_make_label:
2741         \__enumext_standar_ref:
2742         \__enumext_fake_item:
2743         \bool_if:cT { \__enumext_show_length_#1_bool }
2744         {
2745           \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2746         }
2747       }
2748     }
2749   }
2750   \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

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

```

2751   \cs_set_protected:Npn \__enumext_tmp:n #1
2752   {
2753     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2754     {

```

```

2755 \__enumext_calc_hspace:ccccc
2756 { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2757 { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2758 { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2759 { \__enumext_leftmargin_tmp_#1_bool }
2760 \clist_map_inline:nn
2761 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2762 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2763 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2764 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2765 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2766 \skip_zero:N \partopsep
2767 \usecounter { enumX#1 }
2768 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2769 \__enumext_starred_ref:
2770 \str_if_eq:nnTF {#1} { vii }
2771 {
2772   \__enumext_fake_item_vii:
2773   \bool_if:cT { \__enumext_show_length_vii_bool }
2774   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2775 }
2776 {
2777   \__enumext_fake_item_viii:
2778   \bool_if:cT { \__enumext_show_length_#1_bool }
2779   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2780 }
2781 }
2782 }
2783 \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:`)

10.32 The environment enumext

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

```

2784 \NewDocumentEnvironment{enumext}{ 0{ } }
2785 {
2786   \__enumext_safe_exec:
2787   \__enumext_parse_keys:n {#1}
2788   \__enumext_before_list:
2789   \__enumext_start_store_level:
2790   \__enumext_start_list:nn
2791   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2792   {
2793     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2794     \__enumext_before_keys_exec:
2795   }
2796   \__enumext_after_args_exec:
2797 }
2798 {
2799   \__enumext_stop_list:
2800   \__enumext_stop_store_level:
2801   \__enumext_after_list:
2802 }

```

(End of definition for `enumext`. This function is documented on page 5.)

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first execute the function `__enumext_is_not_nested:` which will set the variable `\g__enumext_standard_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_standard_bool` variable to “true”. Finally we set the variable `\l__enumext_standar_first_level_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:`.

```

2803 \cs_new_protected:Nn \__enumext_safe_exec:
2804 {
2805   \__enumext_is_not_nested:
2806   \int_incr:N \l__enumext_level_int
2807   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2808   { \msg_fatal:nn { enumext } { list-too-deep } }
2809   \bool_set_true:N \l__enumext_standar_bool
2810   \__enumext_is_on_first_level:
2811 }

```

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

`__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`.

```

2812 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2813 {
2814   \tl_if_novalue:nF {#1}
2815   {
2816     \str_clear:N \l__enumext_series_str
2817     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2818     {
2819       \keys_set:nn { enumext / level-1 } {#1}
2820       \__enumext_parse_series:n {#1}
2821     }
2822     {
2823       \exp_args:Ne \keys_set:nn
2824         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2825     }
2826     \bool_if:NT \l__enumext_store_active_bool
2827     {
2828       \__enumext_parse_store_keys:n {#1}
2829     }
2830   }
2831 }

```

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

`__enumext_parse_store_keys:n`

The function `__enumext_parse_store_keys:n` searches for the values of the `columns` and `columns-sep` keys in the optional arguments per-level in `enumext` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\l__enumext_store_opt_X_tl` which is used by the function `__enumext_store_level_open:`.

```

2832 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2833 {
2834   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2835   {
2836     \regex_match:nnT { \b columns\b } {#1}
2837     {
2838       \int_set_eq:cc
2839         { \l__enumext_store_columns_ \__enumext_level: _int }
2840         { \l__enumext_columns_ \__enumext_level: _int }
2841       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2842         {
2843           columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2844         }
2845     }
2846   }
2847   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2848   {
2849     \regex_match:nnT { \b columns-sep\b } {#1}
2850     {
2851       \dim_set_eq:cc
2852         { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2853         { \l__enumext_columns_sep_ \__enumext_level: _dim }
2854       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2855         {
2856           columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2857         }
2858     }
2859   }
2860 }

```

(End of definition for `__enumext_parse_store_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.

```

2861 \cs_new_protected:Nn \__enumext_start_store_level:
2862 {
2863   \bool_lazy_all:nT
2864   {
2865     { \bool_if_p:N \l__enumext_store_active_bool }
2866     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2867     { \bool_not_p:n { \g__enumext_starred_bool } }
2868   }
2869   {
2870     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2871     {
2872       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2873       \__enumext_store_level_open:
2874     }
2875   }
2876   \bool_lazy_all:nT
2877   {
2878     { \bool_if_p:N \l__enumext_store_active_bool }
2879     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2880     { \bool_if_p:N \g__enumext_starred_bool }
2881   }
2882   {
2883     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2884     {
2885       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2886       \__enumext_store_level_open:
2887     }
2888   }
2889 }
2890 \cs_new_protected:Nn \__enumext_stop_store_level:
2891 {
2892   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2893   {
2894     \__enumext_store_level_close:
2895   }
2896 }

```

(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.

```

2897 \cs_new_protected:Nn \__enumext_before_list:
2898 {
2899   \__enumext_vspace_above:
2900   \__enumext_before_args_exec:

```

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

```

2901 \__enumext_check_ans:

```

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.

```

2902 \dim_compare:nNnT
2903 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2904 {
2905   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2906   {
2907     \linewidth
2908     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2909     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2910   }

```

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

the *left* and *right* side environments. After these actions, the function `__enumext_multicols_start:` is called to handle the `multicols` environment.

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

```

2911         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2912         \int_gincr:N \g__enumext_minipage_stat_int
2913         \__enumext_mini_addvspace:
2914         \nointerlineskip\noindent
2915         \begin{\__enumext_mini_env*}
2916         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2917     }
2918     \__enumext_multicols_start:
2919 }

```

(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.

```

2920 \cs_new_protected:Nn \__enumext_multicols_start:
2921 {
2922     \int_compare:nNt
2923     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2924     {
2925         \dim_compare:nNt
2926         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2927         {
2928             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2929             {
2930                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2931                 + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2932                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2933                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2934             }
2935         }
2936         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2937         \skip_zero:N \multicolsep
2938         \int_compare:nNt { \l__enumext_level_int } > { 1 }
2939         {
2940             \dim_zero:N \columnseprule
2941         }
2942     }
2943 }

```

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.

```

2942     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2943     {
2944         \__enumext_multi_addvspace:
2945     }
2946     \raggedcolumns
2947     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2948 }
2949 }

```

(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.

```

2950 \cs_new_protected:Nn \__enumext_multicols_stop:
2951 {
2952     \int_compare:nNt
2953     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2954     {
2955         \end{multicols}
2956         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2957         {
2958             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2959         }
2960     }
2961 }

```

```

2960     }
2961 }

```

(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.

```

2962 \cs_new_protected:Nn \__enumext_after_list:
2963 {
2964   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2965   {
2966     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2967     {
2968       \msg_warning:nn { enumext } { missing-miniright }
2969       \miniright
2970     }
2971     \int_gzero:N \g__enumext_minipage_stat_int
2972     \end{__enumext_mini_env*}
2973     \par\addvspace { \l__enumext_minipage_after_skip }
2974   }
2975   { \__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`.

```

2976   \__enumext_check_ans_to_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.

```

2977   \__enumext_after_stop_list:
2978   \__enumext_vspace_below:
2979   \bool_set_false:N \l__enumext_standar_bool
2980   \__enumext_resume_save_counter:
2981 }

```

(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`.

```

2982 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

10.33 The environment `keyans`

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

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

```

2983 \NewDocumentEnvironment{keyans}{ 0{ } }
2984 {
2985   \__enumext_keyans_safe_exec:
2986   \__enumext_keyans_parse_keys:n {#1}
2987   \__enumext_before_list_v:
2988   \__enumext_start_list:nn
2989   { \tl_use:N \l__enumext_label_v_tl }
2990   {
2991     \__enumext_list_arg_two_v:
2992     \__enumext_before_keys_exec_v:
2993   }
2994   \__enumext_after_args_exec_v:
2995 }
2996 {
2997   \__enumext_check_starred_cmd:n { item }
2998   \__enumext_stop_list:
2999   \__enumext_after_list_v:
3000 }

```

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

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

```

3001 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3002 {
3003   \bool_if:NF \l__enumext_store_active_bool
3004   {
3005     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3006   }
3007   \int_incr:N \l__enumext_keyans_level_int
3008   \bool_set_true:N \l__enumext_keyans_env_bool
3009   \__enumext_keyans_save_start_line:
3010   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3011   \bool_set_false:N \l__enumext_store_active_bool
3012   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3013   {
3014     \msg_error:nn { enumext } { keyans-nested }
3015   }
3016   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3017   {
3018     \msg_error:nn { enumext } { keyans-wrong-level }
3019   }
3020 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

3021 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3022 {
3023   \keys_set:nn { enumext / keyans } { #1 }
3024 }

```

(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.

```

3025 \cs_new_protected:Nn \__enumext_before_list_v:
3026 {
3027   \__enumext_vspace_above_v:
3028   \__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.

```

3029   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3030   {
3031     \dim_set:Nn \l__enumext_minipage_left_v_dim
3032     {
3033       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3034     }

```

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*.

```

3035     \bool_set_true:N \l__enumext_minipage_active_v_bool
3036     \int_gincr:N \g__enumext_minipage_stat_int
3037     \__enumext_keyans_mini_addvspace:
3038     \nointerlineskip\noindent
3039     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3040   }

```

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

```

3041   \__enumext_keyans_multicols_start:
3042 }

```

(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.

```
3043 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3044 {
3045   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3046   {
```

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

```
3047     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3048     {
3049       \dim_set:Nn \__enumext_columns_sep_v_dim
3050       {
3051         (
3052           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3053         ) / \__enumext_columns_v_int
3054         - \__enumext_listoffset_v_dim
3055       }
3056     }
3057     \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).

```
3058     \skip_zero:N \multicolsep
3059     \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.

```
3060     \bool_if:NF \__enumext_minipage_active_v_bool
3061     {
3062       \__enumext_keyans_multi_addvspace:
3063     }
3064     \raggedcolumns
3065     \begin{multicols}{\__enumext_columns_v_int}
3066   }
3067 }
```

(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.

```
3068 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3069 {
3070   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3071   {
3072     \end{multicols}
3073     \bool_if:NF \__enumext_minipage_active_v_bool
3074     {
3075       \par\addvspace{\__enumext_multicols_below_v_skip}
3076     }
3077   }
3078 }
```

(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.

```
3079 \cs_new_protected:Nn \__enumext_after_list_v:
3080 {
3081   \bool_if:NTF \__enumext_minipage_active_v_bool
3082   {
3083     \int_compare:nNt { \__enumext_minipage_stat_int } = { 1 }
3084     {
3085       \msg_warning:nn { enumext } { missing-miniright }
3086       \miniright
```

```

3087     }
3088     \int_gzero:N \g__enumext_minipage_stat_int
3089     \end{__enumext_mini_env*}
3090     \par\addvspace{ \l__enumext_minipage_after_skip }
3091   }
3092   { __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.

```

3093     \bool_set_false:N \l__enumext_keyans_env_bool
3094     __enumext_after_stop_list_v:
3095     __enumext_vspace_below_v:
3096   }

```

(End of definition for `__enumext_after_list_v:`)

10.34 The environment `keyanspic` and `\anspic`

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

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

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

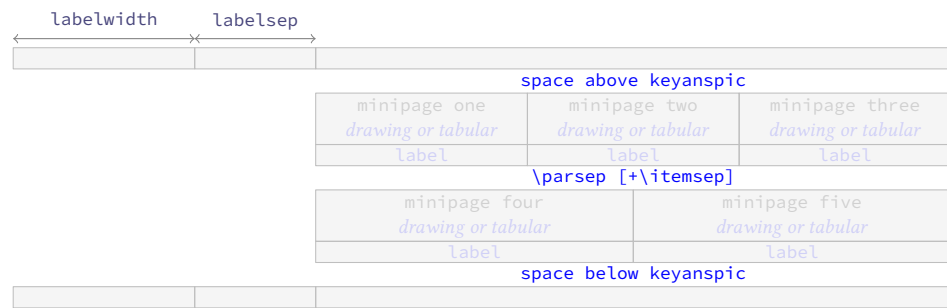


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

10.34.1 The command `\anspic`

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

```

3097 \NewDocumentCommand \anspic { s o +m }
3098 {

```

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

```

3099     \bool_if:NF \l__enumext_store_active_bool
3100     {
3101       \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3102     }
3103     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3104     {
3105       \msg_error:nn { enumext } { keyanspic-wrong-level }
3106     }
3107     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3108     {
3109       \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3110     }

```

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.

```

3111     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3112     {
3113       __enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3114     }
3115   }

```

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

`__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.

```

3116 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3117 {
3118   \stepcounter { enumXvi }
3119   #3 \\\
3120   \bool_if:nT { #1 }
3121   {
3122     \__enumext_keyans_addto_prop:n { #2 }
3123     \__enumext_keyans_store_ref:
3124     \__enumext_keyans_addto_seq:n { #2 }
3125     \int_gincr:N \g__enumext_check_starred_cmd_int
3126     \bool_lazy_or:nnT
3127     { \bool_if_p:N \l__enumext_show_answer_bool }
3128     { \bool_if_p:N \l__enumext_show_position_bool }
3129     {
3130       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3131       \__enumext_keyans_show_left:n { #2 }
3132       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3133     }
3134   }
3135   \tl_use:N \l__enumext_label_font_style_v_tl
3136   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3137 }

```

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

10.34.2 The environment `keyanspic`

`keyanspic`

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

```

3138 \NewDocumentEnvironment{keyanspic}{o}
3139 {
3140   \__enumext_keyans_pic_safe_exec:
3141   \__enumext_start_list:nn
3142   { }
3143   {
3144     \__enumext_keyans_pic_arg_two:
3145   }

```

We apply the “adjusted” vertical spacing above the environment

```

3146   \vspace { \l__enumext_keyans_pic_above_skip }
3147 }

```

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.

```

3148 {
3149   \tl_if_novalue:nTF { #1 }
3150   {
3151     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3152   }
3153   { \__enumext_keyans_pic_do:n { #1 } }
3154   \__enumext_stop_list:
3155   \__enumext_check_starred_cmd:n { anspic }
3156   \setcounter { enumXvi } { 0 }
3157   \vspace { \l__enumext_topsep_v_skip }
3158   %\bool_set_false:N \l__enumext_store_active_bool
3159 }

```

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

`__enumext_keyans_pic_safe_exec:`

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

```

3160 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3161 {
3162   \int_incr:N \l__enumext_keyans_pic_level_int
3163   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3164   {
3165     \msg_error:nn { enumext } { keyanspic-nested }

```



```

3166     }
3167     \__enumext_keyans_save_start_line:
3168 }

```

(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.

```

3169 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3170 {
3171     \dim_compare:nNnT { #1 } < { 0pt }
3172     { \skip_set:Nn #1 { -#1 } }
3173 }

```

(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.

```

3174 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3175 {

```

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.

```

3176     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3177     \__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.

```

3178     \skip_add:Nn \parsep { \itemsep }
3179     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3180     \dim_zero:N \labelwidth
3181     \dim_zero:N \listparindent
3182     \dim_zero:N \labelsep
3183     \skip_zero:N \partopsep
3184     \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 L^AT_EX when not using the \item command.

```

3185     \__enumext_keyans_pic_skip_abs:N \parsep
3186     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3187     {
3188         \box_dp:N \strutbox
3189         + \l__enumext_topsep_v_skip
3190         - \parsep
3191     }
3192     \__enumext_item_std:w \scan_stop:
3193 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

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

```

3194 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3195 {
3196     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3197 }
3198 \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.

```

3199 \cs_new_protected:Npn \__enumext_keyans_pic_row:n
3200 {
3201     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3202     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3203     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }

```

```

3204 \int_step_inline:nnn
3205 { \l__enumext_keyans_pic_above_int + 1 }
3206 { \l__enumext_keyans_pic_below_int }
3207 {
3208   \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3209   \centering
3210   \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3211   \__enumext_endminipage:
3212 }
3213 \par
3214 }

```

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

10.35 The environment `enumext*`

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

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

10.35.1 Functions for item box width

`__enumext_starred_columns_set_vii:`

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

```

3215 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3216 {
3217   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3218   {
3219     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3220     {
3221       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3222       / \l__enumext_columns_vii_int
3223     }
3224   }
3225   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3226   \dim_set:Nn \l__enumext_item_width_vii_dim
3227   {
3228     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3229     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3230     - \l__enumext_labelsep_vii_dim
3231   }
3232   \dim_zero_new:N \itemwidth
3233 }

```

(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`.

```

3234 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3235 {
3236   \int_set:Nn \l__enumext_joined_item_vii_int { #1 }
3237   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3238   {
3239     \msg_warning:nnee { enumext } { item-joined }
3240     { \int_use:N \l__enumext_joined_item_vii_int }
3241     { \int_use:N \l__enumext_columns_vii_int }
3242     \int_set:Nn \l__enumext_joined_item_vii_int
3243     {
3244       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3245     }
3246   }
3247   \int_compare:nNnT
3248   { \l__enumext_joined_item_vii_int }
3249   >
3250   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```

```

3251     {
3252         \msg_warning:nnee { enumext } { item-joined-columns }
3253         { \int_use:N \l__enumext_joined_item_vii_int }
3254         {
3255             \int_eval:n
3256             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3257         }
3258         \int_set:Nn \l__enumext_joined_item_vii_int
3259         {
3260             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3261         }
3262     }

```

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

```

3263     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3264     {
3265         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3266         \int_decr:N \l__enumext_joined_item_aux_vii_int
3267         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3268         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3269         \dim_set:Nn \l__enumext_joined_width_vii_dim
3270         {
3271             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3272             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3273               + \l__enumext_columns_sep_vii_dim
3274               ) * \l__enumext_joined_item_aux_vii_int
3275         }
3276         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3277     }
3278     {
3279         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3280         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3281     }
3282 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n.)

\l__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`.

```

3283 \cs_new_protected:Nn \l__enumext_start_mini_vii:
3284 {
3285     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3286     {
3287         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3288         {
3289             \linewidth
3290             - \l__enumext_minipage_right_vii_dim
3291             - \l__enumext_minipage_hsep_vii_dim
3292         }
3293         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3294         \dim_gset_eq:NN
3295             \g__enumext_minipage_right_vii_dim
3296             \l__enumext_minipage_right_vii_dim
3297         \l__enumext_mini_addvspace_vii:
3298         \nointerlineskip\noindent
3299         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3300     }
3301 }

```

(End of definition for \l__enumext_start_mini_vii:.)

\l__enumext_stop_mini_vii: The function `\l__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 `\l__enumext_after_star_env:n` to execute the `__enumext-mini_env*` on the “right side”.

```

3302 \cs_new_protected:Nn \l__enumext_stop_mini_vii:
3303 {
3304     \bool_if:NT \l__enumext_minipage_active_vii_bool

```

```

3305     {
3306         \end{__enumext_mini_env*}
3307         \hfill
3308         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3309     }
3310 }

```

Finally we execute code passed to the `miniright` key stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “right side”.

```

3311 \__enumext_after_env:nn {enumext*}
3312 {
3313     \bool_if:NT \g__enumext_minipage_active_vii_bool
3314     {
3315         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3316         \par\addvspace { \g__enumext_minipage_right_skip }
3317         \bool_if:NF \g__enumext_minipage_center_vii_bool
3318         {
3319             \centering
3320         }
3321         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3322         \end{__enumext_mini_env*}
3323         \par\addvspace{ \g__enumext_minipage_after_skip }
3324     }
3325     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3326     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3327     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3328     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3329     \bool_gset_false:N \g__enumext_starred_bool
3330 }

```

(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.

```

3331 \NewDocumentEnvironment{enumext*}{ o }
3332 {
3333     \__enumext_safe_exec_vii:
3334     \__enumext_parse_keys_vii:n {#1}
3335     \__enumext_before_list_vii:
3336     \__enumext_start_store_level_vii:
3337     \__enumext_start_list:nn { }
3338     {
3339         \__enumext_list_arg_two_vii:
3340         \__enumext_before_keys_exec_vii:
3341     }
3342     \__enumext_starred_columns_set_vii:
3343     \item[] \scan_stop:
3344     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3345     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3346 }
3347 {
3348     \__enumext_stop_item_tmp_vii:
3349     \__enumext_remove_extra_parsep_vii:
3350     \__enumext_stop_list:
3351     \__enumext_stop_store_level_vii:
3352     \__enumext_after_list_vii:
3353 }

```

(End of definition for `enumext*`. This function is documented on page 5.)

`__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`.

```

3354 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3355 {
3356     \__enumext_is_not_nested:
3357     \int_incr:N \l__enumext_level_h_int
3358     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3359     {
3360         \msg_error:nn { enumext } { nested }
3361     }

```

```

3362     \bool_set_true:N \l__enumext_starred_bool
3363     \__enumext_is_on_first_level:
3364 }

```

(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 function `__enumext_parse_store_keys_vii:n` and reprocess the keys to pass them to the storage sequence.

```

3365 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3366 {
3367     \tl_if_novalue:nF {#1}
3368     {
3369         \str_clear:N \l__enumext_series_str
3370         \keys_set:nn { enumext / enumext* } {#1}
3371         \__enumext_parse_series:n {#1}
3372         \bool_if:NT \l__enumext_store_active_bool
3373         {
3374             \__enumext_parse_store_keys_vii:n {#1}
3375         }
3376     }
3377 }

```

(End of definition for __enumext_parse_keys_vii:n.)

__enumext_parse_store_keys_vii:n

The function `__enumext_parse_store_keys_vii:n` searches for the values of the `columns` and `columns-sep` keys in the optional argument in `enumext*` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\l__enumext_store_opt_vii_tl` which is used by the function `__enumext_store_level_open_vii:.`

```

3378 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3379 {
3380     \bool_if:NF \l__enumext_store_columns_vii_bool
3381     {
3382         \regex_match:nnT { \b columns\b } {#1}
3383         {
3384             \int_set_eq:NN
3385             \l__enumext_store_columns_vii_int
3386             \l__enumext_columns_vii_int
3387             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3388             {
3389                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3390             }
3391         }
3392     }
3393     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3394     {
3395         \regex_match:nnT { \b columns-sep\b } {#1}
3396         {
3397             \dim_set_eq:NN
3398             \l__enumext_store_columns_sep_vii_dim
3399             \l__enumext_columns_sep_vii_dim
3400             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3401             {
3402                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3403             }
3404         }
3405     }
3406 }

```

(End of definition for __enumext_parse_store_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`.

```

3407 \cs_new_protected:Npn \__enumext_before_list_vii:
3408 {
3409     \__enumext_vspace_above_vii:
3410     \__enumext_check_ans: % need by chek-ans
3411     \__enumext_before_args_exec_vii:
3412     \__enumext_start_mini_vii:
3413 }

```

(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.

```

3414 \cs_new_protected:Nn \__enumext_after_list_vii:
3415 {
3416   \__enumext_stop_mini_vii:
3417   \__enumext_after_stop_list_vii:
3418   \__enumext_check_ans_to_hook:
3419   \__enumext_vspace_below_vii:
3420   \bool_set_false:N \l__enumext_starred_bool
3421   \__enumext_resume_save_counter:
3422 }

```

(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`.

`__enumext_stop_store_level_vii:`

```

3423 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3424 {
3425   \bool_if:NT \l__enumext_store_active_bool
3426   {
3427     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3428     {
3429       \__enumext_store_level_open_vii:
3430     }
3431   }
3432 }
3433 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3434 {
3435   \bool_if:NT \l__enumext_store_active_bool
3436   {
3437     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3438     {
3439       \__enumext_store_level_close_vii:
3440     }
3441   }
3442 }

```

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

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

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

```

3443 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3444 {
3445   \__enumext_stop_item_tmp_vii:
3446   \int_incr:N \l__enumext_item_column_pos_vii_int
3447   \int_gincr:N \g__enumext_item_count_all_vii_int
3448   \__enumext_item_peek_args_vii:
3449 }

```

(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).

```

3450 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3451 {
3452   \peek_meaning:NTF (
3453     { \__enumext_joined_item_vii:w }

```

```

3454     { \__enumext_joined_item_vii:w (1) }
3455 }

```

(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_standard_item_vii:w.

```

3456 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3457 {
3458   \__enumext_starred_joined_item_vii:n {#1}
3459   \peek_meaning_remove:NTF *
3460   { \__enumext_starred_item_vii:w }
3461   { \__enumext_standard_item_vii:w }
3462 }

```

(End of definition for __enumext_joined_item_vii:w.)

__enumext_standard_item_vii:w

The function __enumext_standard_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].

```

3463 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3464 {
3465   \bool_set_false:N \l__enumext_item_starred_vii_bool
3466   \peek_meaning:NTF [
3467   {
3468     \bool_set_eq:NN
3469     \l__enumext_wrap_label_vii_bool
3470     \l__enumext_wrap_label_opt_vii_bool
3471     \__enumext_start_item_vii:w
3472   }
3473   {
3474     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3475     \legacy_if_set_true:n { @noitemarg }
3476     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3477   }
3478 }

```

(End of definition for __enumext_standard_item_vii:w.)

__enumext_starred_item_vii:w
 __enumext_starred_item_vii_aux_i:w
 __enumext_starred_item_vii_aux_ii:w
 __enumext_starred_item_vii_aux_iii:w

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

```

3479 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3480 {
3481   \bool_set_true:N \l__enumext_item_starred_vii_bool
3482   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3483   \peek_meaning:NTF [
3484   { \__enumext_starred_item_vii_aux_i:w }
3485   { \__enumext_starred_item_vii_aux_ii:w }
3486 }
3487 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3488 {
3489   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3490   \__enumext_starred_item_vii_aux_ii:w
3491 }
3492 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3493 {
3494   \peek_meaning:NTF [
3495   { \__enumext_starred_item_vii_aux_iii:w }
3496   {
3497     \dim_set_eq:NN
3498     \l__enumext_item_symbol_sep_vii_dim
3499     \l__enumext_labelsep_vii_dim
3500     \legacy_if_set_true:n { @noitemarg }
3501     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]

```



```

3502     }
3503   }
3504   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3505   {
3506     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3507     \legacy_if_set_true:n { @noitemarg }
3508     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3509   }

```

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

10.35.3 Real definition of \item in enumext*

__enumext_start_item_vii:w

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

The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to the value of __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.

```

3510 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3511 {
3512   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3513   \legacy_if:nT { @noitemarg }
3514   {
3515     \legacy_if_set_false:n { @noitemarg }
3516     \legacy_if:nT { @nmbrrlist }
3517     {
3518       \bool_if:NT \l__enumext_hyperref_bool
3519       {
3520         \legacy_if_set_true:n { @hyper@item }
3521       }
3522       \refstepcounter{enumXvii}
3523       \bool_if:NT \l__enumext_check_ans_bool
3524       {
3525         \int_gincr:N \g__enumext_item_number_int
3526       }
3527     }
3528   }

```

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

```

3529   \group_begin:
3530   \lrbox{ \l__enumext_item_text_vii_box }
3531   \bool_if:NF \l__enumext_footnotes_key_bool
3532   {
3533     \__enumext_renew_footnote:
3534   }
3535   \bool_if:NT \l__enumext_item_starred_vii_bool
3536   {
3537     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3538     {
3539       \tl_gset_eq:NN
3540       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3541     }
3542     \mode_leave_vertical:
3543     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3544     \makebox[ 0pt ]{ r }{ \g__enumext_item_symbol_aux_vii_tl }
3545     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3546     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3547   }
3548   \group_begin:
3549   \tl_use:N \l__enumext_label_font_style_vii_tl
3550   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3551   {
3552     \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }

```

```

3553         { \__enumext_wrapper_label_vii:n {#1} }
3554     }
3555     {
3556         \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3557     }
3558     \group_end:
3559     \skip_horizontal:N \__enumext_labelsep_vii_dim
3560     \tl_use:N \__enumext_after_list_args_vii_tl
3561     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3562     \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3563     \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3564     \tl_use:N \__enumext_fake_item_indent_vii_tl
3565 }

```

(End of definition for __enumext_start_item_vii:w.)

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

```

3566 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3567 {
3568     \__enumext_endminipage:
3569     \endlrbox
3570     \group_end:
3571     \box_set_wd:Nn \__enumext_item_text_vii_box
3572     {
3573         \__enumext_joined_width_vii_dim
3574         + \__enumext_labelwidth_vii_dim
3575         + \__enumext_labelsep_vii_dim
3576     }
3577     \int_set:Nn \hbadness { 10000 }
3578     \box_use:N \__enumext_item_text_vii_box
3579     \bool_if:NF \__enumext_footnotes_key_bool
3580     {
3581         \__enumext_print_footnote:
3582     }
3583     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3584     {
3585         \par\noindent
3586         \int_zero:N \__enumext_item_column_pos_vii_int
3587     }
3588     { \hspace{ \__enumext_columns_sep_vii_dim } }
3589 }

```

(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.

```

3590 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3591 {
3592     \int_compare:nNnT
3593     {
3594         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3595     }
3596     =
3597     { \c_zero_int }
3598     {
3599         \par
3600         \vspace{ -\__enumext_itemsep_vii_skip }
3601         \int_gzero:N \g__enumext_item_count_all_vii_int
3602     }
3603 }

```

(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.

```

3604 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

10.36 The environment keyans*

10.36.1 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.

```

3605 \cs_new_protected:Nn \_enumext_starred_columns_set_viii:
3606 {
3607   \dim_compare:nNnT { \_enumext_columns_sep_viii_dim } = { \c_zero_dim }
3608   {
3609     \dim_set:Nn \_enumext_columns_sep_viii_dim
3610     {
3611       ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim )
3612       / \_enumext_columns_viii_int
3613     }
3614   }
3615   \int_set:Nn \_enumext_tmpa_viii_int { \_enumext_columns_viii_int - \c_one_int }
3616   \dim_set:Nn \_enumext_item_width_viii_dim
3617   {
3618     ( \linewidth - \_enumext_columns_sep_viii_dim * \_enumext_tmpa_viii_int )
3619     / \_enumext_columns_viii_int - \_enumext_labelwidth_viii_dim
3620     - \_enumext_labelsep_viii_dim
3621   }
3622   \dim_zero_new:N \itemwidth
3623 }

```

(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`.

```

3624 \cs_new_protected:Npn \_enumext_starred_joined_item_viii:n #1
3625 {
3626   \int_set:Nn \_enumext_joined_item_viii_int {#1}
3627   \int_compare:nNnT { \_enumext_joined_item_viii_int } > { \_enumext_columns_viii_int }
3628   {
3629     \msg_warning:nnee { enumext } { item-joined }
3630     { \int_use:N \_enumext_joined_item_viii_int }
3631     { \int_use:N \_enumext_columns_viii_int }
3632     \int_set:Nn \_enumext_joined_item_viii_int
3633     {
3634       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3635     }
3636   }
3637   \int_compare:nNnT
3638   { \_enumext_joined_item_viii_int }
3639   >
3640   { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3641   {
3642     \msg_warning:nnee { enumext } { item-joined-columns }
3643     { \int_use:N \_enumext_joined_item_viii_int }
3644     {
3645       \int_eval:n
3646       { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3647     }
3648     \int_set:Nn \_enumext_joined_item_viii_int
3649     {
3650       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3651     }
3652   }
3653   \int_compare:nNnTF { \_enumext_joined_item_viii_int } > { \c_one_int }
3654   {
3655     \int_set_eq:NN \_enumext_joined_item_aux_viii_int \_enumext_joined_item_viii_int
3656     \int_decr:N \_enumext_joined_item_aux_viii_int
3657     \int_add:Nn \_enumext_item_column_pos_viii_int { \_enumext_joined_item_aux_viii_int }
3658     \int_gadd:Nn \_enumext_item_count_all_viii_int { \_enumext_joined_item_aux_viii_int }
3659     \dim_set:Nn \_enumext_joined_width_viii_dim
3660     {
3661       \_enumext_item_width_viii_dim * \_enumext_joined_item_viii_int
3662       + ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim
3663         + \_enumext_columns_sep_viii_dim
3664         ) * \_enumext_joined_item_aux_viii_int
3665     }

```

```

3666         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3667     }
3668     {
3669         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3670         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3671     }
3672 }

```

(End of definition for __enumext_starred_joined_item_viii:n)

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

```

\__enumext_stop_mini_viii:
3673 \cs_new_protected:Nn \__enumext_start_mini_viii:
3674 {
3675     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3676     {
3677         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3678         {
3679             \linewidth
3680             - \l__enumext_minipage_right_viii_dim
3681             - \l__enumext_minipage_hsep_viii_dim
3682         }
3683         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3684         \dim_gset_eq:NN
3685             \g__enumext_minipage_right_viii_dim
3686             \l__enumext_minipage_right_viii_dim
3687         \__enumext_mini_addvspace_viii:
3688         \nointerlineskip\noindent
3689         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3690     }
3691 }
3692 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3693 {
3694     \bool_if:NT \l__enumext_minipage_active_viii_bool
3695     {
3696         \end{__enumext_mini_env*}
3697         \hfill
3698         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3699     }
3700 }
3701 \__enumext_after_env:nn {keyans*}
3702 {
3703     \bool_if:NT \g__enumext_minipage_active_viii_bool
3704     {
3705         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3706         \par\addvspace { \g__enumext_minipage_right_skip }
3707         \bool_if:NF \g__enumext_minipage_center_viii_bool
3708         {
3709             \centering
3710         }
3711         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3712         \end{__enumext_mini_env*}
3713         \par\addvspace{ \g__enumext_minipage_after_skip }
3714     }
3715     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3716     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3717     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3718     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3719 }

```

(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.

```

3720 \NewDocumentEnvironment{keyans*}{ o }
3721 {
3722     \__enumext_safe_exec_viii:
3723     \__enumext_parse_keys_viii:n {#1}
3724     \__enumext_before_list_viii:
3725     \__enumext_start_list:nn { }

```

```

3726     {
3727         \__enumext_list_arg_two_viii:
3728         \__enumext_before_keys_exec_viii:
3729     }
3730     \__enumext_starred_columns_set_viii:
3731     \item[] \scan_stop:
3732     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3733     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3734 }
3735 {
3736     \__enumext_stop_item_tmp_viii:
3737     \__enumext_remove_extra_parsep_viii:
3738     \__enumext_check_starred_cmd:n { item }
3739     \__enumext_stop_list:
3740     \__enumext_after_list_viii:
3741 }

```

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

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

```

3742 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3743 {
3744     \int_incr:N \l__enumext_keyans_level_h_int
3745     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3746     {
3747         \msg_error:nn { enumext } { nested }
3748     }
3749     \__enumext_keyans_save_start_line:
3750     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3751     \bool_set_false:N \l__enumext_store_active_bool
3752     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3753     {
3754         \msg_error:nn { enumext } { keyans-wrong-level }
3755     }
3756 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

3757 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3758 {
3759     \tl_if_novalue:nF {#1}
3760     {
3761         \keys_set:nn { enumext / keyans* } {#1}
3762     }
3763 }

```

(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, then call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```

3764 \cs_new_protected:Nn \__enumext_before_list_viii:
3765 {
3766     \__enumext_vspace_above_viii:
3767     \__enumext_before_args_exec_viii:
3768     \__enumext_start_mini_viii:
3769 }

```

(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.

```

3770 \cs_new_protected:Nn \__enumext_after_list_viii:
3771 {
3772     \__enumext_stop_mini_viii:
3773     \__enumext_after_stop_list_viii:
3774     \__enumext_vspace_below_viii:
3775 }

```

(End of definition for `__enumext_after_list_viii:`)

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

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

`__enumext_start_item_tmp_viii:`

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

```
3776 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3777 {
3778   \__enumext_stop_item_tmp_viii:
3779   \int_incr:N \l__enumext_item_column_pos_viii_int
3780   \int_gincr:N \g__enumext_item_count_all_viii_int
3781   \__enumext_item_peek_args_viii:
3782 }
```

(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).

```
3783 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3784 {
3785   \peek_meaning:NTF (
3786     { \__enumext_joined_item_viii:w }
3787     { \__enumext_joined_item_viii:w (1) }
3788   }
```

(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_standard_item_viii:w`.

```
3789 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3790 {
3791   \__enumext_starred_joined_item_viii:n {#1}
3792   \peek_meaning_remove:NTF *
3793     { \__enumext_starred_item_viii:w }
3794     { \__enumext_standard_item_viii:w }
3795 }
```

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

`__enumext_standard_item_viii:w`

The function `__enumext_standard_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item [<custom>]` by means of the function `__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_viii:w [__enumext_label_viii_tl]`.

```
3796 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3797 {
3798   \bool_set_false:N \l__enumext_item_starred_viii_bool
3799   \peek_meaning:NTF [
3800     {
3801       \bool_set_eq:NN
3802       \l__enumext_wrap_label_viii_bool
3803       \l__enumext_wrap_label_opt_viii_bool
3804       \__enumext_start_item_viii:w
3805     }
3806     {
3807       \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

3808         \legacy_if_set_true:n { @noitemarg }
3809         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3810     }
3811 }

```

(End of definition for __enumext_standard_item_viii:w.)

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w

```

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

```

3812 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3813 {
3814     \bool_set_true:N \__enumext_item_starred_viii_bool
3815     \bool_set_true:N \__enumext_wrap_label_viii_bool
3816     \peek_meaning:NTF [
3817         { \__enumext_starred_item_viii_aux_i:w }
3818         { \__enumext_starred_item_viii_aux_ii:w }
3819     }

```

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

```

3820 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3821 {
3822     \tl_clear:N \__enumext_store_keyans_label_tl
3823     \tl_if_no_value:nF { #1 }
3824     {
3825         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3826         {
3827             \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
3828             \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
3829         }
3830         \tl_set:Ne \__enumext_keyans_item_opt_tl { #1 }
3831     }
3832     \__enumext_starred_item_viii_aux_ii:w
3833 }
3834 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3835 {
3836     \legacy_if_set_true:n { @noitemarg }
3837     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3838 }

```

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

```
\__enumext_starred_item_exec:
```

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

```

3839 \cs_new_protected:Nn \__enumext_starred_item_exec:
3840 {
3841     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3842     \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3843     \__enumext_keyans_store_ref:
3844     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3845     \__enumext_keyans_addto_seq_link:
3846     \int_gincr:N \g__enumext_check_starred_cmd_int
3847     \bool_if:NT \__enumext_show_answer_bool
3848     {
3849         \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3850     }
3851     \bool_if:NT \__enumext_show_position_bool
3852     {
3853         \tl_set:Ne \__enumext_mark_answer_sym_tl
3854         {
3855             \group_begin:
3856             \exp_not:N \normalfont
3857             \exp_not:N \footnotesize [ \int_eval:n
3858                 {
3859                     \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3860                 }

```

```

3861         ]
3862     \group_end:
3863 }
3864 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
3865 }
3866 }

```

(End of definition for `__enumext_starred_item_exec:`)

Real definition of `\item` in `keyans*`

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

```

3867 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3868 {
3869     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3870     \legacy_if:nT { @noitemarg }
3871     {
3872         \legacy_if_set_false:n { @noitemarg }
3873         \legacy_if:nT { @nmbrlist }
3874         {
3875             \bool_if:NT \l__enumext_hyperref_bool
3876             {
3877                 \legacy_if_set_true:n { @hyper@item }
3878             }
3879             \refstepcounter{enumXviii}
3880         }
3881     }

```

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

```

3882     \group_begin:
3883     \lrbox{ \l__enumext_item_text_viii_box }
3884     \bool_if:NF \l__enumext_footnotes_key_bool
3885     {
3886         \__enumext_renew_footnote:
3887     }
3888     \bool_if:NT \l__enumext_item_starred_viii_bool
3889     {
3890         \__enumext_starred_item_exec:
3891     }
3892     \group_begin:
3893     \tl_use:N \l__enumext_label_font_style_viii_tl
3894     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3895     {
3896         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3897         { \__enumext_wrapper_label_viii:n {#1} }
3898     }
3899     {
3900         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3901     }
3902     \group_end:
3903     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3904     \tl_use:N \l__enumext_after_list_args_viii_tl
3905     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3906     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3907     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3908     \bool_if:NT \l__enumext_item_starred_viii_bool
3909     {
3910         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3911         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3912     }
3913     {
3914         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3915     }
3916 }

```

(End of definition for `__enumext_start_item_viii:w`)

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

```

3917 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3918 {

```



```

3919     \__enumext_endminipage:
3920     \endlrbox
3921     \group_end:
3922     \box_set_wd:Nn \l__enumext_item_text_viii_box
3923     {
3924         \l__enumext_joined_width_viii_dim
3925         + \l__enumext_labelwidth_viii_dim
3926         + \l__enumext_labelsep_viii_dim
3927     }
3928     \int_set:Nn \hbadness { 10000 }
3929     \box_use:N \l__enumext_item_text_viii_box
3930     \bool_if:NF \l__enumext_footnotes_key_bool
3931     {
3932         \__enumext_print_footnote:
3933     }
3934     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3935     {
3936         \par\noindent
3937         \int_zero:N \l__enumext_item_column_pos_viii_int
3938     }
3939     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3940 }

```

(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.

```

3941 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3942 {
3943     \int_compare:nNnT
3944     {
3945         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3946     }
3947     =
3948     { \c_zero_int }
3949     {
3950         \par
3951         \vspace{ -\l__enumext_itemsep_viii_skip }
3952         \int_gzero:N \g__enumext_item_count_all_viii_int
3953     }
3954 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

10.37 The command \getkeyans

\getkeyans

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

```

3955 \NewDocumentCommand \getkeyans { m }
3956 {
3957     \exp_args:Ne \__enumext_getkeyans_aux:n
3958     { \tl_to_str:e { \text_expand:n {#1} } }
3959 }

```

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

__enumext_getkeyans_aux:n

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

```

3960 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3961 {
3962     \str_if_in:nnTF {#1} { : }
3963     {
3964         \use:e
3965         {
3966             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3967             { {##1} {##2} }
3968         }
3969         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3970     }
3971     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3972 }

```

(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⟩*.

```

3973 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3974 {
3975   \prop_if_exist:cF { g__enumext_#1_prop }
3976   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3977   \group_begin:
3978   \prop_item:cn { g__enumext_#1_prop }{#2}
3979   \group_end:
3980 }

```

(End of definition for __enumext_getkeyans:nn.)

10.38 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 to define a set of *⟨keys⟩* with which we will control the options of the different nesting levels for the *enumext* and *enumext** environment by storing the values of these in the token list variables \l__enumext_print_keyans_X_tl.

```

3981 \keys_define:nn { keyanskey / print }
3982 {
3983   level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3984                       {
3985                         \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3986                       },
3987   level-1 .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3988   level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3989                       {
3990                         \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3991                       },
3992   level-2 .initial:n   = { nosep, label=(\alph*), first=\small, font=\small },
3993   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3994                       {
3995                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3996                       },
3997   level-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
3998   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3999                       {
4000                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
4001                       },
4002   level-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
4003   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
4004                       {
4005                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
4006                       },
4007   level-* .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
4008 }

```

\printkeyans Create a user command to print “all stored content” in *⟨sequence⟩* for *\anskey*, *\item** and *\anspic**.

```

4009 \NewDocumentCommand \printkeyans { s O{} m }
4010 {
4011   \group_begin:
4012   \tl_use:N \l__enumext_print_keyans_i_tl
4013   \tl_use:N \l__enumext_print_keyans_ii_tl
4014   \tl_use:N \l__enumext_print_keyans_iii_tl
4015   \tl_use:N \l__enumext_print_keyans_iv_tl
4016   \tl_use:N \l__enumext_print_keyans_vii_tl
4017   \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4018   \group_end:
4019 }

```

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

__enumext_printkeyans:nnn The internal function __enumext_printkeyans:nnn will check for the existence of the *⟨sequence⟩*, if it does not exist it will return an error message, then it will fetch the content specified by the first argument mapping the *⟨sequence⟩*.

#1: starred

#2: key-val

#3: seq-name

```

4020 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4021 {
4022   \seq_if_exist:cTF { g__enumext_#3_seq }
4023   {
4024     \seq_if_empty:cF { g__enumext_#3_seq }
4025     {
4026       %%\seq_show:c { g__enumext_#3_seq }
4027       \bool_if:nTF {#1}
4028       {
4029         \begin{enumext*}[#2]
4030         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4031         \end{enumext*}
4032       }
4033       {
4034         \begin{enumext}[#2]
4035         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4036         \end{enumext}
4037       }
4038     }
4039   }
4040   {
4041     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4042   }
4043 }

```

(End of definition for __enumext_printkeyans:nnn.)

10.39 The command \setenumext

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

```

4044 \keys_define:nn { enumext / meta-families }
4045 {
4046   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4047   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4048   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4049   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4050   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4051   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4052   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4053   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
4054   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
4055   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
4056   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
4057   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
4058   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4059 }

```

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

```

4060 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4061 {
4062   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
4063   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
4064 }

```

\setenumext Now we define the user command \setenumext.

```

4065 \NewDocumentCommand \setenumext { o +m }
4066 {
4067   \tl_if_novalue:nTF {#1}
4068   {
4069     \seq_map_inline:Nn \c__enumext_all_families_seq
4070     }
4071   {
4072     \seq_clear:N \l__enumext_setkey_tmpa_seq
4073     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4074     \int_set:Nn \l__enumext_setkey_tmpa_int
4075     {
4076       \seq_count:N \l__enumext_setkey_tmpb_seq
4077     }
4078     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4079     {

```

```

4080         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4081         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4082         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4083         {
4084             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4085         }
4086     }
4087     {
4088         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4089     }
4090     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4091     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4092     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4093 }
4094 {
4095     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4096 }
4097 }

```

(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.

```

4098 \cs_new_protected:Npn \__enumext_set_parse:n #1
4099 {
4100     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4101     \int_step_inline:nnn { 0 } { 4 } % <- max level
4102     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4103     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4104     {
4105         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4106         { \tl_trim_spaces:n {#1} }
4107     }
4108     { \__enumext_set_error:nn {#1} { } }
4109 }
4110 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4111 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

10.40 Messages

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

```

4112 \msg_new:nnn { enumext } { package-load }
4113 {
4114     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4115 }
4116 \msg_new:nnn { enumext } { package-not-load }
4117 {
4118     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4119 }
4120 \msg_new:nnn { enumext } { package-load-foot }
4121 {
4122     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4123 }

```

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

```

4124 \msg_new:nnn { enumext } { counters }
4125 {
4126     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
4127     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4128 }

```

Message used in the creation of *(prop list)* by `enumext` package. los valores seria save-name, prop-count, -seqcount, y int-val

```

4129 \msg_new:nnn { enumext } { store-prop }
4130 {
4131     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4132 }
4133 \msg_new:nnn { enumext } { store-seq }
4134 {
4135     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4136 }
4137 \msg_new:nnn { enumext } { store-int }

```

```

4138 {
4139     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4140 }
4141 \msg_new:nnn { enumext } { prop-seq-int-hook }
4142 {
4143     * ~ Package ~ enumext: ~ Answers ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\
4144     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\
4145     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4146 }
4147 \msg_new:nnn { enumext } { item-answer-hook }
4148 {
4149     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
4150     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
4151     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4152 }

```

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

```

4153 \msg_new:nnn { enumext } { invalid-key }
4154 {
4155     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4156 }
4157 \msg_new:nnn { enumext } { unknown-key-family }
4158 {
4159     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4160 }

```

Messages used in length calculation.

```

4161 \msg_new:nnn { enumext } { width-negative }
4162 {
4163     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4164     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4165 }
4166 \msg_new:nnn { enumext } { width-zero }
4167 {
4168     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4169     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4170 }

```

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

```

4171 \msg_new:nnn { enumext } { list-lengths }
4172 {
4173     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4174     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4175     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4176     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4177     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4178     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4179     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4180     \__enumext_show_length:nnn { skip } { topsep } {#1}
4181     \__enumext_show_length:nnn { skip } { parsep } {#1}
4182     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4183     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4184     ****
4185 }

```

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

```

4186 \msg_new:nnn { enumext } { list-lengths-not-nested }
4187 {
4188     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4189     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4190     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4191     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4192     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4193     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4194     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4195     \__enumext_show_length:nnn { skip } { topsep } {#1}
4196     \__enumext_show_length:nnn { skip } { parsep } {#1}
4197     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4198     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4199     ****
4200 }

```

Messages used by `ref` key.

```
4201 \msg_new:nnn { enumext } { key-ref-empty }
4202 {
4203   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4204 }
```

Messages used by `save-ans` key.

```
4205 \msg_new:nnn { enumext } { save-ans-empty }
4206 {
4207   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4208 }
4209 \msg_new:nnn { enumext } { save-ans-log }
4210 {
4211   * ~ Package ~ enumext: ~ Begin ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
ans=#2 ~ \msg_line_context:.
4212 }
4213 \msg_new:nnn { enumext } { save-ans-log-hook }
4214 {
4215   * ~ Package ~ enumext: ~ End ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
ans=#2 ~ \msg_line_context:.
4216 }
4217 \msg_new:nnn { enumext } { save-ans-hook }
4218 {
4219   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4220 }
```

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

```
4221 \msg_new:nnn { enumext } { need-save-ans }
4222 {
4223   Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4224 }
4225 \msg_new:nnn { enumext } { items-same-answer }
4226 {
4227   *****\\
4228   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_
4229   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\\
4230   *****
4231 }
4232 \msg_new:nnn { enumext } { item-greater-answer }
4233 {
4234   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4235   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4236   Items ~ > ~ Answers.
4237 }
4238 \msg_new:nnn { enumext } { item-less-answer }
4239 {
4240   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4241   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4242   Items ~ < ~ Answers.
4243 }
```

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

```
4244 \msg_new:nnn { enumext } { missing-starred }
4245 {
4246   Missing ~ '\c_backslash_str #1*' ~ #2.
4247 }
4248 \msg_new:nnn { enumext } { many-starred }
4249 {
4250   Many ~ '\c_backslash_str #1*' ~ #2.
4251 }
```

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

```
4252 \msg_new:nnn { enumext } { list-too-deep }
4253 {
4254   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4255   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4256 }
```

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

```
4257 \msg_new:nnn { enumext } { anskey-wrong-place }
4258 {
4259   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4260   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
```

```

4261 }
4262 \msg_new:nnn { enumext } { anspic-wrong-place }
4263 {
4264   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4265   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4266 }
4267 \msg_new:nnn { enumext } { command-wrong-place }
4268 {
4269   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4270   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4271 }

```

Messages used by [keyans](#) and [keyanspic](#) environment.

```

4272 \msg_new:nnn { enumext } { keyans-nested }
4273 {
4274   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4275 }
4276 \msg_new:nnn { enumext } { keyans-wrong-level }
4277 {
4278   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4279   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4280 }
4281 \msg_new:nnn { enumext } { wrong-place }
4282 {
4283   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4284   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4285 }
4286 \msg_new:nnn { enumext } { keyanspic-nested }
4287 {
4288   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4289 }
4290 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4291 {
4292   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4293   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4294 }

```

Messages used by [\getkeyans](#) command.

```

4295 \msg_new:nnn { enumext } { undefined-storage-anskey }
4296 {
4297   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4298 }

```

Messages used by [\miniright](#) command.

```

4299 \msg_new:nnn { enumext } { missing-miniright }
4300 {
4301   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4302   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4303 }
4304 \msg_new:nnn { enumext } { wrong-miniright-place }
4305 {
4306   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4307   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4308 }
4309 \msg_new:nnn { enumext } { wrong-miniright-use }
4310 {
4311   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4312   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4313 }

```

Messages used by [enumext*](#) and [keyans*](#) environments.

```

4314 \msg_new:nnn { enumext } { nested }
4315 {
4316   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4317 }
4318 \msg_new:nnn { enumext } { item-joined }
4319 {
4320   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4321 }
4322 \msg_new:nnn { enumext } { item-joined-columns }
4323 {
4324   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
4325 }

```

10.41 Finish package

Finish package implementation.

```
4326 \file_input_stop:
4327 </package>
```


11 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, 3119, 4126, 4143, 4144, 4149, 4150, 4163, 4168, 4173, 4188, 4227, 4228, 4229, 4234, 4235, 4240, 4241, 4254, 4259, 4264, 4269, 4278, 4283, 4292, 4301, 4306, 4311
A	
<code>above</code>	1383
<code>above*</code>	1383
<code>\addvspace</code>	1030, 1058, 1181, 1260, 1323, 1329, 1357, 1374, 2958, 2973, 3075, 3090, 3316, 3323, 3706, 3713
<code>after</code>	868
<code>align</code>	485
<code>\Alph</code>	32, 37
<code>\Alph</code>	437, 546, 591, 659, 4002
<code>\alph</code>	32, 37
<code>\alph</code>	438, 544, 3992
<code>\anskey</code>	11, 66, 2096
<code>\anspic</code>	13, 87, 3097
<code>\anspic*</code>	62
<code>\arabic</code>	27, 32
<code>\arabic</code>	436, 543, 590, 3987, 4007
B	
<code>\b</code>	2836, 2849, 3382, 3395
<code>\baselineskip</code>	45
<code>\baselineskip</code>	2056, 2064
<code>before</code>	868
<code>before*</code>	868
<code>below</code>	1383
<code>below*</code>	1383
bool commands:	
<code>\bool_gset_false:N</code>	299, 300, 301, 3325, 3329, 3715
<code>\bool_gset_true:N</code>	222, 231, 972, 1873, 1879, 3308, 3326, 3698, 3716
<code>\bool_if:NTF</code>	342, 377, 389, 406, 1405, 1419, 1432, 1443, 1454, 1465, 1476, 1487, 1540, 1557, 1562, 1570, 1597, 1635, 1640, 1647, 1651, 1673, 1678, 1686, 1693, 1724, 1732, 1815, 1819, 1823, 1860, 2018, 2042, 2049, 2077, 2108, 2121, 2123, 2134, 2154, 2279, 2290, 2294, 2333, 2348, 2423, 2434, 2438, 2551, 2581, 2655, 2671, 2733, 2743, 2773, 2778, 2826, 2834, 2847, 2892, 2942, 2956, 2964, 3003, 3060, 3073, 3081, 3099, 3304, 3313, 3317, 3372, 3380, 3393, 3425, 3435, 3518, 3523, 3531, 3535, 3550, 3579, 3694, 3703, 3707, 3847, 3851, 3875, 3884, 3888, 3894, 3908, 3930
<code>\bool_if:nTF</code>	1358, 1375, 2162, 2592, 2627, 2691, 3120, 4027
<code>\bool_if_p:N</code>	240, 254, 1704, 1705, 1713, 1714, 1840, 1870, 1871, 1876, 1877, 2145, 2188, 2189, 2213, 2222, 2223, 2235, 2251, 2410, 2411, 2448, 2449, 2865, 2878, 2880, 3127, 3128
<code>\bool_lazy_all:nTF</code>	238, 252, 1838, 2211, 2220, 2233, 2249, 2863, 2876
<code>\bool_lazy_and:nnTF</code>	218, 227, 1703, 1712, 1869, 1875, 2144, 2187, 2409
<code>\bool_lazy_or:nnTF</code>	1755, 1763, 2447, 3126
<code>\bool_new:N</code>	25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88, 93, 94, 97, 115, 117, 119, 122, 123, 132, 133, 134, 142, 143, 157, 168, 170
<code>\bool_not_p:n</code>	219, 228, 2146, 2238, 2253, 2866, 2867, 2879
<code>\bool_set_eq:NN</code>	2559, 2607, 3468, 3801
<code>\bool_set_false:N</code>	386, 1802, 1803, 2979, 3011, 3093, 3158, 3176, 3420, 3465, 3751, 3798
<code>\bool_set_true:N</code>	245, 259, 368, 372, 478, 796, 1389, 1394, 1660, 1775, 1776, 1981, 1988, 2555, 2585, 2603, 2615, 2809, 2872, 2885, 2911, 3008, 3035, 3293, 3362, 3474, 3481, 3482, 3683, 3807, 3814, 3815
box commands:	
<code>\box_dp:N</code>	1077, 1081, 1085, 1096, 1100, 1111, 1120, 1126, 1136, 1149, 1155, 1161, 1192, 1193, 1194, 1197, 1207, 1211, 1220, 1227, 1232, 1240, 1269, 1270, 1273, 1280, 1293, 1301, 1307, 1315, 3188
<code>\box_new:N</code>	58, 163
<code>\box_set_wd:Nn</code>	3571, 3922
<code>\box_use:N</code>	3578, 3929
<code>\box_wd:N</code>	444
C	
<code>\c</code>	202, 203, 696, 698, 710, 712
<code>\cB</code>	203
<code>\cE</code>	203
<code>\centering</code>	1360, 1377, 3209, 3319, 3709
<code>check-ans</code>	1794
Document class:	
<code>article</code>	38
clist commands:	
<code>\clist_const:Nn</code>	175
<code>\clist_map_function:nN</code>	3196
<code>\clist_map_inline:Nn</code>	484, 738, 801, 867, 882, 963, 1399
<code>\clist_map_inline:nn</code>	36, 47, 66, 72, 84, 96, 121, 151, 174, 509, 526, 806, 978, 1505, 1749, 1808, 1958, 1976, 1997, 2208, 2342, 2509, 2720, 2723, 2750, 2760, 2763, 2783
<code>\columnbreak</code>	67
<code>\columnbreak</code>	2148
<code>columns</code>	947
<code>columns*</code>	1977
<code>columns-sep</code>	947
<code>columns-sep*</code>	1977
<code>\columnsep</code>	83, 86
<code>\columnsep</code>	2936, 3057
<code>\columnseprule</code>	83, 86
<code>\columnseprule</code>	2940, 3059
Commands provide by enumext:	
<code>\anskey</code>	24, 25, 59, 60, 64, 66, 68-70, 72, 81, 94, 104, 105, 109
<code>\anspic*</code>	24, 25, 62, 70, 71, 87-89, 104, 105
<code>\anspic</code>	64, 87-89, 109
<code>\getkeyans</code>	64, 104, 110
<code>\item*</code>	24, 25, 62, 64, 70, 71, 75, 76, 95, 102, 104, 105
<code>\itemwidth</code>	90, 98
<code>\item</code>	74, 76, 90, 94-96, 98, 101
<code>\miniright</code>	24, 43, 50, 51, 82, 84-86, 110
<code>\printkeyans</code>	25, 64, 105

\setenumext 25, 106–108

Counters defined by **enumext**:

enumXiii 23, 32

enumXii 23, 32

enumXiv 23, 32

enumXi 23, 32

enumXviii 23, 32

enumXvii 23, 32, 96

enumXvi 23, 32

enumXv 23, 32

cs commands:

\cs_generate_variant:Nn 446, 462, 702, 718, 2010, 2015, 2095, 2710, 3198

\cs_if_exist:NTF 416

\cs_new:Nn 188

\cs_new:Npn 206, 1506, 1515, 1524

\cs_new_eq:NN 352, 353, 354, 358, 359, 391, 392, 395, 396

\cs_new_protected:Nn . 198, 212, 236, 267, 294, 306, 313, 320, 330, 363, 567, 630, 682, 883, 887, 891, 895, 899, 903, 907, 911, 915, 919, 923, 927, 931, 935, 939, 943, 979, 991, 1015, 1032, 1043, 1067, 1142, 1166, 1183, 1245, 1262, 1284, 1319, 1325, 1400, 1414, 1428, 1439, 1450, 1461, 1472, 1483, 1568, 1671, 1684, 1701, 1722, 1773, 1813, 1833, 1867, 1882, 1891, 1896, 1901, 2016, 2040, 2047, 2075, 2082, 2199, 2331, 2346, 2374, 2407, 2443, 2455, 2463, 2514, 2518, 2537, 2588, 2623, 2639, 2649, 2665, 2803, 2861, 2890, 2897, 2920, 2950, 2962, 3001, 3025, 3043, 3068, 3079, 3116, 3160, 3174, 3194, 3199, 3215, 3283, 3302, 3354, 3407, 3414, 3423, 3433, 3450, 3590, 3605, 3673, 3692, 3742, 3764, 3770, 3783, 3839, 3941

\cs_new_protected:Npn 180, 184, 399, 414, 431, 441, 447, 547, 592, 664, 689, 703, 1347, 1366, 1536, 1555, 1625, 1658, 1750, 1906, 2002, 2011, 2131, 2276, 2288, 2310, 2384, 2428, 2547, 2565, 2599, 2611, 2679, 2713, 2753, 2812, 2832, 3021, 3169, 3234, 3365, 3378, 3456, 3463, 3479, 3487, 3492, 3504, 3624, 3757, 3789, 3796, 3812, 3820, 3834, 3960, 3973, 4020, 4098, 4110

\cs_new_protected_nopar:Nn . . . 3443, 3566, 3776, 3917

\cs_new_protected_nopar:Npn 3510, 3867

\cs_set:Nn 2281

\cs_set:Npn 2209, 2247, 3966

\cs_set_eq:NN . . . 3344, 3345, 3512, 3732, 3733, 3869

\cs_set_protected:Nn 807, 823, 835, 847

\cs_set_protected:Npn . 32, 41, 59, 67, 79, 85, 111, 147, 155, 463, 485, 514, 527, 574, 719, 739, 783, 802, 859, 868, 947, 964, 1383, 1494, 1741, 1794, 1923, 1959, 1977, 2201, 2335, 2498, 2711, 2751

\cs_to_str:N 433, 456

D

\d 194

\DeclareDocumentEnvironment 1060

dim commands:

\dim_abs:n 2684, 2689

\dim_add:Nn 3179

\dim_compare:nNnTF . 809, 825, 837, 849, 1349, 1368, 2681, 2686, 2692, 2698, 2700, 2702, 2902, 2925, 3029, 3047, 3171, 3217, 3285, 3607, 3675

\dim_compare:nTF 2172

\dim_gset_eq:NN 3294, 3684

\dim_gzero:N 3328, 3718

\dim_new:N . 55, 62, 63, 64, 81, 118, 128, 164, 165, 171

\dim_set:Nn . . . 444, 797, 1989, 2579, 2684, 2689, 2691, 2694, 2695, 2699, 2701, 2704, 2705, 2707, 2905, 2928, 3031, 3049, 3201, 3219, 3226, 3269, 3287, 3506, 3609, 3616, 3659, 3677

\dim_set_eq:NN 534, 581, 652, 656, 2574, 2722, 2762, 2851, 2936, 3057, 3276, 3279, 3280, 3397, 3497, 3666, 3669, 3670

\dim_use:N 810, 818, 1350, 1356, 2085, 2088, 2093, 2644, 2646, 2903, 2908, 2909, 2916, 2926, 2930, 2931, 2933

\dim_zero:N 2940, 3059, 3180, 3181, 3182

\dim_zero_new:N 3232, 3622

\c_zero_dim 812, 826, 838, 850, 1350, 1368, 2174, 2681, 2686, 2692, 2699, 2903, 2926, 3029, 3047, 3217, 3285, 3607, 3675

E

\end . . . 1353, 1371, 2044, 2079, 2955, 2972, 3072, 3089, 3306, 3322, 3696, 3712, 4031, 4036

\endlist 30

\endlist 353

\endlrbox 3569, 3920

\endminipage 30

\endminipage 359

enumext 5, 2784

enumext internal commands:

\l__enumext__check_start_line_env_tl . . . 28

\l__enumext__ref_the_count_tl 34

\l__enumext__resume_name_tl 55

__enumext_add_pre_parsep: . . . 44, 989, 991, 991

__enumext_after_args_exec: . 41, 883, 895, 2796

__enumext_after_args_exec_v: . 41, 42, 899, 911, 2994

__enumext_after_args_exec_vii: . . . 915, 939

__enumext_after_args_exec_viii: 943

__enumext_after_env:nn . 29, 61, 84, 97, 184, 184, 2982, 3311, 3604, 3701

__enumext_after_hyperref: . . . 30, 361, 363, 363

__enumext_after_list: . . 84, 94, 100, 2801, 2962, 2962

\l__enumext_after_list_args_v_tl 913

\l__enumext_after_list_args_vii_tl 941, 3560

\l__enumext_after_list_args_viii_tl 945, 3904

__enumext_after_list_v: . . 86, 2999, 3079, 3079

__enumext_after_list_vii: . . . 3352, 3414, 3414

__enumext_after_list_viii: . . 3740, 3770, 3770

__enumext_after_star_env:nn 91

__enumext_after_stop_list: . . . 41, 42, 883, 891, 2977

__enumext_after_stop_list_v: 41, 899, 907, 3094

\l__enumext_after_stop_list_v_tl 909

__enumext_after_stop_list_vii: 915, 931, 3417

\l__enumext_after_stop_list_vii_tl . . . 933

__enumext_after_stop_list_viii: . 935, 3773

\l__enumext_after_stop_list_viii_tl . . . 937

\l__enumext_align_label_vii_str . . 3552, 3556

\l__enumext_align_label_viii_str . 3896, 3900

\l__enumext_align_label_X_str 155

\c__enumext_all_envs_clist . . 175, 484, 738, 801, 867, 882, 963, 1399

\c__enumext_all_families_seq . . 106, 4060, 4069, 4091

__enumext_anskey_wrapper:n 1927, 2286

__enumext_at_begin_document:n . . 30, 180, 180, 350, 356

__enumext_before_args_exec: 41, 883, 883, 2900

`__enumext_before_args_exec_v`: 41, 42, 899, 899, 3028
`__enumext_before_args_exec_vii`: .. 915, 915, 3411
`__enumext_before_args_exec_viii`: 919, 3767
`__enumext_before_keys_exec`: 41, 883, 887, 2794
`__enumext_before_keys_exec_v`: .. 41, 899, 903, 2992
`__enumext_before_keys_exec_vii` 915
`__enumext_before_keys_exec_vii`: 42, 923, 3340
`__enumext_before_keys_exec_viii`: .. 42, 927, 3728
`__enumext_before_list`: ... 82, 2788, 2897, 2897
`__enumext_before_list_v`: . 85, 2987, 3025, 3025
`__enumext_before_list_vii`: 93, 3335, 3407, 3407
`__enumext_before_list_viii`: .. 100, 3724, 3764, 3764
`\l__enumext_before_no_starred_key_v_tl` 905
`\l__enumext_before_no_starred_key_vii_-tl` 925
`\l__enumext_before_no_starred_key_viii_-tl` 929
`\l__enumext_before_starred_key_v_tl` ... 901
`\l__enumext_before_starred_key_vii_tl` . 917
`\l__enumext_before_starred_key_viii_tl` 921
`__enumext_calc_hspace:NNNNNN` 78, 2679, 2679, 2710, 2715, 2755
`__enumext_check_ans`: .. 60, 82, 1813, 1813, 2901, 3410
`\g__enumext_check_ans_bool` ... 61, 132, 299, 342, 1873, 1879
`\l__enumext_check_ans_bool` .. 60, 61, 74, 75, 132, 1798, 1803, 1815, 1870, 1876, 2123, 2423, 2551, 2581, 3523
`\g__enumext_check_ans_item_tl` 72
`__enumext_check_ans_level`: . 60, 61, 1813, 1829, 1833
`__enumext_check_ans_msg_greater`: 1888, 1901
`__enumext_check_ans_msg_less`: ... 1886, 1891
`__enumext_check_ans_msg_same_ok`: 1887, 1896
`__enumext_check_ans_show`: . 61, 344, 1882, 1882
`\g__enumext_check_ans_show_bool` 84
`__enumext_check_ans_to_hook`: . 61, 1867, 1867, 2976, 3418
`__enumext_check_starred_cmd:n` 28, 62, 72, 1906, 1906, 2997, 3155, 3738
`\g__enumext_check_starred_cmd_int` 132, 1909, 1915, 1920, 2621, 3125, 3846
`\l__enumext_check_start_line_env_tl` 132, 273, 280, 287, 1912, 1918, 1921
`\l__enumext_columns_sep_v_dim` 3047, 3049, 3057
`\l__enumext_columns_sep_vii_dim` .. 3217, 3219, 3228, 3273, 3399, 3588
`\l__enumext_columns_sep_viii_dim` . 3607, 3609, 3618, 3663, 3939
`\l__enumext_columns_v_int` 1188, 3045, 3053, 3065, 3070
`\l__enumext_columns_vii_int` .. 3222, 3225, 3229, 3237, 3241, 3244, 3250, 3256, 3260, 3386, 3583, 3594
`\l__enumext_columns_viii_int` . 3612, 3615, 3619, 3627, 3631, 3634, 3640, 3646, 3650, 3934, 3945
`\g__enumext_count_item_with_ans_int` 66
`\l__enumext_counter_i_tl` 32, 423
`\l__enumext_counter_ii_tl` 32, 424
`\l__enumext_counter_iii_tl` 32, 425
`\l__enumext_counter_iv_tl` 32, 426
`\c__enumext_counter_style_tl` 27, 37, 200
`\g__enumext_counter_styles_tl` . 23, 32, 55, 434, 452
`\l__enumext_counter_v_tl` 32, 427, 672
`\l__enumext_counter_vi_tl` 32, 428
`\l__enumext_counter_vii_tl` 32, 429, 602
`\l__enumext_counter_viii_tl` 32, 430, 619
`\l__enumext_current_widest_dim` 23, 55, 458, 535, 582, 653, 657
`__enumext_default_item:n` ... 2547, 2547, 2596
`__enumext_define_counters:Nn` 23, 414, 414, 423, 424, 425, 426, 427, 428, 429, 430
`__enumext_endminipage`: 30, 356, 359, 1066, 3211, 3568, 3919
`\g__enumext_envir_name_tl` 137, 246, 260, 304, 339, 1759, 1768, 1894, 1899, 1904
`__enumext_execute_after_env`: 29, 320, 330, 2982, 3604
`__enumext_fake_item`: 807, 807, 2742
`\l__enumext_fake_item_indent_v_dim` 826, 831
`\l__enumext_fake_item_indent_v_tl` 828, 2604, 2608, 2616
`\l__enumext_fake_item_indent_vii_dim` 838, 843
`\l__enumext_fake_item_indent_vii_tl` 840, 3564
`\l__enumext_fake_item_indent_viii_dim` . 850, 855, 3911
`\l__enumext_fake_item_indent_viii_tl` .. 852, 3910, 3914
`\l__enumext_fake_item_indent_X_tl` 85
`__enumext_fake_item_vii`: 807, 835, 2772
`__enumext_fake_item_viii`: 807, 847, 2777
`__enumext_filter_series:n` 54, 1506, 1506, 1548, 1560, 1565
`__enumext_filter_series_key:n` 54, 1506, 1511, 1515
`__enumext_filter_series_pair:nn` .. 54, 1506, 1512, 1524
`\g__enumext_footnote_arg_seq` . 152, 2520, 2533, 2543
`\g__enumext_footnote_int` . 152, 2527, 2530, 2532, 2534
`\g__enumext_footnote_int_seq` . 152, 2521, 2534, 2539, 2542
`__enumext_footnotes_key_bool` 30
`\l__enumext_footnotes_key_bool` 25, 31, 96, 142, 372, 377, 386, 3531, 3579, 3884, 3930
`__enumext_footnotetext:nn` ... 2514, 2514, 2544
`__enumext_getkeyans:nn` .. 105, 3969, 3973, 3973
`__enumext_getkeyans_aux:n` 104, 3957, 3960, 3960
`\l__enumext_hyperref_bool` . 25, 30, 31, 142, 368, 389, 406, 2189, 2411, 3518, 3875
`__enumext_hypertarget:nn` 31, 363, 391, 395, 411
`__enumext_if_is_int:n` 192
`__enumext_if_is_int:nTF` 192, 691, 705
`__enumext_is_not_nested`: 27, 80, 212, 212, 2805, 3356
`__enumext_is_on_first_level`: . 28, 80, 212, 236, 2810, 3363
`\g__enumext_item_anskey_int` .. 72, 132, 297, 317, 318, 326, 2125, 2425
`__enumext_item_answer_diff`: 320, 336
`\g__enumext_item_answer_diff_int` 141, 298, 322, 1884

```

\l__enumext_item_column_pos_vii_int 94, 3244,
    3250, 3256, 3260, 3267, 3446, 3583, 3586
\l__enumext_item_column_pos_viii_int .. 101,
    3634, 3640, 3646, 3650, 3657, 3779, 3934, 3937
l__enumext_item_column_pos_X_int ..... 155
\g__enumext_item_count_all_vii_int 94, 3268,
    3447, 3594, 3601
\g__enumext_item_count_all_viii_int 101, 3658,
    3780, 3945, 3952
\g__enumext_item_count_all_X_int ..... 155
\g__enumext_item_number_int .. 61, 132, 296, 316,
    318, 326, 1844, 1848, 1851, 1854, 1862, 2553, 2583,
    3525
\__enumext_item_peek_args_vii: 94, 3448, 3450,
    3450
\__enumext_item_peek_args_viii: .. 101, 3781,
    3783, 3783
\__enumext_item_starred: .. 77, 2639, 2639, 2657
\l__enumext_item_starred_vii_bool 3465, 3481,
    3535
\l__enumext_item_starred_viii_bool 3798, 3814,
    3888, 3908
\l__enumext_item_starred_X_bool ..... 155
\__enumext_item_std:w 30, 74-76, 89, 350, 354, 2556,
    2562, 2586, 2604, 2608, 2616, 3192
\g__enumext_item_symbol_aux_vii_tl 3489, 3537,
    3540, 3544, 3546
\g__enumext_item_symbol_aux_X_tl ..... 155
\l__enumext_item_symbol_sep_vii_dim .. 3498,
    3506, 3543, 3545
\g__enumext_item_symbol_tl 23, 75, 48, 2571, 2645,
    2662
\l__enumext_item_symbol_vii_tl ..... 3540
\l__enumext_item_text_vii_box 3530, 3571, 3578
\l__enumext_item_text_viii_box 3883, 3922, 3929
\l__enumext_item_text_X_box ..... 155
\l__enumext_item_width_vii_dim ... 3226, 3271,
    3279, 3280
\l__enumext_item_width_viii_dim .. 3616, 3661,
    3669, 3670
\l__enumext_item_width_X_dim ..... 155
\l__enumext_itemindent_X_dim ..... 59
\l__enumext_itemsep_vii_skip ..... 3600
\l__enumext_itemsep_viii_skip ..... 3951
\l__enumext_joined_item_aux_vii_int .. 3265,
    3266, 3267, 3268, 3274
\l__enumext_joined_item_aux_viii_int . 3655,
    3656, 3657, 3658, 3664
\l__enumext_joined_item_aux_X_int .... 155
\__enumext_joined_item_vii:w 94, 95, 3453, 3454,
    3456, 3456
\l__enumext_joined_item_vii_int .. 3236, 3237,
    3240, 3242, 3248, 3253, 3258, 3263, 3265, 3271
\__enumext_joined_item_viii:w . 101, 3786, 3787,
    3789, 3789
\l__enumext_joined_item_viii_int . 3626, 3627,
    3630, 3632, 3638, 3643, 3648, 3653, 3655, 3661
\l__enumext_joined_item_X_int ..... 155
\l__enumext_joined_width_vii_dim . 3269, 3276,
    3279, 3561, 3573
\l__enumext_joined_width_viii_dim 3659, 3666,
    3669, 3905, 3924
\l__enumext_joined_width_X_dim ..... 155
\__enumext_keyans_addto_prop:n 70, 2310, 2310,
    2618, 3122
\__enumext_keyans_addto_seq:n . 71, 2384, 2384,
    2620, 3124
\__enumext_keyans_addto_seq_link: 2384, 2405,
    2407, 3845
\__enumext_keyans_anspic_code:nnn 87, 88, 3113,
    3116, 3116
\__enumext_keyans_default_item:n .. 76, 2599,
    2599, 2635
\l__enumext_keyans_env_bool 20, 2866, 2879, 3008,
    3093
\__enumext_keyans_fake_item: .. 807, 823, 2732
\l__enumext_keyans_item_opt_tl . 102, 97, 2432,
    2445, 2451, 3830
\l__enumext_keyans_level_h_int .. 20, 612, 639,
    2362, 3744, 3745
\l__enumext_keyans_level_int .. 20, 1341, 2112,
    2357, 3007, 3012, 3107
\__enumext_keyans_make_label: 33, 77, 2665, 2665,
    2730
\__enumext_keyans_mini_addvspace: 48, 85, 1245,
    1245, 3037
\__enumext_keyans_mini_right_cmd:n 51, 1343,
    1366, 1366
\__enumext_keyans_mini_set_vskip: . 48, 1183,
    1183, 1247
\__enumext_keyans_multi_addvspace: 86, 1032,
    1043, 3062
\__enumext_keyans_multi_set_vskip: 44, 1032,
    1032, 1045
\__enumext_keyans_multicols_start: .. 85, 86,
    3041, 3043, 3043
\__enumext_keyans_multicols_stop: . 86, 1370,
    3068, 3068, 3092
\__enumext_keyans_parse_keys:n 2986, 3021, 3021
\l__enumext_keyans_pic_above_int . 127, 3202,
    3203, 3205
\l__enumext_keyans_pic_above_skip .. 89, 127,
    3146, 3186
\__enumext_keyans_pic_arg_two: 89, 3144, 3174,
    3174
\l__enumext_keyans_pic_below_int . 127, 3202,
    3203, 3206
\l__enumext_keyans_pic_body_seq .. 87-89, 127,
    3111, 3151, 3210
\__enumext_keyans_pic_do:n 89, 3151, 3153, 3194,
    3194, 3198
\l__enumext_keyans_pic_level_int .. 20, 1333,
    2116, 2313, 2352, 2387, 2465, 3162, 3163
\__enumext_keyans_pic_row:n 89, 3196, 3199, 3199
\__enumext_keyans_pic_safe_exec: .. 88, 3140,
    3160, 3160
\__enumext_keyans_pic_skip_abs:N .. 89, 3169,
    3169, 3185
\l__enumext_keyans_pic_width_dim . 127, 3201,
    3208
\__enumext_keyans_redefine_item: .. 76, 2623,
    2623, 2729
\__enumext_keyans_ref: ..... 37, 664, 682, 2731
\__enumext_keyans_ref:n ..... 36, 661, 664, 664
\__enumext_keyans_safe_exec: . 2985, 3001, 3001
\__enumext_keyans_save_start_line: . 28, 267,
    267, 3009, 3167, 3749
\__enumext_keyans_show_ans: .. 2428, 2436, 2455

```

```

\__enumext_keyans_show_item_opt: . 2428, 2443,
    2616, 3136, 3911
\__enumext_keyans_show_left:n . 76, 2428, 2428,
    2614, 3131
\__enumext_keyans_show_pos: .. 2428, 2440, 2463
\__enumext_keyans_starred_item:n .. 76, 2611,
    2611, 2631
\__enumext_keyans_store_ref: .. 70, 2331, 2331,
    2619, 3123, 3843
\__enumext_keyans_store_ref_aux_i: 70, 2331,
    2343, 2346
\__enumext_keyans_store_ref_aux_ii: 71, 2331,
    2372, 2374
\l__enumext_keyans_tmpa_tl .. 24, 97, 2613, 2617
\__enumext_keyans_wrapper_opt:n .. 1930, 2451
\l__enumext_label_copy_i_tl .. 2243, 2350, 2355,
    2360, 2365
\l__enumext_label_copy_v_tl ..... 2360
\l__enumext_label_copy_vi_tl ..... 2355
\l__enumext_label_copy_vii_tl 2218, 2229, 2260,
    2350
\l__enumext_label_copy_viii_tl ..... 2365
\l__enumext_label_copy_X_tl ..... 144
\l__enumext_label_fill_left_v_tl ..... 2669
\l__enumext_label_fill_left_X_tl ..... 85
\l__enumext_label_fill_right_v_tl .... 2676
\l__enumext_label_fill_right_X_tl ..... 85
\l__enumext_label_font_style_v_tl 2670, 3135
\l__enumext_label_font_style_vii_tl ... 3549
\l__enumext_label_font_style_viii_tl .. 3893
\l__enumext_label_i_tl ..... 527
\l__enumext_label_ii_tl ..... 527
\l__enumext_label_iii_tl ..... 527
\l__enumext_label_iv_tl ..... 527
\__enumext_label_style:Nnn 23, 32, 447, 447, 462,
    532, 579, 650, 654
\l__enumext_label_v_tl .. 70, 71, 647, 2318, 2392,
    2457, 2492, 2613, 2617, 2989, 3130, 3132
\l__enumext_label_vi_tl . 70, 71, 647, 2315, 2389,
    3130, 3132, 3136
\l__enumext_label_vii_tl . 574, 3476, 3501, 3508
\l__enumext_label_viii_tl 574, 3809, 3837, 3841
\l__enumext_label_width_by_box .. 55, 443, 444
\__enumext_label_width_by_box:Nn 32, 441, 441,
    446, 458, 715
\l__enumext_labelsep_i_dim ... 2460, 2495, 3849,
    3864
\l__enumext_labelsep_v_dim ..... 3052
\l__enumext_labelsep_vii_dim . 3221, 3230, 3272,
    3499, 3559, 3575
\l__enumext_labelsep_viii_dim 3611, 3620, 3662,
    3903, 3926
\l__enumext_labelwidth_i_dim . 2460, 2495, 3849,
    3864
\l__enumext_labelwidth_v_dim ..... 3052
\l__enumext_labelwidth_vii_dim ... 3221, 3229,
    3272, 3552, 3556, 3574
\l__enumext_labelwidth_viii_dim .. 3611, 3619,
    3662, 3896, 3900, 3925
\l__enumext_leftmargin_tmp_v_bool . 89, 3176
\l__enumext_leftmargin_tmp_X_bool ..... 59
\l__enumext_leftmargin_tmp_X_dim ..... 59
\l__enumext_leftmargin_X_dim ..... 59
\__enumext_level: 188, 188, 556, 559, 560, 569, 571,
    810, 814, 818, 885, 889, 893, 897, 981, 983, 985, 987,
    1020, 1022, 1024, 1026, 1030, 1070, 1073, 1092, 1101,
    1107, 1112, 1116, 1127, 1131, 1132, 1137, 1173, 1177,
    1350, 1356, 1403, 1405, 1407, 1410, 1417, 1419, 1421,
    1424, 2020, 2028, 2032, 2036, 2281, 2284, 2285, 2555,
    2556, 2560, 2561, 2562, 2569, 2571, 2575, 2576, 2579,
    2585, 2586, 2641, 2644, 2646, 2653, 2654, 2655, 2658,
    2661, 2791, 2793, 2834, 2839, 2840, 2841, 2843, 2847,
    2852, 2853, 2854, 2856, 2872, 2885, 2892, 2903, 2905,
    2908, 2909, 2911, 2916, 2923, 2926, 2928, 2930, 2931,
    2932, 2933, 2936, 2942, 2947, 2953, 2956, 2958, 2964
\l__enumext_level_h_int .. 20, 220, 242, 255, 595,
    632, 1841, 1857, 2237, 2254, 3357, 3358
\l__enumext_level_int . 80, 20, 190, 229, 241, 256,
    332, 993, 1144, 1337, 1835, 2214, 2224, 2230, 2236,
    2244, 2252, 2259, 2745, 2806, 2807, 2817, 2824, 2870,
    2883, 2938, 3016, 3103, 3427, 3437, 3752
\__enumext_list_arg_two_i: ..... 2711
\__enumext_list_arg_two_ii: ..... 2711
\__enumext_list_arg_two_iii: ..... 2711
\__enumext_list_arg_two_iv: ..... 2711
\__enumext_list_arg_two_v: . 76, 2711, 2991, 3177
\__enumext_list_arg_two_vii: ..... 2751, 3339
\__enumext_list_arg_two_viii: .... 2751, 3727
\l__enumext_listoffset_v_dim ..... 3054
\l__enumext_listparindent_vii_dim .... 3562
\l__enumext_listparindent_viii_dim ... 3906
\__enumext_log_answer_vars: .... 313, 313, 341
\__enumext_log_global_vars: .... 306, 306, 340
\__enumext_make_label: 33, 74, 75, 77, 2649, 2649,
    2740
\l__enumext_mark_answer_sym_tl . 65, 122, 1936,
    2090, 2296, 2467, 2480, 3853
\l__enumext_mark_position_str 122, 1940, 1941,
    1964, 1965, 2088
\l__enumext_mark_ref_sym_tl .. 122, 1950, 2194,
    2419
\__enumext_mini_addvspace: .. 47, 82, 1166, 1166,
    2913
\__enumext_mini_addvspace_vii: 50, 1319, 1319,
    3297
\__enumext_mini_addvspace_viii: 50, 1319, 1325,
    3687
\__enumext_mini_env* ..... 1060
\__enumext_mini_right_cmd:n . 50, 51, 1345, 1347,
    1347
\__enumext_mini_set_vskip: . 45, 1067, 1067, 1168
\__enumext_mini_set_vskip_vii: 49, 1262, 1262,
    1321
\__enumext_mini_set_vskip_viii: 49, 1262, 1284,
    1327
\__enumext_minipage:w .. 30, 356, 358, 1062, 3208,
    3561, 3905
\l__enumext_minipage_active_v_bool .. 85, 86,
    3035, 3060, 3073, 3081
\g__enumext_minipage_active_vii_bool ... 91,
    3308, 3313, 3325
\l__enumext_minipage_active_vii_bool . 3293,
    3304
\g__enumext_minipage_active_viii_bool 3698,
    3703, 3715
\l__enumext_minipage_active_viii_bool 3683,
    3694
\g__enumext_minipage_active_X_bool ... 155

```


`\l__enumext_minipage_active_X_bool` [73](#)
`\g__enumext_minipage_after_skip` [73](#), [1266](#), [1278](#),
[3323](#), [3713](#)
`\l__enumext_minipage_after_skip` [45](#), [47](#), [84](#), [86](#),
[73](#), [1083](#), [1098](#), [1118](#), [1134](#), [1149](#), [1155](#), [1161](#), [1175](#),
[1185](#), [1194](#), [1197](#), [1209](#), [1227](#), [1238](#), [1254](#), [1286](#), [1299](#),
[1313](#), [2973](#), [3090](#)
`\g__enumext_minipage_center_vii_bool` . [3317](#),
[3326](#)
`\g__enumext_minipage_center_viii_bool` [3707](#),
[3716](#)
`\g__enumext_minipage_center_X_bool` . . . [155](#)
`\l__enumext_minipage_hsep_v_dim` . . . [85](#), [3033](#)
`\l__enumext_minipage_hsep_vii_dim` [3291](#)
`\l__enumext_minipage_hsep_viii_dim` . . . [3681](#)
`\l__enumext_minipage_left_skip` [45](#), [85](#), [73](#), [1075](#),
[1090](#), [1109](#), [1124](#), [1171](#), [1181](#), [1186](#), [1192](#), [1201](#), [1218](#),
[1230](#), [1250](#), [1260](#), [1264](#), [1269](#), [1273](#), [1287](#), [1291](#), [1305](#),
[1323](#), [1329](#)
`\l__enumext_minipage_left_v_dim` [85](#), [3031](#), [3039](#)
`\l__enumext_minipage_left_vii_dim` [3287](#), [3299](#)
`\l__enumext_minipage_left_viii_dim` [3677](#), [3689](#)
`\l__enumext_minipage_left_X_dim` [73](#)
`\g__enumext_minipage_right_skip` [73](#), [1265](#), [1270](#),
[1274](#), [3316](#), [3706](#)
`\l__enumext_minipage_right_skip` . [45](#), [73](#), [1079](#),
[1094](#), [1114](#), [1129](#), [1187](#), [1193](#), [1205](#), [1223](#), [1234](#), [1288](#),
[1295](#), [1309](#), [1357](#), [1374](#)
`\l__enumext_minipage_right_v_dim` . . [85](#), [1368](#),
[1373](#), [3029](#), [3033](#)
`\g__enumext_minipage_right_vii_dim` [91](#), [3295](#),
[3315](#), [3328](#)
`\l__enumext_minipage_right_vii_dim` [91](#), [3285](#),
[3290](#), [3296](#)
`\g__enumext_minipage_right_viii_dim` . . [3685](#),
[3705](#), [3718](#)
`\l__enumext_minipage_right_viii_dim` . . [3675](#),
[3680](#), [3686](#)
`\g__enumext_minipage_right_X_dim` [155](#)
`\g__enumext_minipage_right_X_skip` [155](#)
`\g__enumext_minipage_stat_int` . [82](#), [85](#), [73](#), [1362](#),
[1379](#), [2912](#), [2966](#), [2971](#), [3036](#), [3083](#), [3088](#)
`\g__enumext_miniright_code_vii_tl` . [92](#), [3321](#),
[3327](#)
`\g__enumext_miniright_code_viii_tl` [3711](#), [3717](#)
`\g__enumext_miniright_code_X_tl` [155](#)
`__enumext_multi_addvspace:` . [44](#), [83](#), [1015](#), [1015](#),
[2944](#)
`__enumext_multi_set_vskip:` . [43](#), [979](#), [979](#), [1017](#)
`\l__enumext_multicols_above_ii_skip` . . . [998](#)
`\l__enumext_multicols_above_iii_skip` . . [1004](#)
`\l__enumext_multicols_above_iv_skip` . . [1010](#)
`\l__enumext_multicols_above_v_skip` [1034](#), [1048](#),
[1058](#)
`\l__enumext_multicols_above_X_skip` [67](#)
`\l__enumext_multicols_below_v_skip` [1038](#), [1052](#),
[3075](#)
`\l__enumext_multicols_below_X_skip` [67](#)
`__enumext_multicols_start:` [83](#), [2918](#), [2920](#), [2920](#)
`__enumext_multicols_stop:` [83](#), [1352](#), [2950](#), [2950](#),
[2975](#)
`__enumext_newlabel:nn` [26](#), [31](#), [69](#), [399](#), [399](#), [2270](#),
[2378](#)
`\l__enumext_newlabel_arg_one_tl` [26](#), [31](#), [69](#), [70](#),
[144](#), [2193](#), [2263](#), [2271](#), [2367](#), [2379](#), [2417](#)
`\l__enumext_newlabel_arg_two_tl` [26](#), [31](#), [68](#), [144](#),
[2217](#), [2227](#), [2241](#), [2257](#), [2272](#), [2354](#), [2359](#), [2364](#), [2380](#)
`__enumext_parse_keys:n` . . . [54](#), [2787](#), [2812](#), [2812](#)
`__enumext_parse_keys_vii:n` [54](#), [3334](#), [3365](#), [3365](#)
`__enumext_parse_keys_viii:n` . [3723](#), [3757](#), [3757](#)
`__enumext_parse_series:n` . . [54](#), [81](#), [1536](#), [1536](#),
[2820](#), [3371](#)
`__enumext_parse_store_keys:n` . [81](#), [2828](#), [2832](#),
[2832](#)
`__enumext_parse_store_keys_vii:n` . [93](#), [3374](#),
[3378](#), [3378](#)
`\l__enumext_parsep_i_skip` [996](#), [998](#), [1147](#), [1195](#)
`\l__enumext_parsep_ii_skip` . . . [1002](#), [1004](#), [1153](#)
`\l__enumext_parsep_iii_skip` . . [1008](#), [1010](#), [1159](#)
`\l__enumext_parsep_vii_skip` [3563](#)
`\l__enumext_parsep_viii_skip` [3907](#)
`\l__enumext_partopsep_v_skip` . [1050](#), [1054](#), [1221](#),
[1225](#), [1232](#), [1236](#), [1252](#), [1256](#)
`\l__enumext_partopsep_viii_skip` [1297](#)
`__enumext_phantomsection:` [31](#), [363](#), [392](#), [396](#), [412](#)
`__enumext_print_footnote:` . . . [2514](#), [2537](#), [3581](#),
[3932](#)
`__enumext_print_keyans_box:NN` [65](#), [2082](#), [2082](#),
[2095](#), [2283](#), [2459](#), [2494](#), [3849](#), [3864](#)
`\l__enumext_print_keyans_i_tl` [3983](#), [4012](#)
`\l__enumext_print_keyans_ii_tl` . . . [3988](#), [4013](#)
`\l__enumext_print_keyans_iii_tl` . . [3993](#), [4014](#)
`\l__enumext_print_keyans_iv_tl` . . . [3998](#), [4015](#)
`\l__enumext_print_keyans_vii_tl` . . [4003](#), [4016](#)
`\l__enumext_print_keyans_X_tl` [111](#)
`__enumext_printkeyans:nnn` [105](#), [4017](#), [4020](#), [4020](#)
`__enumext_redefine_item:` . [75](#), [2588](#), [2588](#), [2739](#)
`\l__enumext_ref_key_arg_tl` [34](#), [37](#), [203](#), [549](#), [550](#),
[563](#), [594](#), [597](#), [608](#), [614](#), [625](#), [666](#), [667](#), [678](#)
`\l__enumext_ref_the_count_tl` . [34](#), [37](#), [556](#), [559](#),
[562](#), [602](#), [604](#), [607](#), [619](#), [621](#), [624](#), [672](#), [674](#), [677](#)
`__enumext_regex_counter_style:` . . [27](#), [34](#), [198](#),
[198](#), [557](#), [603](#), [620](#), [673](#)
`__enumext_register_counter_style:Nn` . . [431](#),
[431](#), [436](#), [437](#), [438](#), [439](#), [440](#)
`__enumext_remove_extra_parsep_vii:` . . [3349](#),
[3590](#), [3590](#)
`__enumext_remove_extra_parsep_viii:` . [3737](#),
[3941](#), [3941](#)
`__enumext_renew_footnote:` . . . [2514](#), [2518](#), [3533](#),
[3886](#)
`\l__enumext_renew_the_count_v_tl` [675](#), [684](#), [686](#)
`\l__enumext_renew_the_count_vii_tl` [605](#), [634](#),
[636](#)
`\l__enumext_renew_the_count_viii_tl` [622](#), [641](#),
[643](#)
`\l__enumext_renew_the_count_X_tl` [37](#)
`__enumext_reset_global_vars:` . . [294](#), [294](#), [347](#)
`\l__enumext_resume_active_bool` [54](#), [57](#), [48](#), [1540](#),
[1660](#)
`__enumext_resume_counter:` . . [56](#), [57](#), [1658](#), [1664](#),
[1671](#)
`__enumext_resume_counter:n` . [54](#), [57](#), [1629](#), [1634](#),
[1658](#), [1658](#), [1728](#), [1736](#)
`__enumext_resume_counter_save_ans:` [57](#), [1658](#),
[1669](#), [1701](#)
`__enumext_resume_counter_series:` . [57](#), [1658](#),
[1667](#), [1684](#)

```

\g__enumext_resume_int ... 48, 1581, 1675, 1676
\__enumext_resume_last:n 54, 55, 1536, 1542, 1555
\l__enumext_resume_name_tl 48, 1577, 1585, 1588,
1604, 1612, 1615, 1661, 1662, 1690, 1697
\__enumext_resume_save_counter: 55, 1568, 1568,
2980, 3421
\__enumext_resume_series:n . 56, 1500, 1625, 1625
\__enumext_resume_starred: . 58, 1501, 1722, 1722
\g__enumext_resume_vii_int .. 94, 48, 1608, 1680,
1681
\__enumext_safe_exec: . . . . . 80, 2786, 2803, 2803
\__enumext_safe_exec_vii: . . . 3333, 3354, 3354
\__enumext_safe_exec_viii: . . . 3722, 3742, 3742
\l__enumext_series_name_tl . . . . . 57
\l__enumext_series_str . 55, 81, 1498, 1538, 1546,
1547, 1549, 1551, 1572, 1575, 1579, 1599, 1602, 1606,
2816, 3369
\__enumext_set_error:nn . . . . . 4098, 4108, 4110
\__enumext_set_parse:n . . . . . 4081, 4098, 4098
\l__enumext_setkey_tmpa_int ... 106, 4074, 4078
\l__enumext_setkey_tmpa_seq .. 106, 4072, 4082,
4088, 4090, 4092, 4105
\l__enumext_setkey_tmpa_tl ... 106, 4080, 4084
\l__enumext_setkey_tmpb_seq .. 106, 4073, 4076,
4080, 4081
\l__enumext_setkey_tmpb_tl 106, 4100, 4102, 4103
\l__enumext_show_answer_bool . 122, 1944, 1968,
2290, 2434, 2448, 3127, 3847
\__enumext_show_length:nnn .. 40, 206, 206, 4174,
4175, 4176, 4177, 4178, 4179, 4180, 4181, 4182, 4183,
4189, 4190, 4191, 4192, 4193, 4194, 4195, 4196, 4197,
4198
\l__enumext_show_position_bool 122, 1947, 1971,
2294, 2438, 2449, 3128, 3851
\g__enumext_standar_bool . 27, 20, 219, 222, 240,
300, 1570, 1635, 1647, 1673, 1686, 1724, 1860, 1871
\l__enumext_standar_bool 84, 20, 2222, 2235, 2251,
2809, 2979
\l__enumext_standar_first_level_bool . 80, 20,
245, 1557, 1704, 1756, 1764, 1819
\__enumext_standar_ref: . . . . 35, 547, 567, 2741
\__enumext_standar_ref:n . . . . 34, 539, 547, 547
\g__enumext_standar_series_tl . 48, 1559, 1560,
1726, 1729
\g__enumext_standard_bool . . . . . 80
\l__enumext_standard_bool . . . . . 80
\l__enumext_standard_first_level_bool .. 28
\__enumext_standard_item_vii:w 95, 3461, 3463,
3463
\__enumext_standard_item_viii:w .. 101, 3794,
3796, 3796
\g__enumext_starred_bool 27, 92, 94, 20, 228, 231,
254, 301, 1597, 1640, 1651, 1678, 1693, 1732, 1840,
1877, 2213, 2223, 2253, 2348, 2867, 2880, 3329
\l__enumext_starred_bool . 92, 94, 20, 2146, 2154,
2238, 2279, 3362, 3420
\__enumext_starred_columns_set_vii: .. 3215,
3215, 3342
\__enumext_starred_columns_set_viii: . 3605,
3605, 3730
\l__enumext_starred_first_level_bool 20, 259,
1562, 1713, 1757, 1765, 1823
\__enumext_starred_item:nn ... 2565, 2565, 2594
\__enumext_starred_item_exec: . 102, 3839, 3839,
3890
\__enumext_starred_item_vii:w . 95, 3460, 3479,
3479
\__enumext_starred_item_vii_aux_i:w .. 3479,
3484, 3487
\__enumext_starred_item_vii_aux_ii:w . 3479,
3485, 3490, 3492
\__enumext_starred_item_vii_aux_iii:w 3479,
3495, 3504
\__enumext_starred_item_viii:w 101, 102, 3793,
3812, 3812
\__enumext_starred_item_viii_aux_i:w .. 102,
3812, 3817, 3820
\__enumext_starred_item_viii_aux_ii:w . 102,
3812, 3818, 3832, 3834
\__enumext_starred_joined_item_vii:n . 90, 95,
3234, 3234, 3458
\__enumext_starred_joined_item_viii:n .. 98,
101, 3624, 3624, 3791
\__enumext_starred_ref: . . . . 36, 592, 630, 2769
\__enumext_starred_ref:n . . . . 35, 586, 592, 592
\g__enumext_starred_series_tl . 48, 1564, 1565,
1734, 1737
\__enumext_start_from:NNn 37, 689, 689, 702, 724
\l__enumext_start_i_int . . . . . 1676, 1688, 1707
\__enumext_start_item_tmp_vii: 92, 3345, 3443,
3443
\__enumext_start_item_tmp_viii: 99, 3733, 3776,
3776
\__enumext_start_item_vii:w . 95, 96, 3471, 3476,
3501, 3508, 3510, 3510
\__enumext_start_item_viii:w .. 101, 3804, 3809,
3837, 3867, 3867
\g__enumext_start_line_tl 28, 132, 247, 261, 303,
1894, 1899, 1904
\__enumext_start_list:nn 30, 77, 89, 350, 352, 2790,
2988, 3141, 3337, 3725
\__enumext_start_mini_vii: . 93, 3283, 3283, 3412
\__enumext_start_mini_viii: . . . 100, 3673, 3673,
3768
\__enumext_start_store_level: . 81, 2789, 2861,
2861
\__enumext_start_store_level_vii: . 94, 3336,
3423, 3423
\l__enumext_start_vii_int ... 1681, 1695, 1716
\l__enumext_start_X_int . . . . . 85, 719
\__enumext_stop_item_tmp_vii: . 92, 94, 96, 3344,
3348, 3445, 3512
\__enumext_stop_item_tmp_viii: . 99, 101, 3732,
3736, 3778, 3869
\__enumext_stop_item_vii: 96, 97, 3512, 3566, 3566
\__enumext_stop_item_viii: 103, 3869, 3917, 3917
\__enumext_stop_list: .. 30, 350, 353, 2799, 2998,
3154, 3350, 3739
\__enumext_stop_mini_vii: 91, 94, 3302, 3302, 3416
\__enumext_stop_mini_viii: 100, 3673, 3692, 3772
\__enumext_stop_store_level: .. 81, 2800, 2861,
2890
\__enumext_stop_store_level_vii: .. 94, 3351,
3423, 3433
\l__enumext_store_active_bool 24, 59, 81, 93, 97,
1705, 1714, 1775, 2108, 2826, 2865, 2878, 3003, 3011,
3099, 3158, 3372, 3425, 3435, 3751

```

`__enumext_store_addto_prop:n` 64, 70, 2002, 2002, 2010, 2133, 2329, 3842
`__enumext_store_addto_seq:n` 64, 72, 2011, 2011, 2015, 2022, 2036, 2044, 2053, 2071, 2079, 2197, 2422
`\l__enumext_store_ans_bool` . 59, 132, 1776, 1802, 2018, 2042, 2049, 2077, 2121
`\l__enumext_store_anskey_arg_tl` . . 24, 67, 97, 2139, 2148, 2150, 2156, 2164, 2167, 2177, 2182, 2185, 2191, 2197
`__enumext_store_anskey_code:nnnn` . 66, 2127, 2131, 2131
`__enumext_store_anskey_show_left:n` 69, 2138, 2288, 2288
`__enumext_store_anskey_show_wrap:n` 69, 2276, 2276, 2292, 2307
`\l__enumext_store_columns_break_bool` . 2102, 2145
`\l__enumext_store_columns_join_int` 97, 2153, 2158
`\l__enumext_store_columns_sep_vii_bool` 3393
`\l__enumext_store_columns_sep_vii_dim` 3398, 3402
`\l__enumext_store_columns_sep_X_bool` . . 111
`\l__enumext_store_columns_sep_X_dim` . . . 111
`\l__enumext_store_columns_vii_bool` . . . 3380
`\l__enumext_store_columns_vii_int` 3385, 3389
`\l__enumext_store_columns_X_bool` 111
`\l__enumext_store_columns_X_int` 111
`__enumext_store_internal_ref:` . . 66, 68, 2136, 2199, 2199
`\l__enumext_store_item_symbol_sep_dim` 2100, 2174, 2179
`\l__enumext_store_item_symbol_tl` . 2098, 2165, 2169
`\l__enumext_store_keyans_item_opt_sep_-tl` 1933, 2323, 2325, 2396, 2400, 3825, 3827
`\l__enumext_store_keyans_item_opt_tl` . . . 97
`\l__enumext_store_keyans_label_tl` 24, 70-72, 102, 97, 2312, 2315, 2318, 2325, 2327, 2329, 2386, 2389, 2392, 2398, 2403, 2413, 2422, 3822, 3827, 3828, 3841, 3842, 3844
`__enumext_store_level_close:` . 64, 2016, 2040, 2894
`__enumext_store_level_close_vii:` 2047, 2075, 3439
`__enumext_store_level_open:` . . 63, 64, 81, 2016, 2016, 2873, 2886
`__enumext_store_level_open_vii:` . . 93, 2047, 2047, 3429
`\g__enumext_store_name_tl` . 24, 84, 97, 302, 309, 310, 311, 334, 339, 1777, 1894, 1899, 1904
`\l__enumext_store_name_tl` . 24, 59, 60, 97, 1591, 1594, 1618, 1621, 1709, 1718, 1752, 1753, 1768, 1777, 1778, 1780, 1781, 1783, 1785, 1786, 1788, 1790, 1791, 1817, 2004, 2006, 2013, 2265, 2266, 2302, 2369, 2370, 2473, 2486, 3859
`\l__enumext_store_opt_vii_tl` . 2051, 2061, 2067, 2071, 3387, 3400
`\l__enumext_store_opt_X_tl` 111
`\l__enumext_store_ref_key_bool` 66, 1953, 2134, 2188, 2333, 2410
`\l__enumext_store_upper_level_X_bool` . . 111
`\l__enumext_store_write_aux_file_tl` 26, 69, 71, 144, 2268, 2274, 2376, 2382
`__enumext_storing_exec:` . . 59, 1750, 1769, 1773
`__enumext_storing_set:n` . . 59, 1745, 1750, 1750
`\l__enumext_the_counter_v_tl` 674
`\l__enumext_the_counter_vii_tl` 604
`\l__enumext_the_counter_viii_tl` 621
`\l__enumext_the_counter_X_tl` 37
`__enumext_tmp:n` 32, 36, 41, 47, 59, 66, 67, 72, 79, 84, 85, 96, 111, 121, 147, 151, 155, 174, 802, 806, 1494, 1505, 1741, 1749, 1794, 1812, 1923, 1958, 1959, 1976, 2201, 2208, 2209, 2230, 2244, 2247, 2259, 2335, 2342, 2711, 2750, 2751, 2783
`__enumext_tmp:nn` 463, 484, 485, 513, 514, 526, 719, 738, 783, 801, 859, 867, 868, 882, 947, 963, 964, 978, 1383, 1399, 1977, 2001, 2498, 2513
`__enumext_tmp:nnn` 527, 543, 544, 545, 546, 574, 590, 591
`__enumext_tmp:nnnnnn` 739, 764, 767, 770, 772, 774, 777, 780
`__enumext_tmp:w` 3966, 3969
`\l__enumext_tmpa_vii_int` 3225, 3228
`\l__enumext_tmpa_viii_int` 3615, 3618
`\l__enumext_tmpa_X_int` 155
`\l__enumext_topsep_v_skip` 1036, 1040, 1190, 1203, 1211, 1216, 1236, 1240, 3157, 3189
`\l__enumext_topsep_vii_skip` . . 1267, 1276, 1280
`\l__enumext_topsep_viii_skip` . 1289, 1311, 1315
`\l__enumext_vspace_a_star_v_bool` 1432
`\l__enumext_vspace_a_star_vii_bool` . . 1454
`\l__enumext_vspace_a_star_viii_bool` . . 1465
`\l__enumext_vspace_a_star_X_bool` 85
`__enumext_vspace_above:` . . 52, 1400, 1400, 2899
`__enumext_vspace_above_v:` . 52, 1428, 1428, 3027
`\l__enumext_vspace_above_v_skip` . . 1430, 1434, 1436
`__enumext_vspace_above_vii:` . . 53, 1450, 1450, 3409
`\l__enumext_vspace_above_vii_skip` 1452, 1456, 1458
`__enumext_vspace_above_viii:` . 53, 1450, 1461, 3766
`\l__enumext_vspace_above_viii_skip` 1463, 1467, 1469
`\l__enumext_vspace_b_star_v_bool` 1443
`\l__enumext_vspace_b_star_vii_bool` . . . 1476
`\l__enumext_vspace_b_star_viii_bool` . . 1487
`\l__enumext_vspace_b_star_X_bool` 85
`__enumext_vspace_below:` . . 52, 1414, 1414, 2978
`__enumext_vspace_below_v:` . 52, 1439, 1439, 3095
`\l__enumext_vspace_below_v_skip` . . 1441, 1445, 1447
`__enumext_vspace_below_vii:` . . 53, 1472, 1472, 3419
`\l__enumext_vspace_below_vii_skip` 1474, 1478, 1480
`__enumext_vspace_below_viii:` . 53, 1472, 1483, 3774
`\l__enumext_vspace_below_viii_skip` 1485, 1489, 1491
`__enumext_widest_from:nnNn` . . 37, 703, 703, 718, 730
`\g__enumext_widest_label_tl` 23, 32, 55, 451, 455, 459
`\l__enumext_wrap_label_opt_v_bool` 2607
`\l__enumext_wrap_label_opt_vii_bool` 95, 3470
`\l__enumext_wrap_label_opt_viii_bool` . . 101, 3803

`\l__enumext_wrap_label_opt_X_bool` [85](#)
`\l__enumext_wrap_label_v_bool` [2603](#), [2607](#), [2615](#),
 [2671](#)
`\l__enumext_wrap_label_vii_bool` [95](#), [3469](#), [3474](#),
 [3482](#), [3550](#)
`\l__enumext_wrap_label_viii_bool` . [101](#), [3802](#),
 [3807](#), [3815](#), [3894](#)
`\l__enumext_wrap_label_X_bool` [85](#)
`__enumext_wrapper_label_v:n` [2673](#), [3136](#)
`__enumext_wrapper_label_vii:n` [3553](#)
`__enumext_wrapper_label_viii:n` [3897](#)
`__enumext_zero_parsep:` . . . [47](#), [1087](#), [1142](#), [1142](#)
`enumext*` [5](#), [3331](#)
`enumXi` [423](#)
`enumXii` [423](#)
`enumXiii` [423](#)
`enumXiv` [423](#)
`enumXv` [423](#)
`enumXvi` [423](#)
`enumXvii` [423](#)
`enumXviii` [423](#)

Environments provide by `enumext`:

`enumext*` [22](#), [23](#), [25–27](#), [29](#), [32](#), [35](#), [36](#), [39](#), [40](#), [42](#), [43](#), [49](#),
 [50](#), [53–56](#), [58–68](#), [70](#), [73](#), [79](#), [80](#), [82](#), [92–94](#), [96](#), [97](#), [99](#),
 [101](#), [103](#), [105](#), [108](#), [110](#)
`enumext` [22](#), [23](#), [25](#), [27](#), [29](#), [32–36](#), [38–56](#), [58–68](#), [70](#), [73–75](#),
 [77](#), [78](#), [80–82](#), [84](#), [85](#), [88](#), [90](#), [91](#), [94](#), [105](#), [108](#), [109](#)
`keyans*` [22–28](#), [32](#), [35–37](#), [39](#), [40](#), [42](#), [43](#), [49](#), [50](#), [53](#), [59](#), [60](#),
 [62–64](#), [70](#), [73](#), [79](#), [100](#), [108](#), [110](#)
`keyanspic` [22–25](#), [28](#), [32](#), [33](#), [36](#), [50](#), [59](#), [60](#), [62](#), [64](#), [70–72](#),
 [87–89](#), [110](#)
`keyans` [22–25](#), [27](#), [28](#), [32](#), [33](#), [36](#), [38–42](#), [44](#), [48](#), [50–52](#), [59](#),
 [60](#), [62–64](#), [70–72](#), [76–78](#), [84](#), [85](#), [87–89](#), [91](#), [101](#), [108](#),
 [110](#)

Environments:

`list` [26](#), [30](#), [77](#), [78](#), [80](#)
`lrbox` [90](#), [96](#), [97](#), [103](#)
`minipage` [26](#), [30](#), [43](#), [45](#), [87–90](#), [96](#), [97](#), [103](#)
`multicols` [43–46](#), [50](#), [83–86](#)

exp commands:

`\exp_after:wN` [3969](#)
`\exp_args:Ne` [2823](#), [3957](#)
`\exp_not:N` . [45](#), [454](#), [562](#), [607](#), [624](#), [677](#), [816](#), [830](#), [831](#),
 [842](#), [843](#), [854](#), [855](#), [2193](#), [2299](#), [2300](#), [2415](#), [2470](#), [2471](#),
 [2483](#), [2484](#), [3856](#), [3857](#), [3966](#)
`\exp_not:n` [249](#), [263](#), [275](#), [282](#), [289](#), [562](#), [563](#), [607](#), [608](#),
 [624](#), [625](#), [677](#), [678](#), [817](#), [1522](#), [1534](#), [1985](#), [1992](#), [2158](#),
 [2169](#), [2179](#), [2193](#), [2194](#), [2271](#), [2379](#), [2417](#), [2419](#), [2843](#),
 [2856](#), [3389](#), [3402](#)

F

`\fbox` [1928](#)
file commands:
 `\file_input_stop:` [4326](#)
`first` [868](#)
`font` [463](#)
`\footnote` [73](#)
`\footnote` [73](#), [2522](#)
`\footnotemark` [2532](#)
`\footnotesize` [2300](#), [2471](#), [2484](#), [3857](#)
`\footnotetext` [2516](#)

G

`\getkeyans` [14](#), [104](#), [3955](#)

group commands:

`\group_begin:` . . [2120](#), [2298](#), [2469](#), [2482](#), [3529](#), [3548](#),
 [3855](#), [3882](#), [3892](#), [3977](#), [4011](#)
`\group_end:` [2129](#), [2305](#), [2476](#), [2489](#), [3558](#), [3570](#), [3862](#),
 [3902](#), [3921](#), [3979](#), [4018](#)

H

`\hbadness` [3577](#), [3928](#)
hbox commands:
 `\hbox_set:Nn` [443](#)
`\hfill` [493](#), [497](#), [502](#), [503](#), [1354](#), [1372](#), [2193](#), [2415](#), [3307](#), [3697](#)
hook commands:
 `\hook_gput_code:nnn` [9](#), [182](#), [186](#), [361](#)
 `\hook_gset_rule:nnnn` [362](#)
`\hspace` [3588](#), [3939](#)
`\hyperlink` [67](#), [72](#)
`\hyperlink` [2193](#), [2415](#)
`\hypertarget` [31](#)
`\hypertarget` [391](#)

I

`\IfHyperBoolean` [369](#)
`\IfPackageLoadedTF` [11](#), [365](#), [379](#)
`\ignorespaces` [819](#)
`\inputlineno` [249](#), [263](#), [275](#), [282](#), [289](#)
int commands:
 `\int_add:Nn` [3267](#), [3657](#)
 `\int_case:nn` [993](#), [1144](#), [1835](#), [1857](#), [1884](#)
 `\int_compare:nNnTF` . . [332](#), [595](#), [612](#), [632](#), [639](#), [1069](#),
 [1188](#), [1333](#), [1337](#), [1341](#), [1908](#), [1914](#), [2112](#), [2116](#), [2313](#),
 [2352](#), [2357](#), [2362](#), [2387](#), [2465](#), [2807](#), [2817](#), [2870](#), [2883](#),
 [2922](#), [2938](#), [2952](#), [2966](#), [3012](#), [3016](#), [3045](#), [3070](#), [3083](#),
 [3103](#), [3107](#), [3163](#), [3237](#), [3247](#), [3263](#), [3358](#), [3427](#), [3437](#),
 [3583](#), [3592](#), [3627](#), [3637](#), [3653](#), [3745](#), [3752](#), [3934](#), [3943](#),
 [4078](#)
 `\int_compare_p:nNn` . . . [220](#), [229](#), [241](#), [242](#), [255](#), [256](#),
 [1841](#), [2214](#), [2224](#), [2236](#), [2237](#), [2252](#), [2254](#)
 `\int_decr:N` [3266](#), [3656](#)
 `\int_eval:n` . . [318](#), [324](#), [2006](#), [2266](#), [2300](#), [2370](#), [2471](#),
 [2484](#), [2726](#), [2768](#), [3255](#), [3645](#), [3857](#)
 `\int_from_alph:n` [697](#), [711](#)
 `\int_from_roman:n` [699](#), [713](#)
 `\int_gadd:Nn` [3268](#), [3658](#)
 `\int_gdecr:N` [1844](#), [1848](#), [1851](#), [1854](#), [1862](#)
 `\int_gincr:N` [1675](#), [1680](#), [2125](#), [2425](#), [2553](#), [2583](#), [2621](#),
 [2912](#), [3036](#), [3125](#), [3447](#), [3525](#), [3780](#), [3846](#)
 `\int_gset:Nn` [322](#), [2530](#)
 `\int_gset_eq:NN` [1574](#), [1581](#), [1587](#), [1593](#), [1601](#), [1608](#),
 [1614](#), [1620](#), [2527](#)
 `\int_gzero:N` . [296](#), [297](#), [298](#), [1362](#), [1379](#), [1920](#), [2971](#),
 [3088](#), [3601](#), [3952](#)
 `\int_if_exist:NnTF` [1549](#), [1585](#), [1591](#), [1612](#), [1618](#), [1788](#)
 `\int_incr:N` [2806](#), [3007](#), [3162](#), [3357](#), [3446](#), [3744](#), [3779](#)
 `\int_mod:nn` [3594](#), [3945](#)
 `\int_new:N` . [20](#), [21](#), [22](#), [23](#), [24](#), [48](#), [49](#), [73](#), [89](#), [101](#), [108](#),
 [116](#), [129](#), [130](#), [138](#), [139](#), [140](#), [141](#), [152](#), [158](#), [159](#), [160](#),
 [161](#), [162](#), [1551](#), [1791](#)
 `\int_set:Nn` [693](#), [697](#), [699](#), [1688](#), [1695](#), [1707](#), [1716](#), [1982](#),
 [2153](#), [3202](#), [3203](#), [3225](#), [3236](#), [3242](#), [3258](#), [3577](#), [3615](#),
 [3626](#), [3632](#), [3648](#), [3928](#), [4074](#)
 `\int_set_eq:NN` . [1676](#), [1681](#), [2838](#), [3265](#), [3384](#), [3655](#)
 `\int_sign:n` [326](#)
 `\int_step_function:nnN` [2230](#), [2244](#), [2259](#)
 `\int_step_inline:nnn` [3204](#), [4101](#)
 `\int_to_roman:n` [190](#), [2210](#), [2248](#)

`\int_use:N` 311, 316, 317, 1070, 1690, 1697, 1709, 1718, 2726, 2745, 2768, 2824, 2923, 2932, 2947, 2953, 3240, 3241, 3253, 3630, 3631, 3643

`\int_zero:N` 3586, 3937

`\c_one_int` . 3225, 3244, 3250, 3256, 3260, 3263, 3615, 3634, 3640, 3646, 3650, 3653

`\c_zero_int` 2214, 2224, 2236, 2237, 2252, 2254, 3427, 3437, 3597, 3948

`\item` 30, 41, 42, 65, 74, 87, 89, 90, 92, 99

`\item` 74, 76, 94, 96, 101, 103, 354, 2024, 2030, 2055, 2063, 2150, 2389, 2392, 2590, 2625, 3343, 3345, 3731, 3733, 3844

`\item*` 6, 12, 62, 2623

`item-pos*` 2498

`item-sym*` 2498

`\itemindent` 23, 78

`\itemindent` 78

`itemindent` 783

`\itemsep` 88, 89

`\itemsep` 3178, 3184

`\itemwidth` 3232, 3276, 3280, 3622, 3666, 3670

K

`keyans` 12, 2983

`keyans*` 12, 3720

`keyanspic` 13, 3138

Keys for environments provide by [enumext](#):

`above*` 24, 51–53

`above` 24, 51–53, 82, 85, 93, 100

`after` 40–42, 84, 87, 94, 100

`align` 24, 33, 77, 96

`before*` 40–42, 82, 93, 100

`before` 40–42, 85

`below*` 24, 51–53

`below` 24, 51–53, 84, 87, 94, 100

`check-ans` 24, 25, 27, 28, 59–62, 66, 72, 74, 75, 82, 84, 97, 109

`columns-sep*` 25, 63, 81, 93

`columns-sep` 42, 64, 81, 83, 86, 93

`columns*` 25, 63, 81, 93

`columns` 24, 42, 43, 45, 51, 64, 81, 83, 86, 93

`first` 40–42, 96

`font` 33, 77, 96

`item-pos*` 66, 67, 73

`item-sym*` 23, 66, 67, 73, 75

`item*-sep` 75

`itemindent` 24, 39, 76, 96

`itemsep` 38, 79

`labelsep` 33, 75, 78, 96

`labelwidth` 32–37, 78

`label` 23, 32, 34, 37, 90

`lisparindent` 79

`list-indent` 23, 39, 89

`list-offset` 39

`listparindent` 39, 96

`mark-ans` 25, 62, 69

`mark-pos` 62, 63

`mark-ref` 25, 62, 68

`mini-env` .. 24, 42, 45, 50, 51, 73, 82, 85, 91, 93, 99, 100

`mini-sep` 24, 42, 82, 85

`miniright*` 24, 43

`miniright` 24, 43, 50, 92

`minirigth*` 27

`minirigth` 27

`no-store` 25, 59, 60

`noitemsep` 38, 47

`nosep` 38, 47

`parindent` 79

`parsep` 38, 79, 96

`partopsep` 38

`ref` 23, 27, 34–36, 109

`resume*` 23, 53, 54, 58, 59, 84

`resume` 23, 53–59, 84, 94

`rightmargin` 39

`save-ans` 24, 54–60, 64, 66, 70, 71, 76, 81, 85, 87, 94, 101, 102, 104, 105, 109

`save-key` 25, 54, 81

`save-ref` 26, 31, 62, 66–68, 70, 72, 76, 102

`save-sep` 62, 102

`series` 23, 53–58, 81, 84

`show-ans` 25, 62, 63, 65, 66, 69, 76, 102

`show-length` 27, 40, 108

`show-pos` 25, 62, 63, 65, 66, 69, 72, 76, 102

`start` 24, 27, 37, 38, 54

`store-brk` 66, 67

`topsep` 38

`widest` 23, 27, 37, 38

`wrap-ans` 62, 65, 69

`wrap-label*` 33, 74, 77, 95, 96, 101

`wrap-label` 33, 77, 95, 96, 101

`wrap-opt` 62

keys commands:

`\keys_define:nn` 465, 487, 516, 529, 576, 647, 721, 741, 785, 804, 861, 870, 949, 966, 1385, 1496, 1743, 1796, 1925, 1961, 1979, 2096, 2500, 3981, 4044

`\l_keys_key_str` 4159

`\keys_set:nn` . 479, 973, 1390, 1395, 1637, 1642, 1729, 1737, 2142, 2819, 2823, 3023, 3370, 3761, 4046, 4047, 4048, 4049, 4050, 4051, 4052, 4053, 4054, 4055, 4056, 4057, 4095

keyval commands:

`\keyval_parse:NNn` 1510

L

`label` 527, 574, 647

Labels provide by [enumext](#):

`\Alph*` 32

`\Roman*` 32

`\alph*` 32

`\arabic*` 27, 32

`\roman*` 32

`\labelsep` 89

`\labelsep` 3179, 3182

`labelsep` 463

`\labelwidth` 32, 89

`\labelwidth` 3179, 3180

`labelwidth` 463

`\leftmargin` 23, 78

`\leftmargin` 78, 3179

legacy commands:

`\legacy_if:nTF` 3513, 3516, 3870, 3873

`\legacy_if_gset_false:n` 1063

`\legacy_if_set_false:n` 3515, 3872

`\legacy_if_set_true:n` 3475, 3500, 3507, 3520, 3808, 3836, 3877

`\linewidth` 82, 85

`\linewidth` 2907, 3033, 3201, 3228, 3289, 3618, 3679

`\list` 30

`\list` 352

`list-indent` 783

`list-offset` 783

N	
\NeedsTeXFormat	3
\newcounter	420
\NewDocumentCommand	1331, 2106, 3097, 3955, 4009, 4065
\NewDocumentEnvironment	2784, 2983, 3138, 3331, 3720
\newlabel	31
\newlabel	403
no-store	<u>1794</u>
\noindent	<u>92, 99</u>
\noindent	2914, 3038, 3298, 3344, 3585, 3688, 3732, 3936
\nointerlineskip	2914, 3038, 3298, 3688

Packages:

- enumext 22, 34, 58, 78, 87, 107
- enumitem 32
- expl3 90
- footnotehyper 31
- hyperref 25, 26, 30, 31, 67, 72, 96, 107
- lua-visual-debug 45
- multicol 22, 107
- shortlst 90
- \par .. 1029, 1057, 1180, 1259, 1322, 1328, 1357, 1374, 2278, 2958, 2973, 3075, 3090, 3213, 3316, 3323, 3585, 3599, 3706, 3713, 3936, 3950
- \parindent 3562, 3906
- \parsep 44, 47, 88, 89
- \parsep 2056, 2064, 2765, 3178, 3185, 3190
- parsep 739
- \parskip 3563, 3907
- \partopsep 89
- \partopsep 2766, 3183
- partopsep 739
- peek commands:
 - \peek_meaning:NTF 3452, 3466, 3483, 3494, 3785, 3799, 3816
 - \peek_meaning_remove:NTF 3459, 3792
 - \peek_remove_spaces:n 2629
- \phantomsection 31
- \phantomsection 392
- prg commands:
 - \prg_do_nothing: 396
 - \prg_new_protected_conditional:Npnn .. 192
 - \prg_replicate:nn 209
 - \prg_return_false: 196
 - \prg_return_true: 195
- \printkeyans 14, 105, 4009
- prop commands:
 - \prop_count:N 309, 2006, 2266, 2302, 2370, 2473, 2486, 3859
 - \prop_gput_if_not_in:Nnn 2004
 - \prop_if_exist:NTF 1778, 3975
 - \prop_item:Nn 3978
 - \prop_new:N 1781
- \ProvidesExplPackage 4

R	
\raggedcolumns	2946, 3064
\ref	68, 70
ref	<u>527, 574, 647</u>
\refstepcounter	3522, 3879
regex commands:	
\regex_match:nnTF	194, 696, 698, 710, 712, 2836, 2849, 3382, 3395
\regex_replace_once:nnN	202
\renewcommand	562, 607, 624, 677
\RenewDocumentCommand	2522, 2590, 2625, 2651, 2667
\RequirePackage	17
resume	<u>1494</u>
resume*	<u>1494</u>
rightmargin	<u>783</u>
\Roman	32, 37

```

\small ..... 3987, 3992, 3997, 4002, 4007
\star ..... 2504
start ..... 719
\stepcounter ..... 2526, 3118
str commands:
  \c_backslash_str 4131, 4135, 4139, 4143, 4144, 4145,
    4149, 4150, 4246, 4250, 4259, 4260, 4264, 4265, 4269,
    4270, 4301, 4302, 4306, 4311, 4312
  \c_colon_str ..... 2265, 2369, 3966
  \c_left_brace_str .. 4211, 4215, 4228, 4234, 4240
  \c_right_brace_str .. 4211, 4215, 4228, 4234, 4240
  \str_case:nn ..... 214, 269
  \str_case:nnTF ..... 1517, 1526
  \str_clear:N ..... 2816, 3369
  \str_count:n ..... 209
  \str_if_empty:NTF ..... 1538, 1579, 1606
  \str_if_eq:nnTF ..... 2727, 2770
  \str_if_in:nnTF ..... 3962
  \str_new:N ..... 126, 167
  \str_set:Nn ... 519, 520, 521, 1940, 1941, 1964, 1965
\string ..... 385
\strutbox . 1077, 1081, 1085, 1096, 1100, 1111, 1120, 1126,
  1136, 1149, 1155, 1161, 1192, 1193, 1194, 1197, 1207,
  1211, 1220, 1227, 1232, 1240, 1269, 1270, 1273, 1280,
  1293, 1301, 1307, 1315, 3188

T
TEX and LATEX 2ε commands:
  \@auxout ..... 401
  \@currentvir ..... 214, 269
  \protected@write ..... 401
text commands:
  \text_expand:n ..... 3958
\textasteriskcentered ..... 1937, 1951
\thepage ..... 407
tl commands:
  \c_space_tl ..... 2451, 4173, 4188, 4211, 4215
  \tl_clear:N .. 492, 498, 1921, 2139, 2312, 2386, 3822
  \tl_clear_new:N ..... 449
  \tl_const:Nn ..... 37, 433
  \tl_gclear:N . 302, 303, 304, 1559, 1564, 2662, 3327,
    3546, 3717
  \tl_gclear_new:N ..... 1546
  \tl_gput_right:Nn ..... 434
  \tl_greplace_all:Nnn ..... 455
  \tl_gset:Nn 246, 247, 260, 261, 1547, 1560, 1565, 1777,
    3489
  \tl_gset_eq:NN ..... 451, 2571, 3539
  \tl_if_blank:nTF ..... 3537
  \tl_if_empty:NTF . 334, 550, 569, 597, 614, 634, 641,
    667, 684, 1572, 1577, 1599, 1604, 1662, 1726, 1734,
    1753, 1817, 2020, 2051, 2165, 2323, 2396, 2445, 2641,
    3825, 4103
  \tl_if_empty:nTF ..... 1627
  \tl_if_exist:NTF ..... 1632
  \tl_if_novalue:nTF .. 2140, 2151, 2320, 2394, 2430,
    2524, 2549, 2567, 2572, 2601, 2814, 3149, 3367, 3759,
    3823, 4067
  \tl_map_inline:Nn ..... 200, 452
  \tl_new:N 34, 39, 40, 43, 44, 50, 52, 53, 54, 56, 57, 90, 91,
    92, 98, 99, 100, 102, 103, 104, 105, 106, 107, 113, 114,
    124, 125, 135, 136, 137, 144, 145, 146, 149, 166, 169
  \tl_put_left:Nn 2028, 2061, 2148, 2457, 2492, 3841,
    3844
  \tl_put_right:Nn 450, 560, 605, 622, 675, 1983, 1990,
    2032, 2067, 2150, 2156, 2164, 2167, 2177, 2182, 2185,

```

2191, 2217, 2227, 2241, 2257, 2263, 2268, 2315, 2318, 2325, 2327, 2354, 2359, 2364, 2367, 2376, 2389, 2392, 2398, 2403, 2413, 2841, 2854, 3387, 3400, 3827, 3828, 3983, 3988, 3993, 3998, 4003	\topsep 2057, 2065
\tl_remove_all:Nn 4102	topsep <u>739</u>
\tl_remove_once:Nn 2205, 2339	\typeout 371, 374, 384, 385
\tl_replace_all:Nnn 454	
\tl_reverse:N 2204, 2206, 2338, 2340	U
\tl_set:Nn . 45, 273, 280, 287, 419, 493, 497, 502, 503, 549, 594, 666, 814, 828, 840, 852, 1661, 1752, 2296, 2432, 2467, 2480, 2569, 3830, 3853, 4100	\u 203
\tl_set_eq:NN 460, 555, 558, 602, 604, 619, 621, 672, 674, 2203, 2337, 2350, 2613, 2617, 3130, 3132	use commands:
\tl_to_str:n 1632, 1638, 1643, 3958	\use:N 210, 2658, 2793
\tl_trim_spaces:n 450, 4088, 4100, 4106	\use:n 1508, 3964
\tl_use:N . 456, 459, 571, 636, 643, 686, 885, 889, 893, 897, 901, 905, 909, 913, 917, 921, 925, 929, 933, 937, 941, 945, 2090, 2210, 2218, 2229, 2243, 2248, 2260, 2556, 2562, 2586, 2604, 2608, 2616, 2653, 2654, 2661, 2669, 2670, 2676, 2791, 2989, 3135, 3321, 3549, 3560, 3564, 3711, 3893, 3904, 3910, 3914, 4012, 4013, 4014, 4015, 4016, 4084	\use_none:nn 395
token commands:	\usecounter 2725, 2767
\token_to_str:N 403	
	V
	\value 1575, 1581, 1588, 1594, 1602, 1608, 1615, 1621
	\vspace 1064, 1407, 1410, 1421, 1424, 1434, 1436, 1445, 1447, 1456, 1458, 1467, 1469, 1478, 1480, 1489, 1491, 2056, 2064, 3146, 3157, 3600, 3951
	W
	widest <u>719</u>
	wrap-ans <u>1923</u>
	wrap-label <u>463</u>
	wrap-label* <u>463</u>
	wrap-opt <u>1923</u>