

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-22^{*}

©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	2	4	The storage system	9
1.1	Description and usage	3	4.1	Keys for storage	9
1.2	The concept of left margin	3	4.2	Keys for internal label and ref	10
1.3	User interface	3	4.3	Keys for debugging and checking	10
1.3.1	Internal counters	3	4.4	The command <code>\anskey</code>	10
1.3.2	Support for <code>multicol</code>	4	4.5	The environment <code>keyans</code>	11
1.3.3	Support for <code>minipage</code>	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	12
1.3.5	Support for <code>\footnote</code>	4	4.6.1	The command <code>\anspic</code>	12
2	The environment <code>enumext</code>	4	4.7	Printing stored content	13
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	13
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	13
3	The command <code>\setenumext</code>	5	5	Full examples	14
3.1	Keys for <code>label</code> and <code>ref</code>	6	6	The way of non-enumerated lists	16
3.2	Keys for spaces	6	7	References	18
3.2.1	Vertical spaces	7	8	Change history	18
3.2.2	Horizontal spaces	7	9	Index of Documentation	19
3.3	Keys for <code>add code</code>	8	10	Implementation	21
3.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	8	11	Index of Implementation	109
3.5	Keys for <code>multicols</code>	9			
3.6	Keys for <code>minipage</code>	9			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

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 2024-05-22.

[†]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 (lpp), version 1.3 or later (<https://www.latex-project.org/lpp1.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[3] package, need to have a modern TeX distribution such as TeX Live or MiKTeX. 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

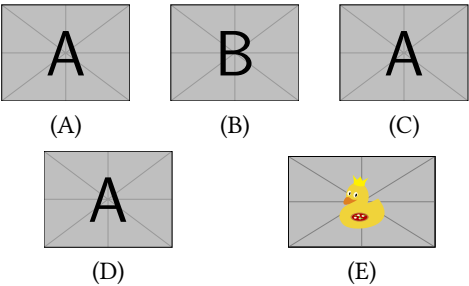
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an “extra space” that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment.
Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[5] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

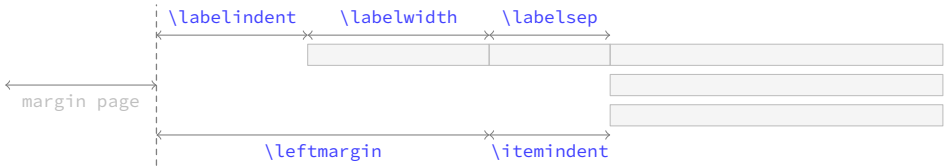


Figure 1: Representation of horizontal lengths in `enumitem`.

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

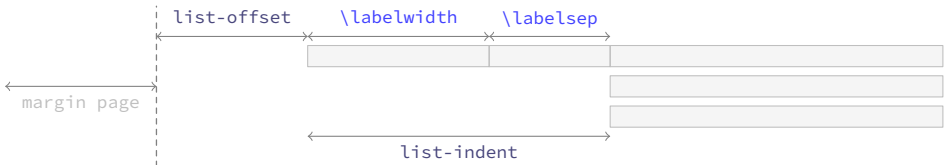


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

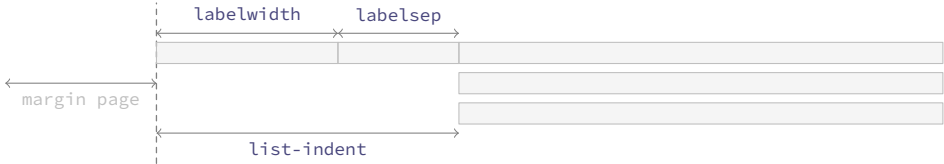


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

The user interface consists in `enumext`, `enumext*`, `keyans`, `keyans*` and `keyanspic` environments, `\anskey`, `\item*` and `\anspic*` commands to \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.

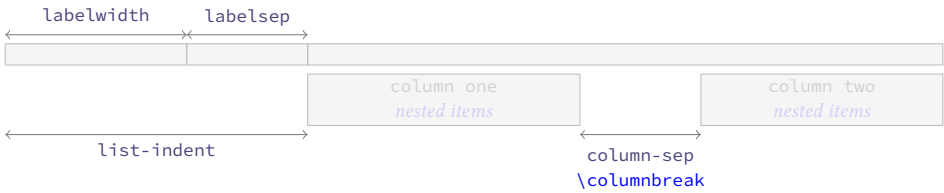


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 \TeX `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§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. \TeX 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 \TeX 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



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.

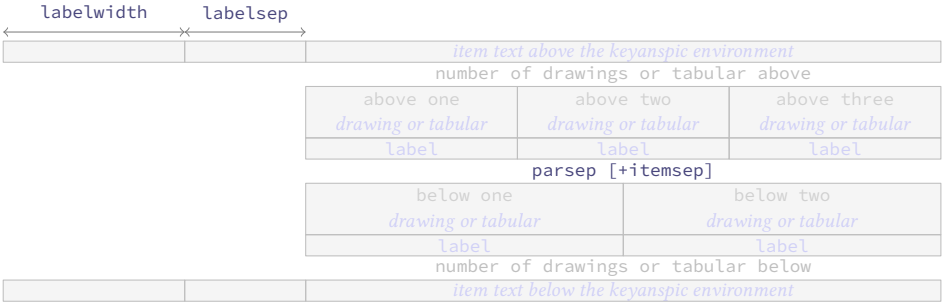


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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}`

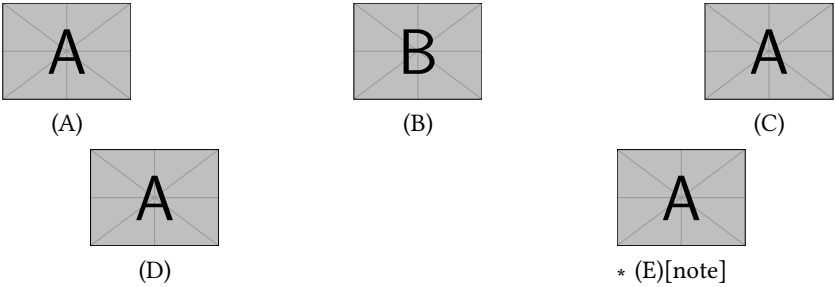
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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
3. Third type of questions
- (1) $2\alpha + 2\delta = 90^\circ$
- (2) $\angle EDF = 45^\circ$
- A

 value
- B

 value
- C

 value
4. Question with image and label below:



A



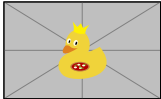
D



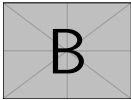
B



C



E



5. Question with image on left side:
- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value

Test keys

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

- * 4. E, A duck
- * 5. D, other note
- *

*

*

*

Example 4

A “simple worksheet” using ducks :) 📄.



Factor $x^2 - 2x + 1$



Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)



True False

- (a) $\alpha > \delta$
- (b) ~~ETX~~ze is cool?



Related to Linux

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

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

1. $(x - 1)^2$
2. $3(x + y + z)$
3. (a) False
- (b) Very True!
4. (a) Yes

- * (b) Yes, dnf
- * (c) i. doesn't exist for now :(
- * ii. very good
- * iii. obsolete
- *

*

*

*

*

Example 5

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

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

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 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 2024-05-22 – 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 / 121

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	\topsep	8
\partopsep	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   {2024-05-22}
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.)

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.)

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.)

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.)

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.)

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
\l__enumext_check_start_line_env_tl
\g__enumext_check_start_line_env_tl
\g__enumext_check_starred_cmd_int
\g__enumext_count_item_anskey_int
\g__enumext_count_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 \tl_new:N \l__enumext_check_start_line_env_tl
135 \tl_new:N \g__enumext_check_start_line_env_tl
136 \int_new:N \g__enumext_check_starred_cmd_int
137 \int_new:N \g__enumext_count_item_anskey_int
138 \int_new:N \g__enumext_count_item_number_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`.

```
139 \bool_new:N \l__enumext_hyperref_bool
140 \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.

```

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

```

149 \int_new:N \g__enumext_footnote_int
150 \seq_new:N \g__enumext_footnote_arg_seq
151 \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.

```

152 \cs_set_protected:Npn \__enumext_tmp:n #1
153 {
154   \bool_new:c { l__enumext_item_starred_#1_bool }
155   \int_new:c { l__enumext_item_column_pos_#1_int }
156   \int_new:c { g__enumext_item_count_all_#1_int }
157   \int_new:c { l__enumext_joined_item_#1_int }
158   \int_new:c { l__enumext_joined_item_aux_#1_int }
159   \int_new:c { l__enumext_tmpa_#1_int }
160   \box_new:c { l__enumext_item_text_#1_box }
161   \dim_new:c { l__enumext_joined_width_#1_dim }
162   \dim_new:c { l__enumext_item_width_#1_dim }
163   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
164   \str_new:c { l__enumext_align_label_#1_str }
165   \bool_new:c { g__enumext_minipage_active_#1_bool }
166   \tl_new:c { g__enumext_miniright_code_#1_tl }
167   \bool_new:c { g__enumext_minipage_center_#1_bool }
168   \dim_new:c { g__enumext_minipage_right_#1_dim }
169   \skip_new:c { g__enumext_minipage_right_#1_skip }
170 }
171 \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`.

```

172 \clist_const:Nn \c__enumext_all_envs_clist
173 {
174   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
175   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
176 }

```

(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.

```

177 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
178 {
179   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
180 }

```

(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.

```
181 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
182 {
183   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
184 }
```

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

`__enumext_level:` Function for check current level in `enumext`.

```
185 \cs_new:Nn \__enumext_level:
186 {
187   \int_to_roman:n { \l__enumext_level_int }
188 }
```

(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
189 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
190 {
191   \regex_match:nnTF { ^[\+\\-]?[\d]+$ } {#1} % $
192   { \prg_return_true: }
193   { \prg_return_false: }
194 }
```

(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.

```
195 \cs_new_protected:Nn \__enumext_regex_counter_style:
196 {
197   \tl_map_inline:Nn \c__enumext_counter_style_tl
198   {
199     \regex_replace_once:nnN { \c{##1}\* }
200     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
201   }
202 }
```

(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.

```
203 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
204 {
205   * ~ #2
206   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
207   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
208 }
```

(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.

```
209 \cs_new_protected:Nn \__enumext_is_not_nested:
210 {
211   \str_case:en { \@currenvir }
212   {
213     {enumext}
214     {
215       \bool_lazy_and:nnT
216       { \bool_not_p:n { \g__enumext_standar_bool } }
217       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
218       {
219         \bool_gset_true:N \g__enumext_standar_bool
220       }
221     }
222 }
```

```

221     }
222     {enumext*}
223     {
224         \bool_lazy_and:nnT
225         { \bool_not_p:n { \g__enumext_starred_bool } }
226         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
227         {
228             \bool_gset_true:N \g__enumext_starred_bool
229         }
230     }
231 }
232 }

```

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_check_start_line_env_tl` to use in messages related to the `check-ans` key.

```

233 \cs_new_protected:Nn \__enumext_is_on_first_level:
234 {
235     \bool_lazy_all:nT
236     {
237         { \bool_if_p:N \g__enumext_standar_bool }
238         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
239         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
240     }
241     {
242         \bool_set_true:N \l__enumext_standar_first_level_bool
243         \tl_gset:Nx \g__enumext_check_start_line_env_tl
244         {
245             in ~ 'enumext' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
246         }
247     }
248     \bool_lazy_all:nT
249     {
250         { \bool_if_p:N \g__enumext_starred_bool }
251         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
252         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
253     }
254     {
255         \bool_set_true:N \l__enumext_starred_first_level_bool
256         \tl_gset:Nx \g__enumext_check_start_line_env_tl
257         {
258             in ~ 'enumext*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
259         }
260     }
261 }

```

(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.

```

262 \cs_new_protected:Nn \__enumext_keyans_save_start_line:
263 {
264     \str_case:en { \@currenvir }
265     {
266         {keyans}
267         {
268             \tl_set:Nx \l__enumext_check_start_line_env_tl
269             {
270                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
271             }
272         }
273         {keyans*}
274         {
275             \tl_set:Nx \l__enumext_check_start_line_env_tl
276             {
277                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
278             }
279         }
280     }

```

```

280     {keyanspic}
281     {
282         \tl_set:Nc \l__enumext_check_start_line_env_tl
283         {
284             in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
285         }
286     }
287 }
288 }

```

(End of definition for `__enumext_keyans_save_start_line:.`)

10.6 Copying list and minipage environments

The `list` environment provided by \TeX 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.

```

289 \__enumext_at_begin_document:n
290 {
291     \cs_new_eq:NN \__enumext_start_list:nn \list
292     \cs_new_eq:NN \__enumext_stop_list: \endlist
293     \cs_new_eq:NN \__enumext_item_std:w \item
294 }

```

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

The `minipage` environment provided by \TeX 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.

```

295 \__enumext_at_begin_document:n
296 {
297     \cs_new_eq:NN \__enumext_minipage:w \minipage
298     \cs_new_eq:NN \__enumext_endminipage: \endminipage
299 }

```

(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.

```

300 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
301 \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”.

```

302 \cs_new_protected:Nn \__enumext_after_hyperref:
303 {
304     \IfPackageLoadedTF { hyperref }
305     {
306         \msg_info:nnn { enumext } { package-load } { hyperref }
307         \bool_set_true:N \l__enumext_hyperref_bool
308         \IfHyperBoolean{hyperfootnotes}
309         {
310             \typeout{hyperfootnotes=true}
311             \bool_set_true:N \l__enumext_footnotes_key_bool
312         }
313     }

```

```

313         { \typeout{hyperfootnotes=false} }
314     }
315     { }

```

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`.

```

316     \bool_if:NT \l__enumext_footnotes_key_bool
317     {
318         \IfPackageLoadedTF { footnotehyper }
319         {
320             \msg_info:nnn { enumext } { package-load } { footnotehyper }
321         }
322         {
323             \typeout{No ~ footnotehyper ~ load}
324             \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
325             \bool_set_false:N \l__enumext_footnotes_key_bool
326         }
327     }

```

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.

```

328     \bool_if:NTF \l__enumext_hyperref_bool
329     {
330         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
331         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
332     }
333     {
334         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
335         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
336     }
337 }

```

(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.

```

338 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
339 {
340     \protected@write \@auxout { }
341     {
342         \token_to_str:N \newlabel {#1}
343         {
344             {#2}
345             \bool_if:NT \l__enumext_hyperref_bool
346             { { \thepage } {#2} {#1} }
347             { }
348         }
349     }
350     \__enumext_hypertarget:nn {#1} { }
351     \__enumext_phantomsection:
352 }

```

(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 a error will be returned and the package loading will be aborted. The arguments taken by the function are:

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

#2: The counter’s name.

```

353 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
354 {
355   \cs_if_exist:cTF { c@ #2 }
356   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
357   {
358     \tl_set:Nn #1 { #2 }
359     \newcounter { #2 }
360   }
361 }

```

(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      362 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     363 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    364 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     365 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv      366 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvii    367 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii   368 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
            369 \__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.

```

370 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
371 {
372   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
373   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
374 }
375 \__enumext_register_counter_style:Nn \arabic { 0 }
376 \__enumext_register_counter_style:Nn \Alph { M }
377 \__enumext_register_counter_style:Nn \alph { m }
378 \__enumext_register_counter_style:Nn \Roman { VIII }
379 \__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.

```

380 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
381 {
382   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
383   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
384 }
385 \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.

```

386 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
387 {
388   \tl_clear_new:N #1
389   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
390   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
391   \tl_map_inline:Nn \g__enumext_counter_styles_tl
392   {
393     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }

```

```

394     \tl_greplac_all:Nne \g__enumext_widest_label_tl { ##1* }
395     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
396   }
397   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
398   { \tl_use:N \g__enumext_widest_label_tl }
399   \tl_set_eq:cN { the #2 } #1
400 }
401 \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.

```

402 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
403 {
404   \keys_define:nn { enumext / #1 }
405   {
406     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
407     font      .value_required:n = true,
408     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
409     labelsep  .initial:n   = {0.3333em},
410     labelsep  .value_required:n = true,
411     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
412     labelwidth .value_required:n = true,
413     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
414     wrap-label .initial:n   = {##1},
415     wrap-label .value_required:n = true,
416     wrap-label* .code:n = {
417       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
418       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
419     },
420     wrap-label* .value_required:n = true,
421   }
422 }
423 \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.

```

424 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
425 {
426   \keys_define:nn { enumext / #1 }
427   {
428     align .choice:,
429     align / left .code:n =
430     {
431       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
432       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
433     },
434     align / right .code:n =
435     {
436       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
437       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
438     },
439     align / center .code:n =
440     {
441       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
442       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
443     },
444     align .initial:n = left,
445     align .value_required:n = true,
446   }
447 }
448 \clist_map_inline:nn
449 {
450   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}

```



```

451 }
452 { \__enumext_tmp:nn #1 }

453 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
454 {
455   \keys_define:nn { enumext / #1 }
456   {
457     align .choice:,
458     align / left .code:n = \str_set:cn { \__enumext_align_label_#2_str } { l },
459     align / right .code:n = \str_set:cn { \__enumext_align_label_#2_str } { r },
460     align / center .code:n = \str_set:cn { \__enumext_align_label_#2_str } { c },
461     align .initial:n = left,
462     align .value_required:n = true,
463   }
464 }
465 \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.

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
466 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
467 {
468   \keys_define:nn { enumext / #1 }
469   {
470     label .code:n = {
471       \__enumext_label_style:cnv { \l__enumext_label_#2_tl }
472       { \__enumext_counter_#2_tl } {##1}
473       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
474       \l__enumext_current_widest_dim
475     },
476     label .initial:n = #3,
477     label .value_required:n = true,
478     ref .code:n = \__enumext_standar_ref:n {##1},
479     ref .value_required:n = true,
480   }
481 }
482 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
483 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
484 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
485 \__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.

```

486 \cs_new_protected:Npn \__enumext_standar_ref:n #1
487 {
488   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
489   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
490   {
491     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
492   }
493   {
494     \tl_set_eq:Nc
495     \l__enumext_ref_the_count_tl { \__enumext_counter_ \__enumext_level: _tl }
496     \__enumext_regex_counter_style:
497     \tl_set_eq:Nc
498     \l__enumext_ref_the_count_tl { \__enumext_the_counter_ \__enumext_level: _tl }
499     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }

```

```

500         {
501             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
502             { \exp_not:V \l__enumext_ref_key_arg_tl }
503         }
504     }
505 }

```

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

```

506 \cs_new_protected:Nn \__enumext_standar_ref:
507 {
508     \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
509     {
510         \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
511     }
512 }

```

(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
513 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
514 {
515     \keys_define:nn { enumext / #1 }
516     {
517         label .code:n = {
518             \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
519             { \l__enumext_counter_#2_tl } {##1}
520             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
521             \l__enumext_current_widest_dim
522         },
523         label .initial:n = #3,
524         label .value_required:n = true,
525         ref .code:n = \__enumext_starred_ref:n {##1},
526         ref .value_required:n = true,
527     }
528 }
529 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
530 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

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

```

531 \cs_new_protected:Npn \__enumext_starred_ref:n #1
532 {
533     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
534     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
535     {
536         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
537         {
538             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
539         }
540         {
541             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
542             \__enumext_regex_counter_style:
543             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
544             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
545             {
546                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
547                 { \exp_not:V \l__enumext_ref_key_arg_tl }
548             }
549         }
550     }
551     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
552     {
553         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
554         {
555             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
556         }
557     }
558 }

```

```

557     {
558       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
559       \__enumext_regex_counter_style:
560       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
561       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
562       {
563         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
564         { \exp_not:V \l__enumext_ref_key_arg_tl }
565       }
566     }
567   }
568 }

```

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.

```

569 \cs_new_protected:Nn \__enumext_starred_ref:
570 {
571   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
572   {
573     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
574     {
575       \tl_use:N \l__enumext_renew_the_count_vii_tl
576     }
577   }
578   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
579   {
580     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
581     {
582       \tl_use:N \l__enumext_renew_the_count_viii_tl
583     }
584   }
585 }

```

(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
586 \keys_define:nn { enumext / keyans }
587 {
588   label .code:n = {
589     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
590     { \l__enumext_counter_v_tl } {#1}
591     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
592     \l__enumext_current_widest_dim
593     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
594     { \l__enumext_counter_vi_tl } {#1}
595     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
596     \l__enumext_current_widest_dim
597   },
598   label .initial:n = (\Alph*),
599   label .value_required:n = true,
600   ref .code:n = \__enumext_keyans_ref:n {#1},
601   ref .value_required:n = true,
602 }

```

(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`.

```

603 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
604 {
605   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
606   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
607   {
608     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
609   }
610   {
611     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
612     \__enumext_regex_counter_style:
613     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl

```

```

614         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
615         {
616             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
617             { \exp_not:V \l__enumext_ref_key_arg_tl }
618         }
619     }
620 }

```

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

```

621 \cs_new_protected:Nn \__enumext_keyans_ref:
622 {
623     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
624     {
625         \tl_use:N \l__enumext_renew_the_count_v_tl
626     }
627 }

```

(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.

```

628 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
629 {
630     \__enumext_if_is_int:nTF { #3 }
631     {
632         \int_set:Nn #2 {#3}
633     }
634     {
635         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
636         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
637         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
638         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
639     }
640 }
641 \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.

```

642 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
643 {
644     \__enumext_if_is_int:nTF {#4}
645     {
646         \setcounter{enumX#1} { #4 }
647     }
648     {
649         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
650         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
651         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
652         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
653     }
654     \__enumext_label_width_by_box:cv
655     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
656 }
657 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

```

start
widest
\__enumext_start_X_int
658 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
659 {
660   \keys_define:nn { enumext / #1 }
661   {
662     start .code:n = {
663       \__enumext_start_from:ccn
664       { l__enumext_label_#2_tl }
665       { l__enumext_start_#2_int } {##1}
666     },
667     start .initial:n = 1,
668     widest .code:n = {
669       \__enumext_widest_from:nccn {#2}
670       { l__enumext_label_#2_tl }
671       { l__enumext_labelwidth_#2_dim } {##1}
672     },
673     widest .value_required:n = true,
674     start .value_required:n = true,
675   }
676 }
677 \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
678 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
679 {
680   \keys_define:nn { enumext / #1 }
681   {
682     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
683     topsep .initial:n = {#3},
684     topsep .value_required:n = true,
685     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
686     partopsep .initial:n = {#4},
687     partopsep .value_required:n = true,
688     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
689     parsep .initial:n = {#5},
690     parsep .value_required:n = true,
691     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
692     itemsep .initial:n = {#6},
693     itemsep .value_required:n = true,
694     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
695     noitemsep .value_forbidden:n = true,
696     nosepp .meta:n = {
697       itemsep = 0pt, parsep = 0pt,
698       topsep = 0pt, partopsep = 0pt,
699     },
700     nosepp .value_forbidden:n = true,
701   }
702 }

```

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

```

703 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
704 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
705 { 4.0pt plus 2.0pt minus 1.0pt }
706 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
707 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
708 { 2.0pt plus 1.0pt minus 1.0pt }
709 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
710 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
711 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
712 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
713 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
714 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
715 { 2.0pt plus 1.0pt minus 1.0pt }
716 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }

```

```

717 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
718 { 4.0pt plus 2.0pt minus 1.0pt }
719 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
720 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
721 { 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.

```

722 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
723 {
724   \keys_define:nn { enumext / #1 }
725   {
726     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
727     itemindent .value_required:n = true,
728     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
729     rightmargin .value_required:n = true,
730     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
731     listparindent .value_required:n = true,
732     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
733     list-offset .value_required:n = true,
734     list-indent .code:n =
735       { \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
736         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1} },
737     list-indent .value_required:n = true,
738   }
739 }
740 \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.

```

741 \cs_set_protected:Npn \__enumext_tmp:n #1
742 {
743   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
744 }
745 \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.

```

746 \cs_set_protected:Nn \__enumext_fake_item:
747 {
748   \dim_compare:nNnT
749     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
750     >
751     { \c_zero_dim }
752   {
753     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
754     {
755       \exp_not:N \mode_leave_vertical:
756       \exp_not:n { \skip_horizontal:n }
757       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } } }
758     \ignorespaces
759   }
760 }
761 }
762 \cs_set_protected:Nn \__enumext_keyans_fake_item:
763 {
764   \dim_compare:nNnT
765     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
766   {
767     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
768     {
769       \exp_not:N \mode_leave_vertical:
770       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim

```

```

771     }
772   }
773 }
774 \cs_set_protected:Nn \__enumext_fake_item_vii:
775 {
776   \dim_compare:nNnT
777   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
778   {
779     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
780     {
781       \exp_not:N \mode_leave_vertical:
782       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
783     }
784   }
785 }
786 \cs_set_protected:Nn \__enumext_fake_item_viii:
787 {
788   \dim_compare:nNnT
789   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
790   {
791     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
792     {
793       \exp_not:N \mode_leave_vertical:
794       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
795     }
796   }
797 }

```

(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.

```

798 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
799 {
800   \keys_define:nn { enumext / #1 }
801   {
802     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
803     show-length .initial:n = false,
804   }
805 }
806 \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*
after
first
807 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
808 {
809   \keys_define:nn { enumext / #1 }
810   {
811     before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
812     before .value_required:n = true,
813     before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
814     before* .value_required:n = true,
815     after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
816     after .value_required:n = true,
817     first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
818     first .value_required:n = true,
819   }
820 }
821 \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.

```
822 \cs_new_protected:Nn \__enumext_before_args_exec:
823 {
824   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
825 }
```

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⟩`.

```
826 \cs_new_protected:Nn \__enumext_before_keys_exec:
827 {
828   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
829 }
```

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

```
830 \cs_new_protected:Nn \__enumext_after_stop_list:
831 {
832   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
833 }
```

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.`

```
834 \cs_new_protected:Nn \__enumext_after_args_exec:
835 {
836   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
837 }
```

(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.

```
838 \cs_new_protected:Nn \__enumext_before_args_exec_v:
839 {
840   \tl_use:N \l__enumext_before_starred_key_v_tl
841 }
```

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⟩`.

```
842 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
843 {
844   \tl_use:N \l__enumext_before_no_starred_key_v_tl
845 }
```

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

```
846 \cs_new_protected:Nn \__enumext_after_stop_list_v:
847 {
848   \tl_use:N \l__enumext_after_stop_list_v_tl
849 }
```

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.`

```
850 \cs_new_protected:Nn \__enumext_after_args_exec_v:
851 {
852   \tl_use:N \l__enumext_after_list_args_v_tl
853 }
```

(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.

```

854 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
855 {
856   \tl_use:N \l__enumext_before_starred_key_vii_tl
857 }
858 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
859 {
860   \tl_use:N \l__enumext_before_starred_key_viii_tl
861 }

```

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`.

```

862 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
863 {
864   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
865 }
866 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
867 {
868   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
869 }

```

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

```

870 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
871 {
872   \tl_use:N \l__enumext_after_stop_list_vii_tl
873 }
874 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
875 {
876   \tl_use:N \l__enumext_after_stop_list_viii_tl
877 }

```

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`.

```

878 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
879 {
880   \tl_use:N \l__enumext_after_list_args_vii_tl
881 }
882 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
883 {
884   \tl_use:N \l__enumext_after_list_args_viii_tl
885 }

```

(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.

```

886 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
887 {
888   \keys_define:nn { enumext / #1 }
889   {
890     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
891     mini-env .value_required:n = true,
892     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
893     mini-sep .initial:n = 0.3333em,
894     mini-sep .value_required:n = true,
895     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
896     columns-sep .value_required:n = true,
897     columns .int_set:c = { l__enumext_columns_#2_int },
898     columns .initial:n = 1,
899     columns .value_required:n = true,
900   }

```

```

901 }
902 \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`.

```

903 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
904 {
905   \keys_define:nn { enumext / #1 }
906   {
907     columns      .initial:n = 2,
908     miniright    .tl_gset:c = { g__enumext_miniright_code_#2_tl },
909     miniright    .value_required:n = true,
910     miniright*   .code:n     = {
911                           \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
912                           \keys_set:nn { enumext / #1 } { miniright = {##1} }
913                         },
914     miniright*   .value_required:n = true,
915   }
916 }
917 \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.

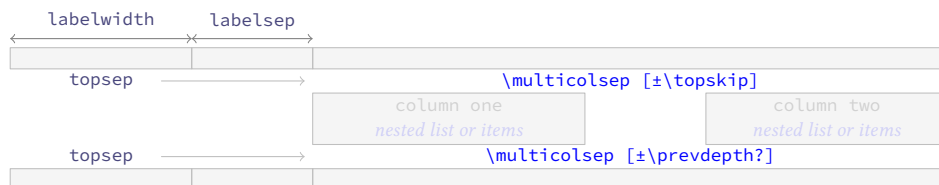


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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*.

```

918 \cs_new_protected:Nn \__enumext_multi_set_vskip:
919 {
920   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
921   {
922     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
923   }
924   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
925   {
926     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
927   }
928   \__enumext_add_pre_parsep:
929 }

```

(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*.

```

930 \cs_new_protected:Nn \__enumext_add_pre_parsep:
931 {
932   \int_case:nn { \l__enumext_level_int }
933   {
934     { 2 }{
935       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
936       {
937         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
938       }
939     }
940     { 3 }{
941       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
942       {
943         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
944       }
945     }
946     { 4 }{
947       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
948       {
949         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
950       }
951     }
952   }
953 }

```

(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 $\langle\textit{vertical mode}\rangle$.

```

954 \cs_new_protected:Nn \__enumext_multi_addvspace:
955 {
956   \__enumext_multi_set_vskip:
957   \mode_if_vertical:T
958   {
959     \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
960     {
961       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
962     }
963     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
964     {
965       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
966     }
967   }
968   \par\nopagebreak
969   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
970 }

```

(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:`

```

971 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
972 {
973   \skip_set:Nn \l__enumext_multicols_above_v_skip
974   {
975     \l__enumext_topsep_v_skip
976   }
977   \skip_set:Nn \l__enumext_multicols_below_v_skip
978   {
979     \l__enumext_topsep_v_skip
980   }
981 }
982 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
983 {

```

```

984 \__enumext_keyans_multi_set_vskip:
985 \mode_if_vertical:T
986 {
987   \skip_add:Nn \l__enumext_multicols_above_v_skip
988   {
989     \skip_use:N \l__enumext_partopsep_v_skip
990   }
991   \skip_add:Nn \l__enumext_multicols_below_v_skip
992   {
993     \skip_use:N \l__enumext_partopsep_v_skip
994   }
995 }
996 \par\nopagebreak
997 \addvspace{ \l__enumext_multicols_above_v_skip }
998 }

```

(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.

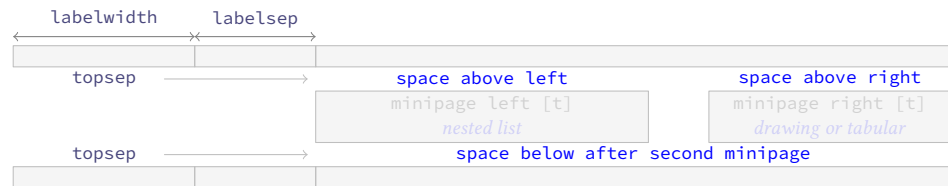


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```

999 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
1000 {
1001   \__enumext_minipage:w [ t ] { #1 }
1002   \legacy_if_gset_false:n { @minipage }
1003   \vspace { 0pt }
1004 }
1005 { \__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`.

```

1006 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1007 {

```

```

1008 \int_compare:nNnTF
1009 { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1010 {

```

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.

```

1011 \skip_if_eq:nnTF
1012 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1013 {
1014   \skip_set:Nn \l__enumext_minipage_left_skip
1015   {
1016     -0.150\box_dp:N \strutbox
1017   }
1018   \skip_set:Nn \l__enumext_minipage_right_skip
1019   {
1020     0.695\box_dp:N \strutbox
1021   }
1022   \skip_set:Nn \l__enumext_minipage_after_skip
1023   {
1024     \box_dp:N \strutbox
1025   }
1026   \__enumext_zero_parsep:
1027 }
1028 {
1029   \skip_set:Nn \l__enumext_minipage_left_skip
1030   {
1031     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1032   }
1033   \skip_set:Nn \l__enumext_minipage_right_skip
1034   {
1035     0.695\box_dp:N \strutbox
1036   }
1037   \skip_set:Nn \l__enumext_minipage_after_skip
1038   {
1039     1.85\box_dp:N \strutbox
1040     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1041   }
1042 }
1043 }
1044 {

```

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.

```

1045 \skip_if_eq:nnTF
1046 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1047 {
1048   \skip_set:Nn \l__enumext_minipage_left_skip
1049   {
1050     0.5\box_dp:N \strutbox
1051     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1052   }
1053   \skip_set:Nn \l__enumext_minipage_right_skip
1054   {
1055     \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1056   }
1057   \skip_set:Nn \l__enumext_minipage_after_skip
1058   {
1059     1.6\box_dp:N \strutbox
1060   }
1061 }
1062 {
1063   \skip_set:Nn \l__enumext_minipage_left_skip
1064   {
1065     0.5875\box_dp:N \strutbox
1066     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1067   }
1068   \skip_set:Nn \l__enumext_minipage_right_skip
1069   {
1070     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1071     + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }

```

```

1072         }
1073         \skip_set:Nn \l__enumext_minipage_after_skip
1074         {
1075             0.325\box_dp:N \strutbox
1076             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1077         }
1078     }
1079 }
1080 }

```

(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.

```

1081 \cs_new_protected:Nn \l__enumext_zero_parsep:
1082 {
1083     \int_case:nn { \l__enumext_level_int }
1084     {
1085         { 2 }{
1086             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1087             {
1088                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1089             }
1090         }
1091         { 3 }{
1092             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1093             {
1094                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1095             }
1096         }
1097         { 4 }{
1098             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1099             {
1100                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1101             }
1102         }
1103     }
1104 }

```

(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*.

```

1105 \cs_new_protected:Nn \l__enumext_mini_addvspace:
1106 {
1107     \l__enumext_mini_set_vskip:
1108     \mode_if_vertical:T
1109     {
1110         \skip_add:Nn \l__enumext_minipage_left_skip
1111         {
1112             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1113         }
1114         \skip_add:Nn \l__enumext_minipage_after_skip
1115         {
1116             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1117         }
1118     }
1119     \par\nopagebreak
1120     \addvspace { \l__enumext_minipage_left_skip }
1121 }

```

(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`.

```

1122 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1123 {
1124   \skip_zero_new:N \l__enumext_minipage_after_skip
1125   \skip_zero_new:N \l__enumext_minipage_left_skip
1126   \skip_zero_new:N \l__enumext_minipage_right_skip
1127   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1128   {
1129     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1130     {
1131       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1132       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1133       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1134       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1135       {
1136         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1137       }
1138     }
1139     {
1140       \skip_set:Nn \l__enumext_minipage_left_skip
1141       {
1142         \skip_use:N \l__enumext_topsep_v_skip
1143       }
1144       \skip_set:Nn \l__enumext_minipage_right_skip
1145       {
1146         0.705\box_dp:N \strutbox
1147       }
1148       \skip_set:Nn \l__enumext_minipage_after_skip
1149       {
1150         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1151       }
1152     }
1153   }
1154   {
1155     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1156     {
1157       \skip_set:Nn \l__enumext_minipage_left_skip
1158       {
1159         0.5\box_dp:N \strutbox
1160         + \l__enumext_partopsep_v_skip
1161       }
1162       \skip_set:Nn \l__enumext_minipage_right_skip
1163       {
1164         \l__enumext_partopsep_v_skip
1165       }
1166       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1167     }
1168     {
1169       \skip_set:Nn \l__enumext_minipage_left_skip
1170       {
1171         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1172       }
1173       \skip_set:Nn \l__enumext_minipage_right_skip
1174       {
1175         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1176       }
1177       \skip_set:Nn \l__enumext_minipage_after_skip
1178       {
1179         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1180       }
1181     }
1182   }
1183 }

```

(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`.

```

1184 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1185 {
1186   \__enumext_keyans_mini_set_vskip:
1187   \mode_if_vertical:T
1188   {
1189     \skip_add:Nn \l__enumext_minipage_left_skip
1190     {
1191       \l__enumext_partopsep_v_skip
1192     }
1193     \skip_add:Nn \l__enumext_minipage_after_skip
1194     {
1195       \l__enumext_partopsep_v_skip
1196     }
1197   }
1198   \par\nopagebreak
1199   \addvspace { \l__enumext_minipage_left_skip }
1200 }

```

(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*`.

```

1201 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1202 {
1203   \skip_zero_new:N \l__enumext_minipage_left_skip
1204   \skip_gzero_new:N \g__enumext_minipage_right_skip
1205   \skip_gzero_new:N \g__enumext_minipage_after_skip
1206   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1207   {
1208     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1209     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1210   }
1211   {
1212     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1213     \skip_gset:Nn \g__enumext_minipage_right_skip
1214     {
1215       \l__enumext_topsep_vii_skip
1216     }
1217     \skip_gset:Nn \g__enumext_minipage_after_skip
1218     {
1219       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1220     }
1221   }
1222 }
1223 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1224 {
1225   \skip_zero_new:N \l__enumext_minipage_after_skip
1226   \skip_zero_new:N \l__enumext_minipage_left_skip
1227   \skip_zero_new:N \l__enumext_minipage_right_skip
1228   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1229   {
1230     \skip_set:Nn \l__enumext_minipage_left_skip
1231     {
1232       0.5\box_dp:N \strutbox
1233     }
1234     \skip_set:Nn \l__enumext_minipage_right_skip
1235     {
1236       \l__enumext_partopsep_viii_skip
1237     }
1238     \skip_set:Nn \l__enumext_minipage_after_skip
1239     {
1240       1.6\box_dp:N \strutbox
1241     }
1242   }
1243 }

```

```

1244     \skip_set:Nn \l__enumext_minipage_left_skip
1245     {
1246         0.5875\box_dp:N \strutbox
1247     }
1248     \skip_set:Nn \l__enumext_minipage_right_skip
1249     {
1250         \l__enumext_topsep_viii_skip
1251     }
1252     \skip_set:Nn \l__enumext_minipage_after_skip
1253     {
1254         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1255     }
1256 }
1257 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext_mini_env*` environment on the *left side* when the `\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.

```

1258 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1259 {
1260     \__enumext_mini_set_vskip_vii:
1261     \par\nopagebreak
1262     \addvspace { \l__enumext_minipage_left_skip }
1263 }
1264 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1265 {
1266     \__enumext_mini_set_vskip_viii:
1267     \par\nopagebreak
1268     \addvspace { \l__enumext_minipage_left_skip }
1269 }

```

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

10.19.4 The command `\miniright`

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

`\miniright`

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

```

1270 \NewDocumentCommand \miniright { s }
1271 {
1272     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1273     {
1274         \msg_error:nnn { enumext } { wrong-miniright-place }
1275     }
1276     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1277     {
1278         \msg_error:nnn { enumext } { wrong-miniright-place }
1279     }
1280     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1281     {
1282         \__enumext_keyans_mini_right_cmd:n {#1}
1283     }
1284     { \__enumext_mini_right_cmd:n {#1} }
1285 }

```

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

`__enumext_mini_right_cmd:n`

The function `__enumext_mini_right_cmd:n` takes as argument the *starred 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 `__enumext_mini_env*` environment on the “left side”, then we open the `__enumext_mini_env*` environment on

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

```

1286 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1287 {
1288   \dim_compare:nNnTF
1289     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1290   {
1291     \__enumext_multicols_stop:
1292     \end{__enumext_mini_env*}
1293     \hfill
1294     \begin{__enumext_mini_env*}
1295       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1296       \par\addvspace { \l__enumext_minipage_right_skip }
1297       \bool_if:nF {#1}
1298       {
1299         \centering
1300       }
1301       \int_gzero:N \g__enumext_minipage_stat_int
1302     }
1303     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1304   }

```

(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.

```

1305 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1306 {
1307   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1308   {
1309     \__enumext_keyans_multicols_stop:
1310     \end{__enumext_mini_env*}
1311     \hfill
1312     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1313       \par\addvspace { \l__enumext_minipage_right_skip }
1314       \bool_if:nF {#1}
1315       {
1316         \centering
1317       }
1318       \int_gzero:N \g__enumext_minipage_stat_int
1319     }
1320     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1321   }

```

(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.

```

1322 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1323 {
1324   \keys_define:nn { enumext / #1 }
1325   {
1326     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1327     above .value_required:n = true,
1328     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1329                 \keys_set:nn { enumext / #1 } { above = {##1} },
1330     above* .value_required:n = true,
1331     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1332     below .value_required:n = true,
1333     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1334                 \keys_set:nn { enumext / #1 } { below = {##1} },
1335     below* .value_required:n = true,

```

```

1336     }
1337 }
1338 \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.

```

1339 \cs_new_protected:Nn \__enumext_vspace_above:
1340 {
1341     \skip_if_eq:nnF
1342     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1343     {
1344         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1345         {
1346             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1347         }
1348         {
1349             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1350         }
1351     }
1352 }

```

(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.

```

1353 \cs_new_protected:Nn \__enumext_vspace_below:
1354 {
1355     \skip_if_eq:nnF
1356     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1357     {
1358         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1359         {
1360             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1361         }
1362         {
1363             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1364         }
1365     }
1366 }

```

(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.

```

1367 \cs_new_protected:Nn \__enumext_vspace_above_v:
1368 {
1369     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1370     {
1371         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1372         {
1373             \vspace*{ \l__enumext_vspace_above_v_skip }
1374         }
1375         { \vspace { \l__enumext_vspace_above_v_skip } }
1376     }
1377 }

```

(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.

```

1378 \cs_new_protected:Nn \__enumext_vspace_below_v:
1379 {
1380     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1381     {
1382         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1383         {
1384             \vspace*{ \l__enumext_vspace_below_v_skip }

```

```

1385     }
1386     { \vspace { \l__enumext_vspace_below_v_skip } }
1387   }
1388 }

```

(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.

```

1389 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1390 {
1391   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1392   {
1393     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1394     {
1395       \vspace*{ \l__enumext_vspace_above_vii_skip }
1396     }
1397     { \vspace { \l__enumext_vspace_above_vii_skip } }
1398   }
1399 }
1400 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1401 {
1402   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1403   {
1404     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1405     {
1406       \vspace*{ \l__enumext_vspace_above_viii_skip }
1407     }
1408     { \vspace { \l__enumext_vspace_above_viii_skip } }
1409   }
1410 }

```

(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.

```

1411 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1412 {
1413   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1414   {
1415     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1416     {
1417       \vspace*{ \l__enumext_vspace_below_vii_skip }
1418     }
1419     { \vspace { \l__enumext_vspace_below_vii_skip } }
1420   }
1421 }
1422 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1423 {
1424   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1425   {
1426     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1427     {
1428       \vspace*{ \l__enumext_vspace_below_viii_skip }
1429     }
1430     { \vspace { \l__enumext_vspace_below_viii_skip } }
1431   }
1432 }

```

(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*
1433 \cs_set_protected:Npn \__enumext_tmp:n #1
1434 {
1435   \keys_define:nn { enumext / #1 }

```

```

1436     {
1437         series .str_set:N = \__enumext_series_str,
1438         series .value_required:n = true,
1439         resume .code:n = \__enumext_resume_series:n {##1},
1440         resume* .code:n = \__enumext_resume_starred:,
1441         resume* .value_forbidden:n = true,
1442     }
1443 }
1444 \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.

```

1445 \cs_new:Npn \__enumext_filter_series:n #1
1446 {
1447     \use:e
1448     {
1449         \keyval_parse:NNn
1450         \__enumext_filter_series_key:n
1451         \__enumext_filter_series_pair:nn {#1}
1452     }
1453 }

```

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.

```

1454 \cs_new:Npn \__enumext_filter_series_key:n #1
1455 {
1456     \str_case:nnF {#1}
1457     {
1458         { resume } {}
1459         { resume* } {}
1460     }
1461     { , { \exp_not:n {#1} } }
1462 }

```

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.

```

1463 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1464 {
1465     \str_case:nnF {#1}
1466     {
1467         { series } {}
1468         { resume } {}
1469         { start } {}
1470         { save-ans } {}
1471         { save-key } {}
1472     }
1473     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1474 }

```

(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).

```

1475 \cs_new_protected:Npn \__enumext_parse_series:n #1
1476 {
1477     \str_if_empty:NTF \l__enumext_series_str
1478     {
1479         \bool_if:NF \l__enumext_resume_active_bool
1480         {

```



```

1481         \__enumext_resume_last:n {#1}
1482     }
1483 }
1484 {
1485     \tl_gclear_new:c { g__enumext_series_ \__enumext_series_str_tl }
1486     \tl_gset:ce { g__enumext_series_ \__enumext_series_str_tl }
1487     { \__enumext_filter_series:n {#1} }
1488     \int_if_exist:cF { g__enumext_series_ \__enumext_series_str_int }
1489     {
1490         \int_new:c { g__enumext_series_ \__enumext_series_str_int }
1491     }
1492 }
1493 }

```

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.

```

1494 \cs_new_protected:Npn \__enumext_resume_last:n #1
1495 {
1496     \bool_if:NT \__enumext_standar_first_level_bool
1497     {
1498         \tl_gclear:N \g__enumext_standar_series_tl
1499         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1500     }
1501     \bool_if:NT \__enumext_starred_first_level_bool
1502     {
1503         \tl_gclear:N \g__enumext_starred_series_tl
1504         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1505     }
1506 }

```

(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).

```

1507 \cs_new_protected:Nn \__enumext_resume_save_counter:
1508 {
1509     \bool_if:NT \g__enumext_standar_bool
1510     {
1511         \tl_if_empty:NF \l__enumext_series_str
1512         {
1513             \int_gset_eq:cN
1514             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1515         }
1516         \tl_if_empty:NTF \l__enumext_resume_name_tl
1517         {
1518             \str_if_empty:NT \l__enumext_series_str
1519             {
1520                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1521             }
1522         }
1523     }
1524     \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1525     {
1526         \int_gset_eq:cN
1527         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1528     }
1529     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1530 }

```

```

1531         {
1532             \int_gset_eq:cN
1533             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1534         }
1535     }
1536     \bool_if:NT \g__enumext_starred_bool
1537     {
1538         \tl_if_empty:NF \l__enumext_series_str
1539         {
1540             \int_gset_eq:cN
1541             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1542         }
1543         \tl_if_empty:NTF \l__enumext_resume_name_tl
1544         {
1545             \str_if_empty:NT \l__enumext_series_str
1546             {
1547                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1548             }
1549         }
1550         {
1551             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1552             {
1553                 \int_gset_eq:cN
1554                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1555             }
1556         }
1557         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1558         {
1559             \int_gset_eq:cN
1560             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1561         }
1562     }
1563 }

```

(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.

```

1564 \cs_new_protected:Npn \__enumext_resume_series:n #1
1565 {
1566     \tl_if_empty:NTF {#1}
1567     {
1568         \__enumext_resume_counter:n { }
1569     }
1570     {
1571         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1572         {
1573             \__enumext_resume_counter:n {#1}
1574             \bool_if:NT \g__enumext_standar_bool
1575             {
1576                 \keys_set:nv { enumext / level-1 }
1577                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1578             }
1579             \bool_if:NT \g__enumext_starred_bool
1580             {
1581                 \keys_set:nv { enumext / enumext* }
1582                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1583             }
1584         }
1585         {
1586             \bool_if:NT \g__enumext_standar_bool
1587             {
1588                 \msg_error:nnn { enumext } { unknown-series } {#1}
1589             }
1590             \bool_if:NT \g__enumext_starred_bool

```

```

1591         {
1592             \msg_error:nnn { enumext } { unknown-series } {#1}
1593         }
1594     }
1595 }
1596 }

```

(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`.

```

1597 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1598 {
1599     \bool_set_true:N \l__enumext_resume_active_bool
1600     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1601     \tl_if_empty:NTF \l__enumext_resume_name_tl
1602     {
1603         \__enumext_resume_counter:
1604     }
1605     {
1606         \__enumext_resume_counter_series:
1607     }
1608     \__enumext_resume_counter_save_ans:
1609 }

```

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.

```

1610 \cs_new_protected:Nn \__enumext_resume_counter:
1611 {
1612     \bool_if:NT \g__enumext_standar_bool
1613     {
1614         \int_gincr:N \g__enumext_resume_int
1615         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1616     }
1617     \bool_if:NT \g__enumext_starred_bool
1618     {
1619         \int_gincr:N \g__enumext_resume_vii_int
1620         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1621     }
1622 }

```

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.

```

1623 \cs_new_protected:Nn \__enumext_resume_counter_series:
1624 {
1625     \bool_if:NT \g__enumext_standar_bool
1626     {
1627         \int_set:Nn \l__enumext_start_i_int
1628         {
1629             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1630         }
1631     }
1632     \bool_if:NT \g__enumext_starred_bool
1633     {
1634         \int_set:Nn \l__enumext_start_vii_int
1635         {
1636             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1637         }
1638     }
1639 }

```

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.

```

1640 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:

```

```

1641 {
1642   \bool_lazy_and:nnT
1643   { \bool_if_p:N \l__enumext_standar_first_level_bool }
1644   { \bool_if_p:N \l__enumext_store_active_bool }
1645   {
1646     \int_set:Nn \l__enumext_start_i_int
1647     {
1648       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1649     }
1650   }
1651   \bool_lazy_and:nnT
1652   { \bool_if_p:N \l__enumext_starred_first_level_bool }
1653   { \bool_if_p:N \l__enumext_store_active_bool }
1654   {
1655     \int_set:Nn \l__enumext_start_vii_int
1656     {
1657       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1658     }
1659   }
1660 }

```

(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.

```

1661 \cs_new_protected:Nn \__enumext_resume_starred:
1662 {
1663   \bool_if:NT \g__enumext_standar_bool
1664   {
1665     \tl_if_empty:NF \g__enumext_standar_series_tl
1666     {
1667       \__enumext_resume_counter:n { }
1668       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1669     }
1670   }
1671   \bool_if:NT \g__enumext_starred_bool
1672   {
1673     \tl_if_empty:NF \g__enumext_starred_series_tl
1674     {
1675       \__enumext_resume_counter:n { }
1676       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1677     }
1678   }
1679 }

```

(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*`.

```

1680 \cs_set_protected:Npn \__enumext_tmp:n #1
1681 {
1682   \keys_define:nn { enumext / #1 }
1683   {
1684     save-ans .code:n = \__enumext_storing_set:n {##1},
1685     save-ans .value_required:n = true,
1686   }
1687 }
1688 \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.

```

1689 \cs_new_protected:Npn \__enumext_storing_set:n #1
1690 {
1691   \tl_set:Nx \l__enumext_store_name_tl {#1}
1692   \tl_if_empty:NTF \l__enumext_store_name_tl
1693   {
1694     \bool_if:NT \l__enumext_standar_first_level_bool
1695     {
1696       \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1697     }
1698     \bool_if:NT \l__enumext_starred_first_level_bool
1699     {
1700       \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1701     }
1702   }
1703   {
1704     \bool_if:NT \l__enumext_standar_first_level_bool
1705     {
1706       \msg_note:nnnV
1707       { enumext } { save-ans-ok } { enumext } \l__enumext_store_name_tl
1708       \__enumext_storing_exec:
1709     }
1710     \bool_if:NT \l__enumext_starred_first_level_bool
1711     {
1712       \msg_note:nnnV
1713       { enumext } { save-ans-ok } { enumext* } \l__enumext_store_name_tl
1714       \__enumext_storing_exec:
1715     }
1716   }
1717 }

```

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*`.

```

1718 \cs_new_protected:Nn \__enumext_storing_exec:
1719 {
1720   \bool_set_true:N \l__enumext_store_active_bool
1721   \bool_set_true:N \l__enumext_store_ans_bool
1722   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1723   {
1724     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1725   }
1726   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1727   {
1728     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1729   }
1730   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1731   {
1732     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1733   }
1734 }

```

(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_count_item_number_int` must match the integer variable `\g__enumext_count_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

```

1735 \cs_set_protected:Npn \__enumext_tmp:n #1
1736 {
1737   \keys_define:nn { enumext / #1 }
1738   {
1739     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1740     check-ans .initial:n = false,
1741     check-ans .value_required:n = true,
1742     no-store .code:n = {
1743       \bool_set_false:N \l__enumext_store_ans_bool
1744       \bool_set_false:N \l__enumext_check_ans_bool
1745     },
1746     no-store .value_forbidden:n = true,
1747   }
1748 }
1749 \clist_map_inline:nn
1750 {
1751   level-1, level-2, level-3, level-4, enumext*
1752 }
1753 { \__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.

```

1754 \cs_new_protected:Nn \__enumext_check_ans:
1755 {
1756   \bool_if:NT \l__enumext_check_ans_bool
1757   {
1758     \tl_if_empty:NTF \l__enumext_store_name_tl
1759     {
1760       \bool_if:NT \l__enumext_standar_first_level_bool
1761       {
1762         \msg_error:nnnn { enumext } { need-save-ans } { check-ans } { enumext }
1763       }
1764       \bool_if:NT \l__enumext_starred_first_level_bool
1765       {
1766         \msg_error:nnnn { enumext } { need-save-ans } { check-ans } { enumext* }
1767       }
1768     }
1769   }
```

```

1770         \__enumext_check_ans_level:
1771     }
1772 }
1773 }

```

The function `__enumext_check_ans_level:` will decrement by “one” the value of the variable `\g__enumext_count_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.

```

1774 \cs_new_protected:Nn \__enumext_check_ans_level:
1775 {
1776   \int_case:nn { \l__enumext_level_int }
1777   {
1778     { 1 }{
1779       \bool_lazy_all:nT
1780       {
1781         { \bool_if_p:N \g__enumext_starred_bool }
1782         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1783       }
1784       {
1785         \int_gdecr:N \g__enumext_count_item_number_int
1786       }
1787     }
1788     { 2 }{
1789       \int_gdecr:N \g__enumext_count_item_number_int
1790     }
1791     { 3 }{
1792       \int_gdecr:N \g__enumext_count_item_number_int
1793     }
1794     { 4 }{
1795       \int_gdecr:N \g__enumext_count_item_number_int
1796     }
1797   }
1798   \int_case:nn { \l__enumext_level_h_int }
1799   {
1800     { 1 }{
1801       \bool_if:NT \g__enumext_standar_bool
1802       {
1803         \int_gdecr:N \g__enumext_count_item_number_int
1804       }
1805     }
1806   }
1807 }

```

(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 value of the local variable `\l__enumext_store_name_tl` saving it to the global variable `\g__enumext_store_name_tl` only if the key `check-ans` is active.

```

1808 \cs_new_protected:Nn \__enumext_check_ans_to_hook:
1809 {
1810   \bool_lazy_and:nnT
1811   { \bool_if_p:N \l__enumext_check_ans_bool }
1812   { \bool_if_p:N \g__enumext_standar_bool }
1813   {
1814     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1815   }
1816   \bool_lazy_and:nnT
1817   { \bool_if_p:N \l__enumext_check_ans_bool }
1818   { \bool_if_p:N \g__enumext_starred_bool }
1819   {
1820     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1821   }
1822 }

```

(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.


```

1823 \cs_new_protected:Nn \__enumext_check_ans_show:
1824 {
1825   \int_compare:nNt { \l__enumext_level_int } = { 0 }
1826   {
1827     \tl_if_empty:NF \g__enumext_store_name_tl
1828     {
1829       \int_compare:nNtF
1830       { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1831       {
1832         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1833       }
1834       {
1835         \msg_warning:nnVV
1836         { enumext } { item-different-answer }
1837         \g__enumext_store_name_tl \g__enumext_check_start_line_env_tl
1838       }
1839     }
1840     \int_gzero:N \g__enumext_count_item_number_int
1841     \int_gzero:N \g__enumext_count_item_anskey_int
1842     \bool_gset_false:N \g__enumext_standar_bool
1843     \bool_gset_false:N \g__enumext_starred_bool
1844     \tl_gclear:N \g__enumext_store_name_tl
1845     \tl_gclear:N \g__enumext_check_start_line_env_tl
1846   }
1847 }

```

(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.

```

1848 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1849 {
1850   \int_compare:nNt
1851   { \g__enumext_check_starred_cmd_int } = { 0 }
1852   {
1853     \msg_warning:nnnV
1854     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1855   }
1856   \int_compare:nNt
1857   { \g__enumext_check_starred_cmd_int } > { 1 }
1858   {
1859     \msg_warning:nnnV
1860     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1861   }
1862   \int_gzero:N \g__enumext_check_starred_cmd_int
1863   \tl_clear:N \l__enumext_check_start_line_env_tl
1864 }

```

(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*`.

```

1865 \cs_set_protected:Npn \__enumext_tmp:n #1
1866 {
1867   \keys_define:nn { enumext / #1 }
1868   {
1869     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1870     wrap-ans .initial:n = \fbox{##1},
1871     wrap-ans .value_required:n = true,
1872     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1873     wrap-opt .initial:n = [{##1}],
1874     wrap-opt .value_required:n = true,
1875     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1876     save-sep .initial:n = {, ~ },
1877     save-sep .value_required:n = true,
1878     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,

```

```

1879     mark-ans     .initial:n = \textasteriskcentered,
1880     mark-ans     .value_required:n = true,
1881     mark-pos     .choice:,
1882     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1883     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1884     mark-pos     .initial:n = right,
1885     mark-pos     .value_required:n = true,
1886     show-ans     .bool_set:N = \__enumext_show_answer_bool,
1887     show-ans     .initial:n = false,
1888     show-ans     .value_required:n = true,
1889     show-pos     .bool_set:N = \__enumext_show_position_bool,
1890     show-pos     .initial:n = false,
1891     show-pos     .value_required:n = true,
1892     mark-ref     .tl_set:N = \__enumext_mark_ref_sym_tl,
1893     mark-ref     .initial:n = \textasteriskcentered,
1894     mark-ref     .value_required:n = true,
1895     save-ref     .bool_set:N = \__enumext_store_ref_key_bool,
1896     save-ref     .initial:n = false,
1897     save-ref     .value_required:n = true,
1898   }
1899 }
1900 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for wrap-ans and others.)

mark-pos For the **keyans** and **keyans*** environments we will only add the keys mark-pos, show-ans and show-pos.

```

show-ans
show-pos 1901 \cs_set_protected:Npn \__enumext_tmp:n #1
1902 {
1903   \keys_define:nn { enumext / #1 }
1904   {
1905     mark-pos .choice:,
1906     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1907     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1908     mark-pos .initial:n = right,
1909     mark-pos .value_required:n = true,
1910     show-ans .bool_set:N = \__enumext_show_answer_bool,
1911     show-ans .initial:n = false,
1912     show-ans .value_required:n = true,
1913     show-pos .bool_set:N = \__enumext_show_position_bool,
1914     show-pos .initial:n = false,
1915     show-pos .value_required:n = true,
1916   }
1917 }
1918 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

columns* For the **enumext** and **enumext*** environments we will only add the keys **columns*** and **columns-sep***.
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.

```

1919 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1920 {
1921   \keys_define:nn { enumext / #1 }
1922   {
1923     columns* .code:n = \bool_set_true:c { \__enumext_store_columns_#2_bool }
1924               \int_set:cn { \__enumext_store_columns_#2_int } {##1}
1925               \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1926               {
1927                 columns = \exp_not:v { \__enumext_store_columns_#2_int },
1928               },
1929     columns* .value_required:n = true,
1930     columns-sep* .code:n = \bool_set_true:c { \__enumext_store_columns_sep_#2_bool }
1931               \dim_set:cn { \__enumext_store_columns_sep_#2_dim } {##1}
1932               \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1933               {
1934                 columns-sep = \exp_not:v { \__enumext_store_columns_sep_#2_dim },
1935               },
1936     columns-sep* .value_required:n = true,
1937   }

```

```

1938   }
1939   \clist_map_inline:nn
1940   {
1941     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1942   }
1943   { \__enumext_tmp:nn #1 }

```

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

10.24.1 Function for storing content in prop list

`__enumext_store_addto_prop:n` 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.

`__enumext_store_addto_prop:V`

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

```

1944 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1945 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1946 {
1947   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1948   {
1949     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1950   }
1951   { #1 }
1952 }
1953 \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` 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`.

`__enumext_store_addto_seq:v` 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.

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

```

1954 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1955 {
1956   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1957 }
1958 \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:` 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 `\l__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`.

```

1959 \cs_new_protected:Nn \__enumext_store_level_open:
1960 {
1961   \bool_if:NT \l__enumext_store_ans_bool
1962   {
1963     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1964     {
1965       \__enumext_store_addto_seq:n
1966       {
1967         \item \begin{enumext}
1968       }
1969     }
1970     {
1971       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1972       {
1973         \item \begin{enumext} [
1974       }
1975       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1976       {
1977

```

```

1978         }
1979         \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1980     }
1981 }
1982 }
1983 \cs_new_protected:Nn \__enumext_store_level_close:
1984 {
1985     \bool_if:NT \l__enumext_store_ans_bool
1986     {
1987         \__enumext_store_addto_seq:n { \end{enumext} }
1988     }
1989 }

```

(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.

```

1990 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1991 {
1992     \bool_if:NT \l__enumext_store_ans_bool
1993     {
1994         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1995         {
1996             \__enumext_store_addto_seq:n
1997             {
1998                 \item \mode_leave_vertical:
1999                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2000                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2001             }
2002         }
2003         {
2004             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
2005             {
2006                 \item \mode_leave_vertical:
2007                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2008                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2009             ]
2010             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
2011             {
2012                 ]
2013             }
2014             \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
2015         }
2016     }
2017 }
2018 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2019 {
2020     \bool_if:NT \l__enumext_store_ans_bool
2021     {
2022         \__enumext_store_addto_seq:n { \end{enumext*} }
2023     }
2024 }

```

(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`

```

2025 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2026 {
2027     \mode_leave_vertical:
2028     \skip_horizontal:n { -\dim_use:N #2 }
2029     \makebox[\opt][ r ]
2030     {

```

```

2031         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2032         {
2033             \tl_use:N \l__enumext_mark_answer_sym_tl
2034         }
2035     }
2036     \skip_horizontal:n { \dim_use:N #2 }
2037 }
2038 \cs_generate_variant:Nn \l__enumext_print_keyans_box:NN { cc }

```

(End of definition for `\l__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`.

```

2039 \keys_define:nn { enumext / anskey }
2040 {
2041     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2042     item-sym* .value_required:n = true,
2043     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2044     item-pos* .value_required:n = true,
2045     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2046     store-brk .default:n = true,
2047     store-brk .value_forbidden:n = true,
2048 }

```

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 `\l__enumext_store_anskey_code:nnnn` will “*store content*” in the `<sequence>` and in the `<prop list>`.

```

2049 \NewDocumentCommand \anskey { d() s o +m }
2050 {
2051     \bool_if:NF \l__enumext_store_active_bool
2052     {
2053         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2054     }
2055     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2056     {
2057         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2058     }
2059     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2060     {
2061         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2062     }
2063     \group_begin:
2064         \bool_if:NT \l__enumext_store_ans_bool
2065         {
2066             \bool_if:NT \l__enumext_check_ans_bool
2067             {
2068                 \int_gincr:N \g__enumext_count_item_anskey_int
2069             }
2070             \l__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2071         }
2072     \group_end:
2073 }

```

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

`\l__enumext_store_anskey_code:nnnn`

The internal function `\l__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 `\l__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.

```

2074 \cs_new_protected:Npn \l__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2075 {
2076     \l__enumext_store_addto_prop:n {#4}

```

```

2077 \bool_if:NT \l__enumext_store_ref_key_bool
2078 {
2079   \__enumext_store_internal_ref:
2080 }
2081 \__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`.

```

2082 \tl_clear:N \l__enumext_store_anskey_arg_tl
2083 \tl_if_novalue:nF {#3}
2084 {
2085   \keys_set:nn { enumext / anskey } {#3}
2086 }
2087 \bool_lazy_and:nnT
2088 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2089 { \bool_not_p:n { \l__enumext_starred_bool } }
2090 {
2091   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2092 }
2093 \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).

```

2094 \tl_if_novalue:nF {#1}
2095 {
2096   \int_set:Nn \l__enumext_store_columns_join_int {#1}
2097   \bool_if:NT \l__enumext_starred_bool
2098   {
2099     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2100     {
2101       ( \exp_not:V \l__enumext_store_columns_join_int )
2102     }
2103   }
2104 }

```

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`.

```

2105 \bool_if:nTF {#2}
2106 {
2107   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2108   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2109   {
2110     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2111     {
2112       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2113     }
2114   }
2115   \dim_compare:nT
2116   {
2117     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2118   }
2119   {
2120     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2121     {
2122       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2123     }
2124   }
2125   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2126 }
2127 {
2128   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2129 }

```

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`.

```

2130 \bool_lazy_and:nnT
2131 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2132 { \bool_if_p:N \l__enumext_hyperref_bool }
2133 {

```

```

2134         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2135         {
2136             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2137             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2138         }
2139     }
2140     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2141 }

```

(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`.

```

2142 \cs_new_protected:Nn \__enumext_store_internal_ref:
2143 {
2144     \cs_set_protected:Npn \__enumext_tmp:n ##1
2145     {
2146         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2147         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2148         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2149         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2150     }
2151     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2152     \cs_set:Npn \__enumext_tmp:n ##1
2153     { . \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.

```

2154     \bool_lazy_all:nT
2155     {
2156         { \bool_if_p:N \g__enumext_starred_bool }
2157         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2158     }
2159     {
2160         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2161         { \tl_use:N \l__enumext_label_copy_vii_tl }
2162     }
2163     \bool_lazy_all:nT
2164     {
2165         { \bool_if_p:N \l__enumext_standar_bool }
2166         { \bool_if_p:N \g__enumext_starred_bool }
2167         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2168     }
2169     {
2170         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2171         {
2172             \tl_use:N \l__enumext_label_copy_vii_tl
2173             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2174         }
2175     }

```

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.

```

2176     \bool_lazy_all:nT
2177     {
2178         { \bool_if_p:N \l__enumext_standar_bool }
2179         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2180         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2181         { \bool_not_p:n { \l__enumext_starred_bool } }
2182     }
2183     {
2184         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2185         {
2186             \tl_use:N \l__enumext_label_copy_i_tl
2187             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2188         }
2189     }
2190     \cs_set:Npn \__enumext_tmp:n ##1

```



```

2191     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2192 \bool_lazy_all:nT
2193 {
2194     { \bool_if:p:N \l__enumext_standar_bool }
2195     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2196     { \bool_not_p:n { \g__enumext_starred_bool } }
2197     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2198 }
2199 {
2200     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2201     {
2202         \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2203         . \tl_use:N \l__enumext_label_copy_vii_tl
2204     }
2205 }

```

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

```

2206 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2207 {
2208     \l__enumext_store_name_tl \c_colon_str
2209     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2210 }

```

Now execute the function `\l__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.

```

2211 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2212 {
2213     \l__enumext_newlabel:nn
2214     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2215     { \l__enumext_newlabel_arg_two_tl }
2216 }
2217 \l__enumext_store_write_aux_file_tl
2218 }

```

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

`\l__enumext_store_anskey_show_wrap:n`

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

```

2219 \cs_new_protected:Npn \l__enumext_store_anskey_show_wrap:n #1
2220 {
2221     \par
2222     \bool_if:NT \l__enumext_starred_bool
2223     {
2224         \cs_set:Nn \l__enumext_level: { vii }
2225     }
2226     \l__enumext_print_keyans_box:cc
2227     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2228     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2229     \l__enumext_anskey_wrapper:n { #1 }
2230 }

```

(End of definition for `\l__enumext_store_anskey_show_wrap:n`.)

`\l__enumext_store_anskey_show_left:n`

The function `\l__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.

```

2231 \cs_new_protected:Npn \l__enumext_store_anskey_show_left:n #1
2232 {
2233     \bool_if:NT \l__enumext_show_answer_bool
2234     {
2235         \l__enumext_store_anskey_show_wrap:n { #1 }
2236     }
2237     \bool_if:NT \l__enumext_show_position_bool
2238     {
2239         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2240         {
2241             \group_begin:
2242             \exp_not:N \normalfont
2243             \exp_not:N \footnotesize [ \int_eval:n

```

```

2244         {
2245             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2246         }
2247     ]
2248     \group_end:
2249 }
2250 \__enumext_store_anskey_show_wrap:n { #1 }
2251 }
2252 }

```

(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.

```

2253 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2254 {
2255     \tl_clear:N \l__enumext_store_keyans_label_tl
2256     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2257     {
2258         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2259     }
2260     {
2261         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2262     }
2263     \tl_if_novalue:nF { #1 }
2264     {
2265         % Set save-sep
2266         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2267         {
2268             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2269         }
2270         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2271     }
2272     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2273 }

```

(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:
 __enumext_keyans_store_ref_aux_i:
 __enumext_keyans_store_ref_aux_ii:

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

```

2274 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2275 {
2276     \bool_if:NT \l__enumext_store_ref_key_bool
2277     {
2278         \cs_set_protected:Npn \__enumext_tmp:n #1
2279         {
2280             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2281             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2282             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2283             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2284         }
2285         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2286         \__enumext_keyans_store_ref_aux_i:
2287     }
2288 }

```

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

```

2289 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2290 {
2291   \bool_if:NT \g__enumext_starred_bool
2292   {
2293     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2294   }
2295   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2296   {
2297     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2298     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2299   }
2300   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2301   {
2302     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2303     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2304   }
2305   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2306   {
2307     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2308     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2309   }
2310   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2311   {
2312     \l__enumext_store_name_tl \c_colon_str
2313     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2314   }
2315   \__enumext_keyans_store_ref_aux_ii:
2316 }

```

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.

```

2317 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2318 {
2319   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2320   {
2321     \__enumext_newlabel:nn
2322     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2323     { \l__enumext_newlabel_arg_two_tl }
2324   }
2325   \l__enumext_store_write_aux_file_tl
2326 }

```

(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 $\langle label \rangle$ `\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 $\langle contents \rangle$ 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.

```

2327 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2328 {
2329   \tl_clear:N \l__enumext_store_keyans_label_tl
2330   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2331   {
2332     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2333   }
2334   {
2335     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2336   }
2337   \tl_if_novalue:nF { #1 }
2338   {
2339     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2340     {
2341       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2342       {
2343         \l__enumext_store_keyans_item_opt_sep_tl

```

```

2344         }
2345     }
2346     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2347 }
2348 \__enumext_keyans_addto_seq_link:
2349 }

```

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_count_item_anskey_int` handled by the `check-ans` key.

```

2350 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2351 {
2352     \bool_lazy_and:nnT
2353     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2354     { \bool_if_p:N \l__enumext_hyperref_bool }
2355     {
2356         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2357         {
2358             \hfill \exp_not:N \hyperlink
2359             {
2360                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2361             }
2362             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2363         }
2364     }
2365     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2366     \bool_if:NT \l__enumext_check_ans_bool
2367     {
2368         \int_gincr:N \g__enumext_count_item_anskey_int
2369     }
2370 }

```

(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.

```

2371 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2372 {
2373     \tl_if_novalue:nF { #1 }
2374     {
2375         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2376     }
2377     \bool_if:NT \l__enumext_show_answer_bool
2378     {
2379         \__enumext_keyans_show_ans:
2380     }
2381     \bool_if:NT \l__enumext_show_position_bool
2382     {
2383         \__enumext_keyans_show_pos:
2384     }
2385 }
2386 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2387 {
2388     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2389     {
2390         \bool_lazy_or:nnT
2391         { \bool_if_p:N \l__enumext_show_answer_bool }
2392         { \bool_if_p:N \l__enumext_show_position_bool }
2393         {
2394             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2395         }
2396     }
2397 }
2398 \cs_new_protected:Nn \__enumext_keyans_show_ans:

```

```

2399 {
2400   \tl_put_left:Nn \l__enumext_label_v_tl
2401   {
2402     \__enumext_print_keyans_box:NN
2403     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2404   }
2405 }
2406 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2407 {
2408   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2409   {
2410     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2411     {
2412       \group_begin:
2413       \exp_not:N \normalfont
2414       \exp_not:N \footnotesize [ \int_eval:n
2415         {
2416           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2417         }
2418       ]
2419       \group_end:
2420     }
2421   }
2422   {
2423     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2424     {
2425       \group_begin:
2426       \exp_not:N \normalfont
2427       \exp_not:N \footnotesize [ \int_eval:n
2428         {
2429           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2430         }
2431       ]
2432       \group_end:
2433     }
2434   }
2435   \tl_put_left:Nn \l__enumext_label_v_tl
2436   {
2437     \__enumext_print_keyans_box:NN
2438     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2439   }
2440 }

```

(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*
2441 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2442 {
2443   \keys_define:nn { enumext / #1 }
2444   {
2445     item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2446     item-sym* .value_required:n = true,
2447     item-sym* .initial:n = { $\star$ },
2448     item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2449     item-pos* .value_required:n = true,
2450   }
2451 }
2452 \clist_map_inline:nn
2453 {
2454   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2455 }
2456 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining \footnote command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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](#).

```

2457 \cs_new_protected:Nn \__enumext_footnotetext:nn
2458 {
2459   \footnotetext[#1]{#2}
2460 }
2461 \cs_new_protected:Nn \__enumext_renew_footnote:
2462 {
2463   \seq_gclear:N \g__enumext_footnote_arg_seq
2464   \seq_gclear:N \g__enumext_footnote_int_seq
2465   \RenewDocumentCommand \footnote { o +m }
2466   {
2467     \tl_if_novalue:nTF {##1}
2468     {
2469       \stepcounter{footnote}
2470       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2471     }
2472     {
2473       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2474     }
2475     \footnotemark [ \g__enumext_footnote_int ]
2476     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2477     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2478   }
2479 }
2480 \cs_new_protected:Nn \__enumext_print_footnote:
2481 {
2482   \seq_if_empty:NF \g__enumext_footnote_int_seq
2483   {
2484     \seq_map_pairwise_function:NNN
2485     \g__enumext_footnote_int_seq
2486     \g__enumext_footnote_arg_seq
2487     \__enumext_footnotetext:nn
2488   }
2489 }

```

(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).

```

2490 \cs_new_protected:Npn \__enumext_default_item:n #1
2491 {
2492   \tl_if_novalue:nTF {#1}
2493   {
2494     \bool_if:NT \l__enumext_check_ans_bool
2495     {
2496       \int_gincr:N \g__enumext_count_item_number_int
2497     }
2498     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2499     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2500   }
2501   {
2502     \bool_set_eq:cc
2503     { l__enumext_wrap_label_ \__enumext_level: _bool }

```

```

2504         { \__enumext_wrap_label_opt_ \__enumext_level: _bool }
2505         \__enumext_item_std:w [#1] \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl
2506     }
2507 }

```

(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 `\label` separated from it by the value set by the `labelsep` key and can be *offset* using the second optional argument `[\langle offset \rangle]`.

#1: __enumext_item_symbol_X_tl

#2: __enumext_item_symbol_sep_X_dim

First we will make a copy of `__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 `__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 `__enumext_check_ans_bool` set by the key `check-ans`, set the boolean variable `__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 `__enumext_wrap_label_X_bool` and the vars `__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label:` (§10.30).

```

2508 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2509 {
2510     \tl_if_novalue:nF {#1}
2511     {
2512         \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2513     }
2514     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2515     \tl_if_novalue:nTF {#2}
2516     {
2517         \dim_set_eq:cc
2518         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2519         { \__enumext_labelsep_ \__enumext_level: _dim }
2520     }
2521     {
2522         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2523     }
2524     \bool_if:NT \__enumext_check_ans_bool
2525     {
2526         \int_gincr:N \g__enumext_count_item_number_int
2527     }
2528     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2529     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2530 }

```

(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).

```

2531 \cs_new_protected:Npn \__enumext_redefine_item:
2532 {
2533     \RenewDocumentCommand \item { s o o }
2534     {
2535         \bool_if:nTF {##1}
2536         {
2537             \__enumext_starred_item:nn {##2} {##3}
2538         }
2539         { \__enumext_default_item:n {##2} }
2540     }
2541 }

```

(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`.

```

2542 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2543 {
2544   \tl_if_novalue:nTF { #1 }
2545   {
2546     \bool_set_true:N \__enumext_wrap_label_v_bool
2547     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2548   }
2549   {
2550     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2551     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2552   }
2553 }

```

(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 $\langle label \rangle$.

```

2554 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2555 {
2556   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2557   \__enumext_keyans_show_left:n { #1 }
2558   \bool_set_true:N \__enumext_wrap_label_v_bool
2559   \__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 $\langle sequence \rangle$.

```

2560   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2561   \__enumext_keyans_addto_prop:n { #1 }
2562   \__enumext_keyans_store_ref:
2563   \__enumext_keyans_addto_seq:n { #1 }
2564   \int_gincr:N \g__enumext_check_starred_cmd_int
2565 }

```

(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).

```

2566 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2567 {
2568   \RenewDocumentCommand \item { s o }
2569   {
2570     \bool_if:nTF {##1}
2571     {
2572       \peek_remove_spaces:n
2573       {
2574         \__enumext_keyans_starred_item:n {##2}
2575       }
2576     }
2577     {
2578       \__enumext_keyans_default_item:n {##2}
2579     }
2580   }
2581 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 11.)

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.

```

2582 \cs_new_protected:Nn \__enumext_item_starred:
2583 {
2584   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2585   {
2586     \mode_leave_vertical:
2587     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2588     \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
2589     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2590   }
2591 }
```

(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).

```

2592 \cs_new_protected:Nn \__enumext_make_label:
2593 {
2594   \RenewDocumentCommand \makeLabel { m }
2595   {
2596     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2597     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2598     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2599     {
2600       \__enumext_item_starred:
2601       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2602     }
2603     { ##1 }
2604     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2605     \tl_gclear:N \g__enumext_item_symbol_tl
2606   }
2607 }
```

(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).

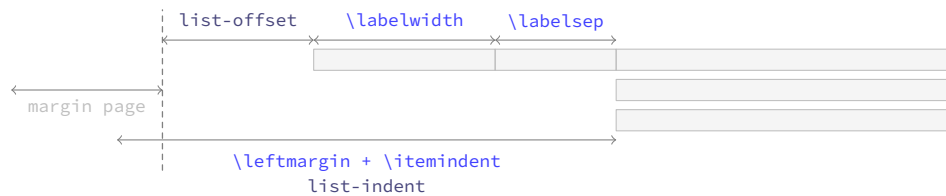
```

2608 \cs_new_protected:Nn \__enumext_keyans_make_label:
2609 {
2610   \RenewDocumentCommand \makeLabel { m }
2611   {
2612     \tl_use:N \l__enumext_label_fill_left_v_tl
2613     \tl_use:N \l__enumext_label_font_style_v_tl
2614     \bool_if:NTF \l__enumext_wrap_label_v_bool
2615     {
2616       \__enumext_wrapper_label_v:n { ##1 }
2617     }
2618     { ##1 }
2619     \tl_use:N \l__enumext_label_fill_right_v_tl
2620   }
2621 }
```

(End of definition for `__enumext_keyans_make_label:`)

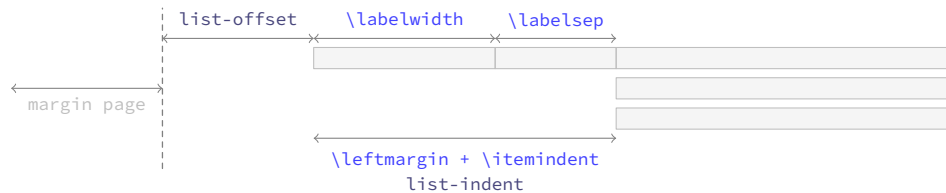
10.31 Second argument of the lists

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`.

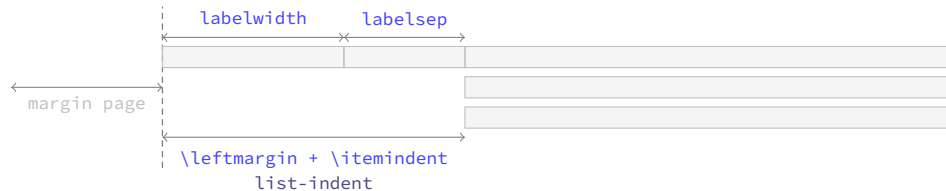
Figure 9: Representation of standard horizontal lengths in `list` environment.

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).

```
2622 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2623 {
2624   \dim_compare:nNt { #1 } < { \c_zero_dim }
2625   {
2626     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2627     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2628   }
2629   \dim_compare:nNt { #2 } < { \c_zero_dim }
2630   {
2631     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2632     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2633   }
2634 }
```

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

```
2634 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2635 \dim_compare:nNtF { #4 } < { \c_zero_dim }
2636 {
2637   \dim_set:Nn #6 { #1 + #2 - #4 }
2638   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2639 }
```

```

2640 {
2641   \dim_compare:nNnT { #4 } = { #1 + #2 }
2642   { \dim_set:Nn #6 { \c_zero_dim } }
2643   \dim_compare:nNnT { #4 } < { #1 + #2 }
2644   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2645   \dim_compare:nNnT { #4 } > { #1 + #2 }
2646   {
2647     \dim_set:Nn #6 { -#1 - #2 + #4 }
2648     \dim_set:Nn #6 { #6*-1 }
2649   }
2650   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2651 }
2652 }
2653 \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:
2654 \cs_set_protected:Npn \__enumext_tmp:n #1
2655 {
2656   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2657   {
2658     \__enumext_calc_hspace:ccccc
2659     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2660     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2661     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2662     { \__enumext_leftmargin_tmp_#1_bool }
2663     \clist_map_inline:nn
2664       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2665       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2666     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2667       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2668     \usecounter { enumX#1 }
2669     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2670     \str_if_eq:nnTF {#1} { v }
2671     {
2672       \__enumext_keyans_redefine_item:
2673       \__enumext_keyans_make_label:
2674       \__enumext_keyans_ref:
2675       \__enumext_keyans_fake_item:
2676       \bool_if:cT { \__enumext_show_length_#1_bool }
2677       {
2678         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2679       }
2680     }
2681     {
2682       \__enumext_redefine_item:
2683       \__enumext_make_label:
2684       \__enumext_standar_ref:
2685       \__enumext_fake_item:
2686       \bool_if:cT { \__enumext_show_length_#1_bool }
2687       {
2688         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2689       }
2690     }
2691   }
2692 }
2693 \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.

```

2694 \cs_set_protected:Npn \__enumext_tmp:n #1
2695 {
2696   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2697   {

```

```

2698 \__enumext_calc_hspace:ccccc
2699 { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2700 { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2701 { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2702 { \__enumext_leftmargin_tmp_#1_bool }
2703 \clist_map_inline:nn
2704 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2705 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2706 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2707 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2708 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2709 \skip_zero:N \partopsep
2710 \usecounter { enumX#1 }
2711 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2712 \__enumext_starred_ref:
2713 \str_if_eq:nnTF {#1} { vii }
2714 {
2715     \__enumext_fake_item_vii:
2716     \bool_if:cT { \__enumext_show_length_vii_bool }
2717     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2718 }
2719 {
2720     \__enumext_fake_item_viii:
2721     \bool_if:cT { \__enumext_show_length_#1_bool }
2722     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2723 }
2724 }
2725 }
2726 \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.

```

2727 \NewDocumentEnvironment{enumext}{ 0{ } }
2728 {
2729     \__enumext_safe_exec:
2730     \__enumext_parse_keys:n {#1}
2731     \__enumext_before_list:
2732     \__enumext_start_store_level:
2733     \__enumext_start_list:nn
2734     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2735     {
2736         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2737         \__enumext_before_keys_exec:
2738     }
2739     \__enumext_after_args_exec:
2740 }
2741 {
2742     \__enumext_stop_list:
2743     \__enumext_stop_store_level:
2744     \__enumext_after_list:
2745 }

```

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

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first execute the function `__enumext_is_not_nested:` which will set the variable `\g__enumext_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:`.

```

2746 \cs_new_protected:Nn \__enumext_safe_exec:
2747 {
2748     \__enumext_is_not_nested:
2749     \int_incr:N \l__enumext_level_int
2750     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2751     { \msg_fatal:nn { enumext } { list-too-deep } }
2752     \bool_set_true:N \l__enumext_standar_bool
2753     \__enumext_is_on_first_level:
2754 }

```

(End of definition for `__enumext_safe_exec:`)

`__enumext_parse_keys:n`

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

```

2755 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2756 {
2757   \tl_if_novalue:nF {#1}
2758   {
2759     \str_clear:N \l__enumext_series_str
2760     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2761     {
2762       \keys_set:nn { enumext / level-1 } {#1}
2763       \__enumext_parse_series:n {#1}
2764     }
2765     {
2766       \exp_args:Ne \keys_set:nn
2767       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2768     }
2769     \bool_if:NT \l__enumext_store_active_bool
2770     {
2771       \__enumext_parse_store_keys:n {#1}
2772     }
2773   }
2774 }

```

(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:`.

```

2775 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2776 {
2777   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2778   {
2779     \regex_match:nnT { \b columns\b } {#1}
2780     {
2781       \int_set_eq:cc
2782       { \l__enumext_store_columns_ \__enumext_level: _int }
2783       { \l__enumext_columns_ \__enumext_level: _int }
2784       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2785       {
2786         columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2787       }
2788     }
2789   }
2790   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2791   {
2792     \regex_match:nnT { \b columns-sep\b } {#1}
2793     {
2794       \dim_set_eq:cc
2795       { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2796       { \l__enumext_columns_sep_ \__enumext_level: _dim }
2797       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2798       {
2799         columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2800       }
2801     }
2802   }
2803 }

```

(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.

```

2804 \cs_new_protected:Nn \__enumext_start_store_level:
2805 {
2806   \bool_lazy_all:nT
2807   {
2808     { \bool_if_p:N \l__enumext_store_active_bool }
2809     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2810     { \bool_not_p:n { \g__enumext_starred_bool } }
2811   }
2812   {
2813     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2814     {
2815       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2816       \__enumext_store_level_open:
2817     }
2818   }
2819   \bool_lazy_all:nT
2820   {
2821     { \bool_if_p:N \l__enumext_store_active_bool }
2822     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2823     { \bool_if_p:N \g__enumext_starred_bool }
2824   }
2825   {
2826     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2827     {
2828       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2829       \__enumext_store_level_open:
2830     }
2831   }
2832 }
2833 \cs_new_protected:Nn \__enumext_stop_store_level:
2834 {
2835   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2836   {
2837     \__enumext_store_level_close:
2838   }
2839 }

```

(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.

```

2840 \cs_new_protected:Nn \__enumext_before_list:
2841 {
2842   \__enumext_vspace_above:
2843   \__enumext_before_args_exec:

```

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

```

2844   \__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.

```

2845   \dim_compare:nNnT
2846   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2847   {
2848     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2849     {
2850       \linewidth
2851       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2852       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2853     }

```

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*.

```

2854         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2855         \int_gincr:N \g__enumext_minipage_stat_int
2856         \__enumext_mini_addvspace:
2857         \nointerlineskip\noindent
2858         \begin{\__enumext_mini_env*}
2859         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2860     }
2861     \__enumext_multicols_start:
2862 }

```

(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.

```

2863 \cs_new_protected:Nn \__enumext_multicols_start:
2864 {
2865     \int_compare:nNt
2866     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2867     {
2868         \dim_compare:nNt
2869         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2870         {
2871             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2872             {
2873                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2874                 + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2875                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2876                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2877             }
2878         }
2879         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2880         \skip_zero:N \multicolsep
2881         \int_compare:nNt { \l__enumext_level_int } > { 1 }
2882         {
2883             \dim_zero:N \columnseprule
2884         }
2885     }
2886 }

```

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.

```

2885         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2886         {
2887             \__enumext_multi_addvspace:
2888         }
2889         \raggedcolumns
2890         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2891     }
2892 }

```

(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.

```

2893 \cs_new_protected:Nn \__enumext_multicols_stop:
2894 {
2895     \int_compare:nNt
2896     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2897     {
2898         \end{multicols}
2899         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2900         {
2901             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2902         }
2903     }
2904 }

```

```

2903     }
2904 }

```

(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.

```

2905 \cs_new_protected:Nn \__enumext_after_list:
2906 {
2907   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2908   {
2909     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2910     {
2911       \msg_warning:nn { enumext } { missing-miniright }
2912       \miniright
2913     }
2914     \int_gzero:N \g__enumext_minipage_stat_int
2915     \end{__enumext_mini_env*}
2916     \par\addvspace { \l__enumext_minipage_after_skip }
2917   }
2918   { \__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`.

```

2919   \__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.

```

2920   \__enumext_after_stop_list:
2921   \__enumext_vspace_below:
2922   \bool_set_false:N \l__enumext_standar_bool
2923   \__enumext_resume_save_counter:
2924 }

```

(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`.

```

2925 \__enumext_after_env:nn {enumext} { \__enumext_check_ans_show: }

```

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.

```

2926 \NewDocumentEnvironment{keyans}{ 0{ } }
2927 {
2928   \__enumext_keyans_safe_exec:
2929   \__enumext_keyans_parse_keys:n {#1}
2930   \__enumext_before_list_v:
2931   \__enumext_start_list:nn
2932   { \tl_use:N \l__enumext_label_v_tl }
2933   {
2934     \__enumext_list_arg_two_v:
2935     \__enumext_before_keys_exec_v:
2936   }
2937   \__enumext_after_args_exec_v:
2938 }
2939 {
2940   \__enumext_check_starred_cmd:n { item }
2941   \__enumext_stop_list:
2942   \__enumext_after_list_v:
2943 }

```

(End of definition for `keyans`. This function is documented on page 11.)

`__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.

```

2944 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2945 {
2946   \bool_if:NF \l__enumext_store_active_bool
2947   {
2948     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
2949   }
2950   \int_incr:N \l__enumext_keyans_level_int
2951   \bool_set_true:N \l__enumext_keyans_env_bool
2952   \__enumext_keyans_save_start_line:
2953   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2954   \bool_set_false:N \l__enumext_store_active_bool
2955   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2956   {
2957     \msg_error:nn { enumext } { keyans-nested }
2958   }
2959   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2960   {
2961     \msg_error:nn { enumext } { keyans-wrong-level }
2962   }
2963 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

2964 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2965 {
2966   \keys_set:nn { enumext / keyans } { #1 }
2967 }

```

(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.

```

2968 \cs_new_protected:Nn \__enumext_before_list_v:
2969 {
2970   \__enumext_vspace_above_v:
2971   \__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.

```

2972   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2973   {
2974     \dim_set:Nn \l__enumext_minipage_left_v_dim
2975     {
2976       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2977     }

```

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

```

2978     \bool_set_true:N \l__enumext_minipage_active_v_bool
2979     \int_gincr:N \g__enumext_minipage_stat_int
2980     \__enumext_keyans_mini_addvspace:
2981     \nointerlineskip\noindent
2982     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2983   }

```

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

```

2984   \__enumext_keyans_multicols_start:
2985 }

```

(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.

```

2986 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2987 {
2988   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
2989   {
2990     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
2991     {
2992       \dim_set:Nn \__enumext_columns_sep_v_dim
2993       {
2994         (
2995           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
2996         ) / \__enumext_columns_v_int
2997         - \__enumext_listoffset_v_dim
2998       }
2999     }
3000     \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).

```

3001     \skip_zero:N \multicolsep
3002     \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.

```

3003     \bool_if:NF \__enumext_minipage_active_v_bool
3004     {
3005       \__enumext_keyans_multi_addvspace:
3006     }
3007     \raggedcolumns
3008     \begin{multicols}{\__enumext_columns_v_int}
3009   }
3010 }

```

(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.

```

3011 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3012 {
3013   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3014   {
3015     \end{multicols}
3016     \bool_if:NF \__enumext_minipage_active_v_bool
3017     {
3018       \par\addvspace{\__enumext_multicols_below_v_skip}
3019     }
3020   }
3021 }

```

(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.

```

3022 \cs_new_protected:Nn \__enumext_after_list_v:
3023 {
3024   \bool_if:NTF \__enumext_minipage_active_v_bool
3025   {
3026     \int_compare:nNt { \__enumext_minipage_stat_int } = { 1 }
3027     {
3028       \msg_warning:nn { enumext } { missing-miniright }
3029       \miniright

```

```

3030     }
3031     \int_gzero:N \g__enumext_minipage_stat_int
3032     \end{__enumext_mini_env*}
3033     \par\addvspace{ \l__enumext_minipage_after_skip }
3034   }
3035   { __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.

```

3036     \bool_set_false:N \l__enumext_keyans_env_bool
3037     __enumext_after_stop_list_v:
3038     __enumext_vspace_below_v:
3039   }

```

(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.

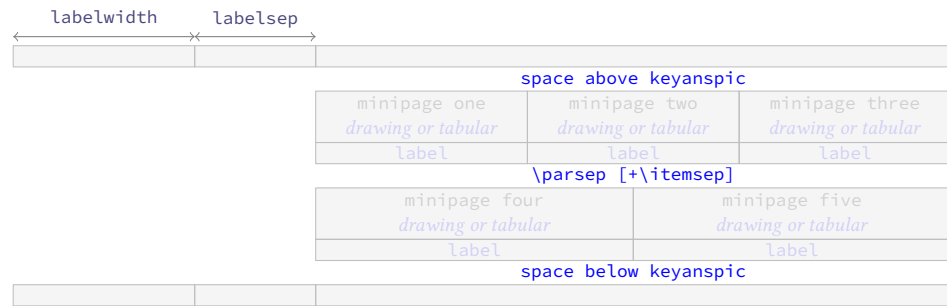


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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.

```

3040 \NewDocumentCommand \anspic { s o +m }
3041 {

```

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

```

3042   \bool_if:NF \l__enumext_store_active_bool
3043   {
3044     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3045   }
3046   \int_compare:nNt { \l__enumext_level_int } > { 1 }
3047   {
3048     \msg_error:nn { enumext } { keyanspic-wrong-level }
3049   }
3050   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3051   {
3052     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3053   }

```

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.

```

3054   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3055   {
3056     __enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3057   }
3058 }

```

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

_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.

```

3059 \cs_new_protected:Nn \_enumext_keyans_anspic_code:nnn
3060 {
3061   \stepcounter { enumXvi }
3062   #3 \\\
3063   \bool_if:nT { #1 }
3064   {
3065     \_enumext_keyans_addto_prop:n { #2 }
3066     \_enumext_keyans_store_ref:
3067     \_enumext_keyans_addto_seq:n { #2 }
3068     \int_gincr:N \g__enumext_check_starred_cmd_int
3069     \bool_lazy_or:nnT
3070     { \bool_if_p:N \l__enumext_show_answer_bool }
3071     { \bool_if_p:N \l__enumext_show_position_bool }
3072     {
3073       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3074       \_enumext_keyans_show_left:n { #2 }
3075       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3076     }
3077   }
3078   \tl_use:N \l__enumext_label_font_style_v_tl
3079   \_enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \_enumext_keyans_show_item_opt:
3080 }

```

(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.

```

3081 \NewDocumentEnvironment{keyanspic}{ o }
3082 {
3083   \_enumext_keyans_pic_safe_exec:
3084   \_enumext_start_list:nn
3085   { }
3086   {
3087     \_enumext_keyans_pic_arg_two:
3088   }

```

We apply the “adjusted” vertical spacing above the environment

```

3089   \vspace { \l__enumext_keyans_pic_above_skip }
3090 }

```

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.

```

3091 {
3092   \tl_if_novalue:nTF { #1 }
3093   {
3094     \_enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3095   }
3096   { \_enumext_keyans_pic_do:n { #1 } }
3097   \_enumext_stop_list:
3098   \_enumext_check_starred_cmd:n { anspic }
3099   \setcounter { enumXvi } { 0 }
3100   \vspace { \l__enumext_topsep_v_skip }
3101   %\bool_set_false:N \l__enumext_store_active_bool
3102 }

```

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

_enumext_keyans_pic_safe_exec:

The function _enumext_keyans_pic_safe_exec: check nested and level position inside the *enumext* environment.

```

3103 \cs_new_protected:Nn \_enumext_keyans_pic_safe_exec:
3104 {
3105   \int_incr:N \l__enumext_keyans_pic_level_int
3106   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3107   {
3108     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3109     }
3110     \__enumext_keyans_save_start_line:
3111 }

```

(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.

```

3112 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3113 {
3114     \dim_compare:nNnT { #1 } < { 0pt }
3115     { \skip_set:Nn #1 { -#1 } }
3116 }

```

(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.

```

3117 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3118 {

```

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.

```

3119     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3120     \__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.

```

3121     \skip_add:Nn \parsep { \itemsep }
3122     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3123     \dim_zero:N \labelwidth
3124     \dim_zero:N \listparindent
3125     \dim_zero:N \labelsep
3126     \skip_zero:N \partopsep
3127     \skip_zero:N \itemsep

```

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

```

3128     \__enumext_keyans_pic_skip_abs:N \parsep
3129     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3130     {
3131         \box_dp:N \strutbox
3132         + \l__enumext_topsep_v_skip
3133         - \parsep
3134     }
3135     \__enumext_item_std:w \scan_stop:
3136 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

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

```

3137 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3138 {
3139     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3140 }
3141 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

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

```

3142 \cs_new_protected:Npn \__enumext_keyans_pic_row:n
3143 {
3144     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3145     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3146     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }

```



```

3147 \int_step_inline:nnn
3148 { \l__enumext_keyans_pic_above_int + 1 }
3149 { \l__enumext_keyans_pic_below_int }
3150 {
3151   \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3152   \centering
3153   \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3154   \__enumext_endminipage:
3155 }
3156 \par
3157 }

```

(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.

```

3158 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3159 {
3160   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3161   {
3162     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3163     {
3164       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3165       / \l__enumext_columns_vii_int
3166     }
3167   }
3168   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3169   \dim_set:Nn \l__enumext_item_width_vii_dim
3170   {
3171     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3172     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3173     - \l__enumext_labelsep_vii_dim
3174   }
3175   \dim_zero_new:N \itemwidth
3176 }

```

(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`.

```

3177 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3178 {
3179   \int_set:Nn \l__enumext_joined_item_vii_int { #1 }
3180   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3181   {
3182     \msg_warning:nnee { enumext } { item-joined }
3183     { \int_use:N \l__enumext_joined_item_vii_int }
3184     { \int_use:N \l__enumext_columns_vii_int }
3185     \int_set:Nn \l__enumext_joined_item_vii_int
3186     {
3187       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3188     }
3189   }
3190   \int_compare:nNnT
3191   { \l__enumext_joined_item_vii_int }
3192   >
3193   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```

```

3194 {
3195   \msg_warning:nnee { enumext } { item-joined-columns }
3196   { \int_use:N \l__enumext_joined_item_vii_int }
3197   {
3198     \int_eval:n
3199     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3200   }
3201   \int_set:Nn \l__enumext_joined_item_vii_int
3202   {
3203     \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3204   }
3205 }

```

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

```

3206 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3207 {
3208   \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3209   \int_decr:N \l__enumext_joined_item_aux_vii_int
3210   \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3211   \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3212   \dim_set:Nn \l__enumext_joined_width_vii_dim
3213   {
3214     \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3215     + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3216       + \l__enumext_columns_sep_vii_dim
3217     ) * \l__enumext_joined_item_aux_vii_int
3218   }
3219   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3220 }
3221 {
3222   \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3223   \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3224 }
3225 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n.)

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

```

3226 \cs_new_protected:Nn \__enumext_start_mini_vii:
3227 {
3228   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3229   {
3230     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3231     {
3232       \linewidth
3233       - \l__enumext_minipage_right_vii_dim
3234       - \l__enumext_minipage_hsep_vii_dim
3235     }
3236     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3237     \dim_gset_eq:NN
3238     \g__enumext_minipage_right_vii_dim
3239     \l__enumext_minipage_right_vii_dim
3240     \__enumext_mini_addvspace_vii:
3241     \nointerlineskip\noindent
3242     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3243   }
3244 }

```

(End of definition for __enumext_start_mini_vii:.)

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

```

3245 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3246 {
3247   \bool_if:NT \l__enumext_minipage_active_vii_bool

```

```

3248     {
3249         \end{__enumext_mini_env*}
3250         \hfill
3251         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3252     }
3253 }

```

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”.

```

3254 \__enumext_after_env:nn {enumext*}
3255 {
3256     \bool_if:NT \g__enumext_minipage_active_vii_bool
3257     {
3258         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3259         \par\addvspace { \g__enumext_minipage_right_skip }
3260         \bool_if:NF \g__enumext_minipage_center_vii_bool
3261         {
3262             \centering
3263         }
3264         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3265         \end{__enumext_mini_env*}
3266         \par\addvspace{ \g__enumext_minipage_after_skip }
3267     }
3268     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3269     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3270     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3271     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3272     \bool_gset_false:N \g__enumext_starred_bool
3273 }

```

(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.

```

3274 \NewDocumentEnvironment{enumext*}{ o }
3275 {
3276     \__enumext_safe_exec_vii:
3277     \__enumext_parse_keys_vii:n {#1}
3278     \__enumext_before_list_vii:
3279     \__enumext_start_store_level_vii:
3280     \__enumext_start_list:nn { }
3281     {
3282         \__enumext_list_arg_two_vii:
3283         \__enumext_before_keys_exec_vii:
3284     }
3285     \__enumext_starred_columns_set_vii:
3286     \item[] \scan_stop:
3287     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3288     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3289 }
3290 {
3291     \__enumext_stop_item_tmp_vii:
3292     \__enumext_remove_extra_parsep_vii:
3293     \__enumext_stop_list:
3294     \__enumext_stop_store_level_vii:
3295     \__enumext_after_list_vii:
3296 }

```

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

`__enumext_safe_exec_vii:` First check the maximum nesting level for the `enumext*` environment then set the vars `\l__enumext_starred_bool` and `\g__enumext_starred_bool`.

```

3297 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3298 {
3299     \__enumext_is_not_nested:
3300     \int_incr:N \l__enumext_level_h_int
3301     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3302     {
3303         \msg_error:nn { enumext } { nested }
3304     }

```

```

3305     \bool_set_true:N \l__enumext_starred_bool
3306     \l__enumext_is_on_first_level:
3307 }

```

(End of definition for \l__enumext_safe_exec_vii:.)

\l__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 `\l__enumext_parse_store_keys_vii:n` and reprocess the keys to pass them to the storage sequence.

```

3308 \cs_new_protected:Npn \l__enumext_parse_keys_vii:n #1
3309 {
3310     \tl_if_novalue:NF {#1}
3311     {
3312         \str_clear:N \l__enumext_series_str
3313         \keys_set:nn { enumext / enumext* } {#1}
3314         \l__enumext_parse_series:n {#1}
3315         \bool_if:NT \l__enumext_store_active_bool
3316         {
3317             \l__enumext_parse_store_keys_vii:n {#1}
3318         }
3319     }
3320 }

```

(End of definition for \l__enumext_parse_keys_vii:n.)

\l__enumext_parse_store_keys_vii:n

The function `\l__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 `\l__enumext_store_level_open_vii:.`

```

3321 \cs_new_protected:Npn \l__enumext_parse_store_keys_vii:n #1
3322 {
3323     \bool_if:NF \l__enumext_store_columns_vii_bool
3324     {
3325         \regex_match:nnT { \b columns\b } {#1}
3326         {
3327             \int_set_eq:NN
3328             \l__enumext_store_columns_vii_int
3329             \l__enumext_columns_vii_int
3330             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3331             {
3332                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3333             }
3334         }
3335     }
3336     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3337     {
3338         \regex_match:nnT { \b columns-sep\b } {#1}
3339         {
3340             \dim_set_eq:NN
3341             \l__enumext_store_columns_sep_vii_dim
3342             \l__enumext_columns_sep_vii_dim
3343             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3344             {
3345                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3346             }
3347         }
3348     }
3349 }

```

(End of definition for \l__enumext_parse_store_keys_vii:n.)

\l__enumext_before_list_vii:

The function `\l__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 `\l__enumext_start_mini_vii:` handle by `mini-env`.

```

3350 \cs_new_protected:Npn \l__enumext_before_list_vii:
3351 {
3352     \l__enumext_vspace_above_vii:
3353     \l__enumext_check_ans: % need by chek-ans
3354     \l__enumext_before_args_exec_vii:
3355     \l__enumext_start_mini_vii:
3356 }

```

(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.

```

3357 \cs_new_protected:Nn \__enumext_after_list_vii:
3358 {
3359     \__enumext_stop_mini_vii:
3360     \__enumext_after_stop_list_vii:
3361     \__enumext_check_ans_to_hook:
3362     \__enumext_vspace_below_vii:
3363     \bool_set_false:N \l__enumext_starred_bool
3364     \__enumext_resume_save_counter:
3365 }

```

(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:`

```

3366 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3367 {
3368     \bool_if:NT \l__enumext_store_active_bool
3369     {
3370         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3371         {
3372             \__enumext_store_level_open_vii:
3373         }
3374     }
3375 }
3376 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3377 {
3378     \bool_if:NT \l__enumext_store_active_bool
3379     {
3380         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3381         {
3382             \__enumext_store_level_close_vii:
3383         }
3384     }
3385 }

```

(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`.

```

3386 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3387 {
3388     \__enumext_stop_item_tmp_vii:
3389     \int_incr:N \l__enumext_item_column_pos_vii_int
3390     \int_gincr:N \g__enumext_item_count_all_vii_int
3391     \__enumext_item_peek_args_vii:
3392 }

```

(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).

```

3393 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3394 {
3395     \peek_meaning:NTF (
3396         { \__enumext_joined_item_vii:w }

```

```

3397     { \__enumext_joined_item_vii:w (1) }
3398   }

```

(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.

```

3399 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3400 {
3401   \__enumext_starred_joined_item_vii:n {#1}
3402   \peek_meaning_remove:NTF *
3403   { \__enumext_starred_item_vii:w }
3404   { \__enumext_standard_item_vii:w }
3405 }

```

(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].

```

3406 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3407 {
3408   \bool_set_false:N \l__enumext_item_starred_vii_bool
3409   \peek_meaning:NTF [
3410   {
3411     \bool_set_eq:NN
3412       \l__enumext_wrap_label_vii_bool
3413       \l__enumext_wrap_label_opt_vii_bool
3414     \__enumext_start_item_vii:w
3415   }
3416   {
3417     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3418     \legacy_if_set_true:n { @noitemarg }
3419     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3420   }
3421 }

```

(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⟩].

```

3422 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3423 {
3424   \bool_set_true:N \l__enumext_item_starred_vii_bool
3425   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3426   \peek_meaning:NTF [
3427   { \__enumext_starred_item_vii_aux_i:w }
3428   { \__enumext_starred_item_vii_aux_ii:w }
3429 }
3430 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3431 {
3432   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3433   \__enumext_starred_item_vii_aux_ii:w
3434 }
3435 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3436 {
3437   \peek_meaning:NTF [
3438   { \__enumext_starred_item_vii_aux_iii:w }
3439   {
3440     \dim_set_eq:NN
3441       \l__enumext_item_symbol_sep_vii_dim
3442       \l__enumext_labelsep_vii_dim
3443     \legacy_if_set_true:n { @noitemarg }
3444     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]

```

```

3445     }
3446   }
3447   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3448   {
3449     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3450     \legacy_if_set_true:n { @noitemarg }
3451     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3452   }

```

(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.

```

3453   \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3454   {
3455     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3456     \legacy_if:nT { @noitemarg }
3457     {
3458       \legacy_if_set_false:n { @noitemarg }
3459       \legacy_if:nT { @nmbrlist }
3460       {
3461         \bool_if:NT \l__enumext_hyperref_bool
3462         {
3463           \legacy_if_set_true:n { @hyper@item }
3464         }
3465         \refstepcounter{enumXvii}
3466         \bool_if:NT \l__enumext_check_ans_bool
3467         {
3468           \int_gincr:N \g__enumext_count_item_number_int
3469         }
3470       }
3471     }

```

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.

```

3472   \group_begin:
3473   \lrbox{ \l__enumext_item_text_vii_box }
3474   \bool_if:NF \l__enumext_footnotes_key_bool
3475   {
3476     \__enumext_renew_footnote:
3477   }
3478   \bool_if:NT \l__enumext_item_starred_vii_bool
3479   {
3480     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3481     {
3482       \tl_gset_eq:NN
3483       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3484     }
3485     \mode_leave_vertical:
3486     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3487     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3488     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3489     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3490   }
3491   \group_begin:
3492   \tl_use:N \l__enumext_label_font_style_vii_tl
3493   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3494   {
3495     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]

```



```

3496         { \__enumext_wrapper_label_vii:n {#1} }
3497     }
3498     {
3499         \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3500     }
3501     \group_end:
3502     \skip_horizontal:N \__enumext_labelsep_vii_dim
3503     \tl_use:N \__enumext_after_list_args_vii_tl
3504     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3505         \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3506         \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3507         \tl_use:N \__enumext_fake_item_indent_vii_tl
3508     }

```

(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.

```

3509 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3510 {
3511     \__enumext_endminipage:
3512     \endlrbox
3513     \group_end:
3514     \box_set_wd:Nn \__enumext_item_text_vii_box
3515     {
3516         \__enumext_joined_width_vii_dim
3517         + \__enumext_labelwidth_vii_dim
3518         + \__enumext_labelsep_vii_dim
3519     }
3520     \int_set:Nn \hbadness { 10000 }
3521     \box_use:N \__enumext_item_text_vii_box
3522     \bool_if:NF \__enumext_footnotes_key_bool
3523     {
3524         \__enumext_print_footnote:
3525     }
3526     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3527     {
3528         \par\noindent
3529         \int_zero:N \__enumext_item_column_pos_vii_int
3530     }
3531     { \hspace{ \__enumext_columns_sep_vii_dim } }
3532 }

```

(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.

```

3533 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3534 {
3535     \int_compare:nNnT
3536     {
3537         \int_mod:nn { \__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3538     }
3539     =
3540     { \c_zero_int }
3541     {
3542         \par
3543         \vspace{ -\__enumext_itemsep_vii_skip }
3544         \int_gzero:N \__enumext_item_count_all_vii_int
3545     }
3546 }

```

(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.

```

3547 \__enumext_after_env:nn {enumext*} { \__enumext_check_ans_show: }

```

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.

```

3548 \cs_new_protected:Nn \_enumext_starred_columns_set_viii:
3549 {
3550   \dim_compare:nNt { \_enumext_columns_sep_viii_dim } = { \c_zero_dim }
3551   {
3552     \dim_set:Nn \_enumext_columns_sep_viii_dim
3553     {
3554       ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim )
3555       / \_enumext_columns_viii_int
3556     }
3557   }
3558   \int_set:Nn \_enumext_tmpa_viii_int { \_enumext_columns_viii_int - \c_one_int }
3559   \dim_set:Nn \_enumext_item_width_viii_dim
3560   {
3561     ( \linewidth - \_enumext_columns_sep_viii_dim * \_enumext_tmpa_viii_int )
3562     / \_enumext_columns_viii_int - \_enumext_labelwidth_viii_dim
3563     - \_enumext_labelsep_viii_dim
3564   }
3565   \dim_zero_new:N \itemwidth
3566 }

```

(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`.

```

3567 \cs_new_protected:Npn \_enumext_starred_joined_item_viii:n #1
3568 {
3569   \int_set:Nn \_enumext_joined_item_viii_int {#1}
3570   \int_compare:nNt { \_enumext_joined_item_viii_int } > { \_enumext_columns_viii_int }
3571   {
3572     \msg_warning:nnee { enumext } { item-joined }
3573     { \int_use:N \_enumext_joined_item_viii_int }
3574     { \int_use:N \_enumext_columns_viii_int }
3575     \int_set:Nn \_enumext_joined_item_viii_int
3576     {
3577       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3578     }
3579   }
3580   \int_compare:nNt
3581   { \_enumext_joined_item_viii_int }
3582   >
3583   { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3584   {
3585     \msg_warning:nnee { enumext } { item-joined-columns }
3586     { \int_use:N \_enumext_joined_item_viii_int }
3587     {
3588       \int_eval:n
3589       { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3590     }
3591     \int_set:Nn \_enumext_joined_item_viii_int
3592     {
3593       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3594     }
3595   }
3596   \int_compare:nNtTF { \_enumext_joined_item_viii_int } > { \c_one_int }
3597   {
3598     \int_set_eq:NN \_enumext_joined_item_aux_viii_int \_enumext_joined_item_viii_int
3599     \int_decr:N \_enumext_joined_item_aux_viii_int
3600     \int_add:Nn \_enumext_item_column_pos_viii_int { \_enumext_joined_item_aux_viii_int }
3601     \int_gadd:Nn \_enumext_item_count_all_viii_int { \_enumext_joined_item_aux_viii_int }
3602     \dim_set:Nn \_enumext_joined_width_viii_dim
3603     {
3604       \_enumext_item_width_viii_dim * \_enumext_joined_item_viii_int
3605       + ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim
3606         + \_enumext_columns_sep_viii_dim
3607         ) * \_enumext_joined_item_aux_viii_int
3608     }

```

```

3609         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3610     }
3611     {
3612         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3613         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3614     }
3615 }

```

(End of definition for __enumext_starred_joined_item_viii:n)

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

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:
3616 \cs_new_protected:Nn \__enumext_start_mini_viii:
3617 {
3618     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3619     {
3620         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3621         {
3622             \linewidth
3623             - \l__enumext_minipage_right_viii_dim
3624             - \l__enumext_minipage_hsep_viii_dim
3625         }
3626         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3627         \dim_gset_eq:NN
3628             \g__enumext_minipage_right_viii_dim
3629             \l__enumext_minipage_right_viii_dim
3630         \__enumext_mini_addvspace_viii:
3631         \nointerlineskip\noindent
3632         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3633     }
3634 }
3635 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3636 {
3637     \bool_if:NT \l__enumext_minipage_active_viii_bool
3638     {
3639         \end{__enumext_mini_env*}
3640         \hfill
3641         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3642     }
3643 }
3644 \__enumext_after_env:nn {keyans*}
3645 {
3646     \bool_if:NT \g__enumext_minipage_active_viii_bool
3647     {
3648         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3649         \par\addvspace { \g__enumext_minipage_right_skip }
3650         \bool_if:NF \g__enumext_minipage_center_viii_bool
3651         {
3652             \centering
3653         }
3654         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3655         \end{__enumext_mini_env*}
3656         \par\addvspace{ \g__enumext_minipage_after_skip }
3657     }
3658     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3659     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3660     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3661     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3662 }

```

(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.

```

3663 \NewDocumentEnvironment{keyans*}{ o }
3664 {
3665     \__enumext_safe_exec_viii:
3666     \__enumext_parse_keys_viii:n {#1}
3667     \__enumext_before_list_viii:
3668     \__enumext_start_list:nn { }

```

```

3669     {
3670         \__enumext_list_arg_two_viii:
3671         \__enumext_before_keys_exec_viii:
3672     }
3673     \__enumext_starred_columns_set_viii:
3674     \item[] \scan_stop:
3675     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3676     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3677 }
3678 {
3679     \__enumext_stop_item_tmp_viii:
3680     \__enumext_remove_extra_parsep_viii:
3681     \__enumext_check_starred_cmd:n { item }
3682     \__enumext_stop_list:
3683     \__enumext_after_list_viii:
3684 }

```

(End of definition for `keyans*`. This function is documented on page 11.)

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

```

3685 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3686 {
3687     \int_incr:N \__enumext_keyans_level_h_int
3688     \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
3689     {
3690         \msg_error:nn { enumext } { nested }
3691     }
3692     \__enumext_keyans_save_start_line:
3693     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3694     \bool_set_false:N \__enumext_store_active_bool
3695     \int_compare:nNnT { \__enumext_level_int } > { 1 }
3696     {
3697         \msg_error:nn { enumext } { keyans-wrong-level }
3698     }
3699 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

3700 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3701 {
3702     \tl_if_novalue:nF {#1}
3703     {
3704         \keys_set:nn { enumext / keyans* } {#1}
3705     }
3706 }

```

(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`.

```

3707 \cs_new_protected:Nn \__enumext_before_list_viii:
3708 {
3709     \__enumext_vspace_above_viii:
3710     \__enumext_before_args_exec_viii:
3711     \__enumext_start_mini_viii:
3712 }

```

(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.

```

3713 \cs_new_protected:Nn \__enumext_after_list_viii:
3714 {
3715     \__enumext_stop_mini_viii:
3716     \__enumext_after_stop_list_viii:
3717     \__enumext_vspace_below_viii:
3718 }

```

(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`.

```
3719 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3720 {
3721   \__enumext_stop_item_tmp_viii:
3722   \int_incr:N \l__enumext_item_column_pos_viii_int
3723   \int_gincr:N \g__enumext_item_count_all_viii_int
3724   \__enumext_item_peek_args_viii:
3725 }
```

(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).

```
3726 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3727 {
3728   \peek_meaning:NTF (
3729     { \__enumext_joined_item_viii:w }
3730     { \__enumext_joined_item_viii:w (1) }
3731   }
```

(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`.

```
3732 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3733 {
3734   \__enumext_starred_joined_item_viii:n {#1}
3735   \peek_meaning_remove:NTF *
3736     { \__enumext_starred_item_viii:w }
3737     { \__enumext_standard_item_viii:w }
3738 }
```

(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]`.

```
3739 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3740 {
3741   \bool_set_false:N \l__enumext_item_starred_viii_bool
3742   \peek_meaning:NTF [
3743     {
3744       \bool_set_eq:NN
3745         \l__enumext_wrap_label_viii_bool
3746         \l__enumext_wrap_label_opt_viii_bool
3747       \__enumext_start_item_viii:w
3748     }
3749     {
3750       \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

3751         \legacy_if_set_true:n { @noitemarg }
3752         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3753     }
3754 }

```

(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].

```

3755 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3756 {
3757     \bool_set_true:N \l__enumext_item_starred_viii_bool
3758     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3759     \peek_meaning:NTF [
3760         { \__enumext_starred_item_viii_aux_i:w }
3761         { \__enumext_starred_item_viii_aux_ii:w }
3762     }

```

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

```

3763 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3764 {
3765     \tl_clear:N \l__enumext_store_keyans_label_tl
3766     \tl_if_no_value:nF { #1 }
3767     {
3768         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3769         {
3770             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3771             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3772         }
3773         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3774     }
3775     \__enumext_starred_item_viii_aux_ii:w
3776 }
3777 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3778 {
3779     \legacy_if_set_true:n { @noitemarg }
3780     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3781 }

```

(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.

```

3782 \cs_new_protected:Nn \__enumext_starred_item_exec:
3783 {
3784     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3785     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3786     \__enumext_keyans_store_ref:
3787     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3788     \__enumext_keyans_addto_seq_link:
3789     \int_gincr:N \g__enumext_check_starred_cmd_int
3790     \bool_if:NT \l__enumext_show_answer_bool
3791     {
3792         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3793     }
3794     \bool_if:NT \l__enumext_show_position_bool
3795     {
3796         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3797         {
3798             \group_begin:
3799             \exp_not:N \normalfont
3800             \exp_not:N \footnotesize [ \int_eval:n
3801                 {
3802                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3803                 }

```

```

3804         ]
3805     \group_end:
3806 }
3807 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
3808 }
3809 }

```

(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.

```

3810 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3811 {
3812     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3813     \legacy_if:nT { @noitemarg }
3814     {
3815         \legacy_if_set_false:n { @noitemarg }
3816         \legacy_if:nT { @nmbrlist }
3817         {
3818             \bool_if:NT \l__enumext_hyperref_bool
3819             {
3820                 \legacy_if_set_true:n { @hyper@item }
3821             }
3822             \refstepcounter{enumXviii}
3823         }
3824     }

```

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

```

3825     \group_begin:
3826     \lrbox{ \l__enumext_item_text_viii_box }
3827     \bool_if:NF \l__enumext_footnotes_key_bool
3828     {
3829         \__enumext_renew_footnote:
3830     }
3831     \bool_if:NT \l__enumext_item_starred_viii_bool
3832     {
3833         \__enumext_starred_item_exec:
3834     }
3835     \group_begin:
3836     \tl_use:N \l__enumext_label_font_style_viii_tl
3837     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3838     {
3839         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3840         { \__enumext_wrapper_label_viii:n {#1} }
3841     }
3842     {
3843         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3844     }
3845     \group_end:
3846     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3847     \tl_use:N \l__enumext_after_list_args_viii_tl
3848     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3849     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3850     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3851     \bool_if:NT \l__enumext_item_starred_viii_bool
3852     {
3853         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3854         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent:
3855     }
3856     {
3857         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3858     }
3859 }

```

(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.

```

3860 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3861 {

```

```

3862     \__enumext_endminipage:
3863     \endlrbox
3864     \group_end:
3865     \box_set_wd:Nn \l__enumext_item_text_viii_box
3866     {
3867         \l__enumext_joined_width_viii_dim
3868         + \l__enumext_labelwidth_viii_dim
3869         + \l__enumext_labelsep_viii_dim
3870     }
3871     \int_set:Nn \hbadness { 10000 }
3872     \box_use:N \l__enumext_item_text_viii_box
3873     \bool_if:NF \l__enumext_footnotes_key_bool
3874     {
3875         \__enumext_print_footnote:
3876     }
3877     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3878     {
3879         \par\noindent
3880         \int_zero:N \l__enumext_item_column_pos_viii_int
3881     }
3882     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3883 }

```

(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.

```

3884 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3885 {
3886     \int_compare:nNnT
3887     {
3888         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3889     }
3890     =
3891     { \c_zero_int }
3892     {
3893         \par
3894         \vspace{ -\l__enumext_itemsep_viii_skip }
3895         \int_gzero:N \g__enumext_item_count_all_viii_int
3896     }
3897 }

```

(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.

```

3898 \NewDocumentCommand \getkeyans { m }
3899 {
3900     \exp_args:Ne \__enumext_getkeyans_aux:n
3901     { \tl_to_str:e { \text_expand:n {#1} } }
3902 }

```

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

__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.

```

3903 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3904 {
3905     \str_if_in:nnTF {#1} { : }
3906     {
3907         \use:e
3908         {
3909             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3910             { {##1} {##2} }
3911         }
3912         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3913     }
3914     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3915 }

```


(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>*.

```

3916 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3917 {
3918   \prop_if_exist:cF { g__enumext_#1_prop }
3919   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3920   \group_begin:
3921   \prop_item:cn { g__enumext_#1_prop }{#2}
3922   \group_end:
3923 }

```

(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.

```

3924 \keys_define:nn { keyanskey / print }
3925 {
3926   level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3927                       {
3928                         \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3929                       },
3930   level-1 .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3931   level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3932                       {
3933                         \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3934                       },
3935   level-2 .initial:n   = { nosep, label=(\alph*), first=\small, font=\small },
3936   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3937                       {
3938                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3939                       },
3940   level-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
3941   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3942                       {
3943                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3944                       },
3945   level-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
3946   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3947                       {
3948                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3949                       },
3950   level-* .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3951 }

```

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

```

3952 \NewDocumentCommand \printkeyans { s O{} m }
3953 {
3954   \group_begin:
3955   \tl_use:N \l__enumext_print_keyans_i_tl
3956   \tl_use:N \l__enumext_print_keyans_ii_tl
3957   \tl_use:N \l__enumext_print_keyans_iii_tl
3958   \tl_use:N \l__enumext_print_keyans_iv_tl
3959   \tl_use:N \l__enumext_print_keyans_vii_tl
3960   \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3961   \group_end:
3962 }

```

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

__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

```

3963 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3964 {
3965   \seq_if_exist:cTF { g__enumext_#3_seq }
3966   {
3967     \seq_if_empty:cF { g__enumext_#3_seq }
3968     {
3969       %%\seq_show:c { g__enumext_#3_seq }
3970       \bool_if:nTF {#1}
3971       {
3972         \begin{enumext*}[#2]
3973         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3974         \end{enumext*}
3975       }
3976       {
3977         \begin{enumext}[#2]
3978         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3979         \end{enumext}
3980       }
3981     }
3982   }
3983   {
3984     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3985   }
3986 }

```

(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.

```

3987 \keys_define:nn { enumext / meta-families }
3988 {
3989   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3990   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3991   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3992   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3993   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3994   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3995   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3996   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3997   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3998   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3999   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
4000   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
4001   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4002 }

```

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

```

4003 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4004 {
4005   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
4006   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
4007 }

```

\setenumext Now we define the user command \setenumext.

```

4008 \NewDocumentCommand \setenumext { o +m }
4009 {
4010   \tl_if_novalue:nTF {#1}
4011   {
4012     \seq_map_inline:Nn \c__enumext_all_families_seq
4013     {
4014       \seq_clear:N \l__enumext_setkey_tmpa_seq
4015       \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4016       \int_set:Nn \l__enumext_setkey_tmpa_int
4017       {
4018         \seq_count:N \l__enumext_setkey_tmpb_seq
4019       }
4020       \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4021       {

```

```

4023         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4024         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4025         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4026         {
4027             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4028         }
4029     }
4030     {
4031         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4032     }
4033     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4034     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4035     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4036 }
4037 {
4038     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4039 }
4040 }

```

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

`__enumext_set_parse:n`
`__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```

4041 \cs_new_protected:Npn \__enumext_set_parse:n #1
4042 {
4043     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4044     \int_step_inline:nnn { 0 } { 4 } %<- max level
4045     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4046     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4047     {
4048         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4049         { \tl_trim_spaces:n {#1} }
4050     }
4051     { \__enumext_set_error:nn {#1} { } }
4052 }
4053 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4054 { \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.

```

4055 \msg_new:nnn { enumext } { package-load }
4056 {
4057     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4058 }
4059 \msg_new:nnn { enumext } { package-not-load }
4060 {
4061     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4062 }
4063 \msg_new:nnn { enumext } { package-load-foot }
4064 {
4065     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4066 }

```

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

```

4067 \msg_new:nnn { enumext } { counters }
4068 {
4069     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4070     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4071 }

```

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

```

4072 \msg_new:nnn { enumext } { invalid-key }
4073 {
4074     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4075 }
4076 \msg_new:nnn { enumext } { unknown-key-family }
4077 {
4078     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4079 }

```

Messages used in length calculation.

```
4080 \msg_new:nnn { enumext } { width-negative }
4081 {
4082   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4083   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4084 }
4085 \msg_new:nnn { enumext } { width-zero }
4086 {
4087   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4088   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4089 }
```

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

```
4090 \msg_new:nnn { enumext } { list-lengths }
4091 {
4092   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4093   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4094   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4095   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4096   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4097   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4098   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4099   \__enumext_show_length:nnn { skip } { topsep } {#1}
4100   \__enumext_show_length:nnn { skip } { parsep } {#1}
4101   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4102   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4103   ****
4104 }
```

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

```
4105 \msg_new:nnn { enumext } { list-lengths-not-nested }
4106 {
4107   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4108   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4109   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4110   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4111   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4112   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4113   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4114   \__enumext_show_length:nnn { skip } { topsep } {#1}
4115   \__enumext_show_length:nnn { skip } { parsep } {#1}
4116   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4117   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4118   ****
4119 }
```

Messages used by `ref` key.

```
4120 \msg_new:nnn { enumext } { key-ref-empty }
4121 {
4122   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4123 }
```

Messages used by `save-ans` key.

```
4124 \msg_new:nnn { enumext } { save-ans-empty }
4125 {
4126   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4127 }
4128 \msg_new:nnn { enumext } { save-ans-ok }
4129 {
4130   Set ~ 'save-ans=#2' ~ in ~ '#1' ~ \msg_line_context:.
4131 }
```

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

```
4132 \msg_new:nnn { enumext } { need-save-ans }
4133 {
4134   Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4135 }
4136 \msg_new:nnn { enumext } { items-same-answer }
4137 {
4138   \
4139   ****~Checking~answers~on~'#1'~OK~****\
4140   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \
4141   ****
```

```

4142     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4143     \\
4144   }
4145   \msg_new:nnn { enumext } { item-different-answer }
4146   {
4147     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4148     answer ~ stored ~ in ~ '#1' ~ #2.
4149   }

```

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

```

4150   \msg_new:nnn { enumext } { missing-starred }
4151   {
4152     Missing ~ '\c_backslash_str #1*' ~ #2.
4153   }
4154   \msg_new:nnn { enumext } { many-starred }
4155   {
4156     Many ~ '\c_backslash_str #1*' ~ #2.
4157   }

```

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

```

4158   \msg_new:nnn { enumext } { list-too-deep }
4159   {
4160     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4161     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4162   }

```

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

```

4163   \msg_new:nnn { enumext } { anskey-wrong-place }
4164   {
4165     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4166     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4167   }
4168   \msg_new:nnn { enumext } { anspic-wrong-place }
4169   {
4170     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4171     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4172   }
4173   \msg_new:nnn { enumext } { command-wrong-place }
4174   {
4175     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4176     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4177   }

```

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

```

4178   \msg_new:nnn { enumext } { keyans-nested }
4179   {
4180     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4181   }
4182   \msg_new:nnn { enumext } { keyans-wrong-level }
4183   {
4184     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4185     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4186   }
4187   \msg_new:nnn { enumext } { wrong-place }
4188   {
4189     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4190     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
4191   }
4192   \msg_new:nnn { enumext } { keyanspic-nested }
4193   {
4194     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4195   }
4196   \msg_new:nnn { enumext } { keyanspic-wrong-level }
4197   {
4198     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4199     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4200   }

```

Messages used by `\getkeyans` command.

```

4201   \msg_new:nnn { enumext } { undefined-storage-anskey }
4202   {
4203     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4204   }

```

Messages used by `\miniright` command.

```

4205 \msg_new:nnn { enumext } { missing-miniright }
4206 {
4207     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
4208     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4209 }
4210 \msg_new:nnn { enumext } { wrong-miniright-place }
4211 {
4212     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
4213     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4214 }
4215 \msg_new:nnn { enumext } { wrong-miniright-use }
4216 {
4217     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
4218     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4219 }

```

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

```

4220 \msg_new:nnn { enumext } { nested }
4221 {
4222     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4223 }
4224 \msg_new:nnn { enumext } { item-joined }
4225 {
4226     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4227 }
4228 \msg_new:nnn { enumext } { item-joined-columns }
4229 {
4230     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
4231 }

```

10.41 Finish package

Finish package implementation.

```

4232 \file_input_stop:
4233 </package>

```

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.

\backslash^*	199
\backslash^+	191
\backslash^-	191
$\backslash\backslash$	207, 3062, 4069, 4082, 4087, 4092, 4107, 4138, 4139, 4140, 4143, 4160, 4165, 4170, 4175, 4184, 4189, 4198, 4207, 4212, 4217	

above	1322
above*	1322
\addvspace	969, 997, 1120, 1199, 1262, 1268, 1296, 1313, 2901, 2916, 3018, 3033, 3259, 3266, 3649, 3656
after	807
align	424
\Alph	31, 36
\Alph	376, 485, 530, 598, 3945
\alph	31, 36
\alph	377, 483, 3935
\anskey	11, 65, 2039
\anspic	13, 86, 3040
\anspic*	61
\arabic	27, 31
\arabic	375, 482, 529, 3930, 3950

```

\b ..... 2779, 2792, 3325, 3338
\baselineskip ..... 44
\baselineskip ..... 1999, 2007
before ..... 807
before* ..... 807
below ..... 1322
below* ..... 1322
bool commands:
  \bool_gset_false:N .. 1842, 1843, 3268, 3272, 3658
  \bool_gset_true:N   219, 228, 911, 3251, 3269, 3641,
    3659
  \bool_if:NTF . 316, 328, 345, 1344, 1358, 1371, 1382,
    1393, 1404, 1415, 1426, 1479, 1496, 1501, 1509, 1536,
    1574, 1579, 1586, 1590, 1612, 1617, 1625, 1632, 1663,
    1671, 1694, 1698, 1704, 1710, 1756, 1760, 1764, 1801,
    1961, 1985, 1992, 2020, 2051, 2064, 2066, 2077, 2097,
    2222, 2233, 2237, 2276, 2291, 2366, 2377, 2381, 2494,
    2524, 2598, 2614, 2676, 2686, 2716, 2721, 2769, 2777,
    2790, 2835, 2885, 2899, 2907, 2946, 3003, 3016, 3024,
    3042, 3247, 3256, 3260, 3315, 3323, 3336, 3368, 3378,
    3461, 3466, 3474, 3478, 3493, 3522, 3637, 3646, 3650,
    3790, 3794, 3818, 3827, 3831, 3837, 3851, 3873
  \bool_if:nTF 1297, 1314, 2105, 2535, 2570, 2634, 3063,
    3970
  \bool_if_p:N . 237, 250, 1643, 1644, 1652, 1653, 1781,
    1811, 1812, 1817, 1818, 2088, 2131, 2132, 2156, 2165,
    2166, 2178, 2194, 2353, 2354, 2391, 2392, 2808, 2821,
    2823, 3070, 3071
  \bool_lazy_all:nTF 235, 248, 1779, 2154, 2163, 2176,
    2192, 2806, 2819
  \bool_lazy_and:nnTF .. 215, 224, 1642, 1651, 1810,
    1816, 2087, 2130, 2352
  \bool_lazy_or:nnTF ..... 2390, 3069

```

```
\bool_new:N 25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88,
93, 94, 97, 115, 117, 119, 122, 123, 132, 133, 139, 140,
154, 165, 167
\bool_not_p:n 216, 225, 2089, 2181, 2196, 2809, 2810,
2822
\bool_set_eq:NN . . . . . 2502, 2550, 3411, 3744
\bool_set_false:N 325, 1743, 1744, 2922, 2954, 3036,
3101, 3119, 3363, 3408, 3694, 3741
\bool_set_true:N 242, 255, 307, 311, 417, 735, 1328,
1333, 1599, 1720, 1721, 1923, 1930, 2498, 2528, 2546,
2558, 2752, 2815, 2828, 2854, 2951, 2978, 3236, 3305,
3417, 3424, 3425, 3626, 3750, 3757, 3758
```

```

\box_dp:N . . . 1016, 1020, 1024, 1035, 1039, 1050, 1059,
              1065, 1075, 1088, 1094, 1100, 1131, 1132, 1133, 1136,
              1146, 1150, 1159, 1166, 1171, 1179, 1208, 1209, 1212,
              1219, 1232, 1240, 1246, 1254, 3131
\box_new:N . . . . . 58, 160
\box_set_wd:Nn . . . . . 3514, 3865
\box_use:N . . . . . 3521, 3872
\box_wd:N . . . . . 383

```

\c	199, 200, 635, 637, 649, 651
\cB	200
\cE	200
\centering	1299, 1316, 3152, 3262, 3652
check-ans	<u>1735</u>
Document class:	
article	37
clist commands:	
\clist_const:Nn	172
\clist_map_function:nN	3139
\clist_map_inline:Nn	423, 677, 740, 806, 821, 902, 1338
\clist_map_inline:nn	36, 47, 66, 72, 84, 96, 121, 148, 171, 448, 465, 745, 917, 1444, 1688, 1749, 1900, 1918, 1939, 2151, 2285, 2452, 2663, 2666, 2693, 2703, 2706, 2726
\columnbreak	66
\columnbreak	2091
columns	<u>886</u>
columns*	<u>1919</u>
columns-sep	<u>886</u>
columns-sep*	<u>1919</u>
\columnsep	82, 85
\columnsep	2879, 3000
\columnseprule	82, 85
\columnseprule	2883, 3002
Commands provide by enumext :	
\anskey	24, 25, 58, 59, 63, 65, 67–69, 71, 80, 93, 103, 104, 108
\anspic*	24, 25, 61, 69, 70, 86–88, 103, 104
\anspic	63, 86–88, 108
\getkeyans	63, 103, 108
\item* ..	24, 25, 61, 63, 69, 70, 74, 75, 94, 101, 103, 104
\itemwidth	89, 97
\item	73, 75, 89, 93–95, 97, 100
\miniright	24, 42, 49, 50, 81, 83–85, 109
\printkeyans	25, 63, 104

`\setenumext` 25, 105, 106
 Counters defined by `enumext`:
 `enumXiii` 23, 31
 `enumXii` 23, 31
 `enumXiv` 23, 31
 `enumXi` 23, 31
 `enumXviii` 23, 31
 `enumXvii` 23, 31, 95
 `enumXvi` 23, 31
 `enumXv` 23, 31
 cs commands:
 `\cs_generate_variant:Nn` 385, 401, 641, 657, 1944,
 1953, 1958, 2038, 2653, 3141
 `\cs_if_exist:NTF` 355
 `\cs_new:Nn` 185
 `\cs_new:Npn` 203, 1445, 1454, 1463
 `\cs_new_eq:NN` 291, 292, 293, 297, 298, 330, 331, 334,
 335
 `\cs_new_protected:Nn` . 195, 209, 233, 262, 302, 506,
 569, 621, 822, 826, 830, 834, 838, 842, 846, 850, 854,
 858, 862, 866, 870, 874, 878, 882, 918, 930, 954, 971,
 982, 1006, 1081, 1105, 1122, 1184, 1201, 1223, 1258,
 1264, 1339, 1353, 1367, 1378, 1389, 1400, 1411, 1422,
 1507, 1610, 1623, 1640, 1661, 1718, 1754, 1774, 1808,
 1823, 1959, 1983, 1990, 2018, 2025, 2142, 2274, 2289,
 2317, 2350, 2386, 2398, 2406, 2457, 2461, 2480, 2531,
 2566, 2582, 2592, 2608, 2746, 2804, 2833, 2840, 2863,
 2893, 2905, 2944, 2968, 2986, 3011, 3022, 3059, 3103,
 3117, 3137, 3142, 3158, 3226, 3245, 3297, 3350, 3357,
 3366, 3376, 3393, 3533, 3548, 3616, 3635, 3685, 3707,
 3713, 3726, 3782, 3884
 `\cs_new_protected:Npn` 177, 181, 338, 353, 370, 380,
 386, 486, 531, 603, 628, 642, 1286, 1305, 1475, 1494,
 1564, 1597, 1689, 1848, 1945, 1954, 2074, 2219, 2231,
 2253, 2327, 2371, 2490, 2508, 2542, 2554, 2622, 2656,
 2696, 2755, 2775, 2964, 3112, 3177, 3308, 3321, 3399,
 3406, 3422, 3430, 3435, 3447, 3567, 3700, 3732, 3739,
 3755, 3763, 3777, 3903, 3916, 3963, 4041, 4053
 `\cs_new_protected_nopar:Nn` . . . 3386, 3509, 3719,
 3860
 `\cs_new_protected_nopar:Npn` 3453, 3810
 `\cs_set:Nn` 2224
 `\cs_set:Npn` 2152, 2190, 3909
 `\cs_set_eq:NN` . . 3287, 3288, 3455, 3675, 3676, 3812
 `\cs_set_protected:Nn` 746, 762, 774, 786
 `\cs_set_protected:Npn` . 32, 41, 59, 67, 79, 85, 111,
 144, 152, 402, 424, 453, 466, 513, 658, 678, 722, 741,
 798, 807, 886, 903, 1322, 1433, 1680, 1735, 1865, 1901,
 1919, 2144, 2278, 2441, 2654, 2694
 `\cs_to_str:N` 372, 395

D

`\d` 191
`\DeclareDocumentEnvironment` 999
 dim commands:
 `\dim_abs:n` 2627, 2632
 `\dim_add:Nn` 3122
 `\dim_compare:nNnTF` . 748, 764, 776, 788, 1288, 1307,
 2624, 2629, 2635, 2641, 2643, 2645, 2845, 2868, 2972,
 2990, 3114, 3160, 3228, 3550, 3618
 `\dim_compare:nTF` 2115
 `\dim_gset_eq:NN` 3237, 3627
 `\dim_gzero:N` 3271, 3661
 `\dim_new:N` . 55, 62, 63, 64, 81, 118, 128, 161, 162, 168
 `\dim_set:Nn` . . 383, 736, 1931, 2522, 2627, 2632, 2634,
 2637, 2638, 2642, 2644, 2647, 2648, 2650, 2848, 2871,

 2974, 2992, 3144, 3162, 3169, 3212, 3230, 3449, 3552,
 3559, 3602, 3620
 `\dim_set_eq:NN` 473, 520, 591, 595, 2517, 2665, 2705,
 2794, 2879, 3000, 3219, 3222, 3223, 3340, 3440, 3609,
 3612, 3613
 `\dim_use:N` 749, 757, 1289, 1295, 2028, 2031, 2036, 2587,
 2589, 2846, 2851, 2852, 2859, 2869, 2873, 2874, 2876
 `\dim_zero:N` 2883, 3002, 3123, 3124, 3125
 `\dim_zero_new:N` 3175, 3565
 `\c_zero_dim` 751, 765, 777, 789, 1289, 1307, 2117, 2624,
 2629, 2635, 2642, 2846, 2869, 2972, 2990, 3160, 3228,
 3550, 3618

E

`\end` . . 1292, 1310, 1987, 2022, 2898, 2915, 3015, 3032, 3249,
 3265, 3639, 3655, 3974, 3979
`\endlist` 29
`\endlist` 292
`\endlrbox` 3512, 3863
`\endminipage` 29
`\endminipage` 298
`enumext` 5, 2727
 enumext internal commands:

`\l__enumext__check_start_line_env_tl` . . . 28
 `\l__enumext__ref_the_count_tl` 33
 `\l__enumext__resume_name_tl` 54
 `__enumext_add_pre_parsep:` . . . 43, 928, 930, 930
 `__enumext_after_args_exec:` . 40, 822, 834, 2739
 `__enumext_after_args_exec_v:` . 40, 41, 838, 850,
 2937
 `__enumext_after_args_exec_vii:` . . . 854, 878
 `__enumext_after_args_exec_viii:` 882
 `__enumext_after_env:nn` 60, 83, 96, 181, 181, 2925,
 3254, 3547, 3644
 `__enumext_after_hyperref:` . . . 29, 300, 302, 302
 `__enumext_after_list:` 83, 93, 99, 2744, 2905, 2905
 `\l__enumext_after_list_args_v_tl` 852
 `\l__enumext_after_list_args_vii_tl` 880, 3503
 `\l__enumext_after_list_args_viii_tl` 884, 3847
 `__enumext_after_list_v:` . . 85, 2942, 3022, 3022
 `__enumext_after_list_vii:` . . 3295, 3357, 3357
 `__enumext_after_list_viii:` . . 3683, 3713, 3713
 `__enumext_after_star_env:nn` 90
 `__enumext_after_stop_list:` . . 40, 41, 822, 830,
 2920
 `__enumext_after_stop_list_v:` 40, 838, 846, 3037
 `\l__enumext_after_stop_list_v_tl` 848
 `__enumext_after_stop_list_vii:` 854, 870, 3360
 `\l__enumext_after_stop_list_vii_tl` . . . 872
 `__enumext_after_stop_list_viii:` . 874, 3716
 `\l__enumext_after_stop_list_viii_tl` . . . 876
 `\l__enumext_align_label_vii_str` . . 3495, 3499
 `\l__enumext_align_label_viii_str` . 3839, 3843
 `\l__enumext_align_label_X_str` 152
 `\c__enumext_all_envs_clist` . . 172, 423, 677, 740,
 806, 821, 902, 1338
 `\c__enumext_all_families_seq` . . 105, 4003, 4012,
 4034
 `__enumext_anskey_wrapper:n` 1869, 2229
 `__enumext_at_begin_document:n` . . 29, 177, 177,
 289, 295
 `__enumext_before_args_exec:` 40, 822, 822, 2843
 `__enumext_before_args_exec_v:` 40, 41, 838, 838,
 2971

`__enumext_before_args_exec_vii:` .. [854](#), [854](#), [3354](#)
`__enumext_before_args_exec_viii:` [858](#), [3710](#)
`__enumext_before_keys_exec:` [40](#), [822](#), [826](#), [2737](#)
`__enumext_before_keys_exec_v:` .. [40](#), [838](#), [842](#), [2935](#)
`__enumext_before_keys_exec_vii` [854](#)
`__enumext_before_keys_exec_vii:` [41](#), [862](#), [3283](#)
`__enumext_before_keys_exec_viii:` .. [41](#), [866](#), [3671](#)
`__enumext_before_list:` ... [81](#), [2731](#), [2840](#), [2840](#)
`__enumext_before_list_v:` . [84](#), [2930](#), [2968](#), [2968](#)
`__enumext_before_list_vii:` [92](#), [3278](#), [3350](#), [3350](#)
`__enumext_before_list_viii:` .. [99](#), [3667](#), [3707](#), [3707](#)
`\l__enumext_before_no_starred_key_v_tl` [844](#)
`\l__enumext_before_no_starred_key_vii_-tl` [864](#)
`\l__enumext_before_no_starred_key_viii_-tl` [868](#)
`\l__enumext_before_starred_key_v_tl` ... [840](#)
`\l__enumext_before_starred_key_vii_tl` . [856](#)
`\l__enumext_before_starred_key_viii_tl` [860](#)
`__enumext_calc_hspace:NNNNNN` [77](#), [2622](#), [2622](#), [2653](#), [2658](#), [2698](#)
`__enumext_check_ans:` .. [59](#), [81](#), [1754](#), [1754](#), [2844](#), [3353](#)
`\l__enumext_check_ans_bool` [59](#), [73](#), [74](#), [132](#), [1739](#), [1744](#), [1756](#), [1811](#), [1817](#), [2066](#), [2366](#), [2494](#), [2524](#), [3466](#)
`\g__enumext_check_ans_item_tl` [71](#)
`__enumext_check_ans_level:` . [59](#), [60](#), [1754](#), [1770](#), [1774](#)
`__enumext_check_ans_show:` [60](#), [1823](#), [1823](#), [2925](#), [3547](#)
`\g__enumext_check_ans_show_bool` [83](#)
`__enumext_check_ans_to_hook:` . [60](#), [1808](#), [1808](#), [2919](#), [3361](#)
`__enumext_check_starred_cmd:n` [28](#), [61](#), [71](#), [1848](#), [1848](#), [2940](#), [3098](#), [3681](#)
`\g__enumext_check_starred_cmd_int` [132](#), [1851](#), [1857](#), [1862](#), [2564](#), [3068](#), [3789](#)
`\g__enumext_check_start_line_env_tl` . [28](#), [132](#), [243](#), [256](#), [1837](#), [1845](#)
`\l__enumext_check_start_line_env_tl` [132](#), [268](#), [275](#), [282](#), [1854](#), [1860](#), [1863](#)
`\l__enumext_columns_sep_v_dim` [2990](#), [2992](#), [3000](#)
`\l__enumext_columns_sep_vii_dim` .. [3160](#), [3162](#), [3171](#), [3216](#), [3342](#), [3531](#)
`\l__enumext_columns_sep_viii_dim` . [3550](#), [3552](#), [3561](#), [3606](#), [3882](#)
`\l__enumext_columns_v_int` [1127](#), [2988](#), [2996](#), [3008](#), [3013](#)
`\l__enumext_columns_vii_int` .. [3165](#), [3168](#), [3172](#), [3180](#), [3184](#), [3187](#), [3193](#), [3199](#), [3203](#), [3329](#), [3526](#), [3537](#)
`\l__enumext_columns_viii_int` . [3555](#), [3558](#), [3562](#), [3570](#), [3574](#), [3577](#), [3583](#), [3589](#), [3593](#), [3877](#), [3888](#)
`\g__enumext_count_item_anskey_int` .. [71](#), [132](#), [1830](#), [1841](#), [2068](#), [2368](#)
`\g__enumext_count_item_number_int` .. [60](#), [132](#), [1785](#), [1789](#), [1792](#), [1795](#), [1803](#), [1830](#), [1840](#), [2496](#), [2526](#), [3468](#)
`\g__enumext_count_item_with_ans_int` [65](#)
`\l__enumext_counter_i_tl` [32](#), [362](#)
`\l__enumext_counter_ii_tl` [32](#), [363](#)
`\l__enumext_counter_iii_tl` [32](#), [364](#)
`\l__enumext_counter_iv_tl` [32](#), [365](#)
`\c__enumext_counter_style_tl` [27](#), [37](#), [197](#)
`\g__enumext_counter_styles_tl` . [23](#), [31](#), [55](#), [373](#), [391](#)
`\l__enumext_counter_v_tl` [32](#), [366](#), [611](#)
`\l__enumext_counter_vii_tl` [32](#), [367](#)
`\l__enumext_counter_viii_tl` [32](#), [368](#), [541](#)
`\l__enumext_counter_viii_tl` [32](#), [369](#), [558](#)
`\l__enumext_current_widest_dim` [23](#), [55](#), [397](#), [474](#), [521](#), [592](#), [596](#)
`__enumext_default_item:n` ... [2490](#), [2490](#), [2539](#)
`__enumext_define_counters:Nn` [23](#), [353](#), [353](#), [362](#), [363](#), [364](#), [365](#), [366](#), [367](#), [368](#), [369](#)
`__enumext_endminipage:` [29](#), [295](#), [298](#), [1005](#), [3154](#), [3511](#), [3862](#)
`__enumext_fake_item:` [746](#), [746](#), [2685](#)
`\l__enumext_fake_item_indent_v_dim` [765](#), [770](#)
`\l__enumext_fake_item_indent_v_tl` [767](#), [2547](#), [2551](#), [2559](#)
`\l__enumext_fake_item_indent_vii_dim` [777](#), [782](#)
`\l__enumext_fake_item_indent_vii_tl` [779](#), [3507](#)
`\l__enumext_fake_item_indent_viii_dim` . [789](#), [794](#), [3854](#)
`\l__enumext_fake_item_indent_viii_tl` .. [791](#), [3853](#), [3857](#)
`\l__enumext_fake_item_indent_X_tl` [85](#)
`__enumext_fake_item_vii:` [746](#), [774](#), [2715](#)
`__enumext_fake_item_viii:` [746](#), [786](#), [2720](#)
`__enumext_filter_series:n` [53](#), [1445](#), [1445](#), [1487](#), [1499](#), [1504](#)
`__enumext_filter_series_key:n` [53](#), [1445](#), [1450](#), [1454](#)
`__enumext_filter_series_pair:nn` .. [53](#), [1445](#), [1451](#), [1463](#)
`\g__enumext_footnote_arg_seq` . [149](#), [2463](#), [2476](#), [2486](#)
`\g__enumext_footnote_int` . [149](#), [2470](#), [2473](#), [2475](#), [2477](#)
`\g__enumext_footnote_int_seq` . [149](#), [2464](#), [2477](#), [2482](#), [2485](#)
`__enumext_footnotes_key_bool` [29](#)
`\l__enumext_footnotes_key_bool` [25](#), [30](#), [95](#), [139](#), [311](#), [316](#), [325](#), [3474](#), [3522](#), [3827](#), [3873](#)
`__enumext_footnotetext:nn` ... [2457](#), [2457](#), [2487](#)
`__enumext_getkeyans:nn` .. [104](#), [3912](#), [3916](#), [3916](#)
`__enumext_getkeyans_aux:n` [103](#), [3900](#), [3903](#), [3903](#)
`\l__enumext_hyperref_bool` . [25](#), [29](#), [30](#), [139](#), [307](#), [328](#), [345](#), [2132](#), [2354](#), [3461](#), [3818](#)
`__enumext_hypertarget:nn` [30](#), [302](#), [330](#), [334](#), [350](#)
`__enumext_if_is_int:n` [189](#)
`__enumext_if_is_int:nTF` [189](#), [630](#), [644](#)
`__enumext_is_not_nested:` [27](#), [79](#), [209](#), [209](#), [2748](#), [3299](#)
`__enumext_is_on_first_level:` . [28](#), [79](#), [209](#), [233](#), [2753](#), [3306](#)
`\l__enumext_item_column_pos_vii_int` [93](#), [3187](#), [3193](#), [3199](#), [3203](#), [3210](#), [3389](#), [3526](#), [3529](#)
`\l__enumext_item_column_pos_viii_int` .. [100](#), [3577](#), [3583](#), [3589](#), [3593](#), [3600](#), [3722](#), [3877](#), [3880](#)
`\l__enumext_item_column_pos_X_int` [152](#)
`\g__enumext_item_count_all_vii_int` [93](#), [3211](#), [3390](#), [3537](#), [3544](#)
`\g__enumext_item_count_all_viii_int` [100](#), [3601](#), [3723](#), [3888](#), [3895](#)

```

\g__enumext_item_count_all_X_int . . . . . 152
\__enumext_item_peek_args_vii: 93, 3391, 3393,
    3393
\__enumext_item_peek_args_viii: . . 100, 3724,
    3726, 3726
\__enumext_item_starred: . . 76, 2582, 2582, 2600
\l__enumext_item_starred_vii_bool 3408, 3424,
    3478
\l__enumext_item_starred_viii_bool 3741, 3757,
    3831, 3851
\l__enumext_item_starred_X_bool . . . . . 152
\__enumext_item_std:w 29, 73–75, 88, 289, 293, 2499,
    2505, 2529, 2547, 2551, 2559, 3135
\g__enumext_item_symbol_aux_vii_tl 3432, 3480,
    3483, 3487, 3489
\g__enumext_item_symbol_aux_X_tl . . . . . 152
\l__enumext_item_symbol_sep_vii_dim . . 3441,
    3449, 3486, 3488
\g__enumext_item_symbol_tl 23, 74, 48, 2514, 2588,
    2605
\l__enumext_item_symbol_vii_tl . . . . . 3483
\l__enumext_item_text_vii_box 3473, 3514, 3521
\l__enumext_item_text_viii_box 3826, 3865, 3872
\l__enumext_item_text_X_box . . . . . 152
\l__enumext_item_width_vii_dim . . 3169, 3214,
    3222, 3223
\l__enumext_item_width_viii_dim . . 3559, 3604,
    3612, 3613
\l__enumext_item_width_X_dim . . . . . 152
\l__enumext_itemindent_X_dim . . . . . 59
\l__enumext_itemsep_vii_skip . . . . . 3543
\l__enumext_itemsep_viii_skip . . . . . 3894
\l__enumext_joined_item_aux_vii_int . . 3208,
    3209, 3210, 3211, 3217
\l__enumext_joined_item_aux_viii_int . 3598,
    3599, 3600, 3601, 3607
\l__enumext_joined_item_aux_X_int . . . 152
\__enumext_joined_item_vii:w 93, 94, 3396, 3397,
    3399, 3399
\l__enumext_joined_item_vii_int . . 3179, 3180,
    3183, 3185, 3191, 3196, 3201, 3206, 3208, 3214
\__enumext_joined_item_viii:w . 100, 3729, 3730,
    3732, 3732
\l__enumext_joined_item_viii_int . 3569, 3570,
    3573, 3575, 3581, 3586, 3591, 3596, 3598, 3604
\l__enumext_joined_item_X_int . . . . . 152
\l__enumext_joined_width_vii_dim . 3212, 3219,
    3222, 3504, 3516
\l__enumext_joined_width_viii_dim 3602, 3609,
    3612, 3848, 3867
\l__enumext_joined_width_X_dim . . . . . 152
\__enumext_keyans_addto_prop:n 69, 2253, 2253,
    2561, 3065
\__enumext_keyans_addto_seq:n . 70, 2327, 2327,
    2563, 3067
\__enumext_keyans_addto_seq_link: 2327, 2348,
    2350, 3788
\__enumext_keyans_anspic_code:nnn 86, 87, 3056,
    3059, 3059
\__enumext_keyans_default_item:n . . 75, 2542,
    2542, 2578
\l__enumext_keyans_env_bool 20, 2809, 2822, 2951,
    3036
\__enumext_keyans_fake_item: . . 746, 762, 2675
\l__enumext_keyans_item_opt_tl . 101, 97, 2375,
    2388, 2394, 3773
\l__enumext_keyans_level_h_int . . 20, 551, 578,
    2305, 3687, 3688
\l__enumext_keyans_level_int . . 20, 1280, 2055,
    2300, 2950, 2955, 3050
\__enumext_keyans_make_label: 32, 76, 2608, 2608,
    2673
\__enumext_keyans_mini_addvspace: 47, 84, 1184,
    1184, 2980
\__enumext_keyans_mini_right_cmd:n 50, 1282,
    1305, 1305
\__enumext_keyans_mini_set_vskip: . 47, 1122,
    1122, 1186
\__enumext_keyans_multi_addvspace: . 85, 971,
    982, 3005
\__enumext_keyans_multi_set_vskip: . 43, 971,
    971, 984
\__enumext_keyans_multicols_start: . . 84, 85,
    2984, 2986, 2986
\__enumext_keyans_multicols_stop: . 85, 1309,
    3011, 3011, 3035
\__enumext_keyans_parse_keys:n 2929, 2964, 2964
\l__enumext_keyans_pic_above_int . 127, 3145,
    3146, 3148
\l__enumext_keyans_pic_above_skip . . 88, 127,
    3089, 3129
\__enumext_keyans_pic_arg_two: 88, 3087, 3117,
    3117
\l__enumext_keyans_pic_below_int . 127, 3145,
    3146, 3149
\l__enumext_keyans_pic_body_seq . . 86–88, 127,
    3054, 3094, 3153
\__enumext_keyans_pic_do:n 88, 3094, 3096, 3137,
    3137, 3141
\l__enumext_keyans_pic_level_int . . 20, 1272,
    2059, 2256, 2295, 2330, 2408, 3105, 3106
\__enumext_keyans_pic_row:n 88, 3139, 3142, 3142
\__enumext_keyans_pic_safe_exec: . . 87, 3083,
    3103, 3103
\__enumext_keyans_pic_skip_abs:N . . 88, 3112,
    3112, 3128
\l__enumext_keyans_pic_width_dim . 127, 3144,
    3151
\__enumext_keyans_redefine_item: . . 75, 2566,
    2566, 2672
\__enumext_keyans_ref: . . . . . 36, 603, 621, 2674
\__enumext_keyans_ref:n . . . . . 35, 600, 603, 603
\__enumext_keyans_safe_exec: . 2928, 2944, 2944
\__enumext_keyans_save_start_line: . 28, 262,
    262, 2952, 3110, 3692
\__enumext_keyans_show_ans: . . 2371, 2379, 2398
\__enumext_keyans_show_item_opt: . 2371, 2386,
    2559, 3079, 3854
\__enumext_keyans_show_left:n . 75, 2371, 2371,
    2557, 3074
\__enumext_keyans_show_pos: . . 2371, 2383, 2406
\__enumext_keyans_starred_item:n . . 75, 2554,
    2554, 2574
\__enumext_keyans_store_ref: . . 69, 2274, 2274,
    2562, 3066, 3786
\__enumext_keyans_store_ref_aux_i: 70, 2274,
    2286, 2289
\__enumext_keyans_store_ref_aux_ii: 70, 2274,

```

2315, 2317
 \l__enumext_keyans_tmpa_tl . . 24, 97, 2556, 2560
 __enumext_keyans_wrapper_opt:n . . 1872, 2394
 \l__enumext_label_copy_i_tl . . 2186, 2293, 2298,
 2303, 2308
 \l__enumext_label_copy_v_tl 2303
 \l__enumext_label_copy_vi_tl 2298
 \l__enumext_label_copy_vii_tl 2161, 2172, 2203,
 2293
 \l__enumext_label_copy_viii_tl 2308
 \l__enumext_label_copy_X_tl 141
 \l__enumext_label_fill_left_v_tl 2612
 \l__enumext_label_fill_left_X_tl 85
 \l__enumext_label_fill_right_v_tl 2619
 \l__enumext_label_fill_right_X_tl 85
 \l__enumext_label_font_style_v_tl 2613, 3078
 \l__enumext_label_font_style_vii_tl . . 3492
 \l__enumext_label_font_style_viii_tl . . 3836
 \l__enumext_label_i_tl 466
 \l__enumext_label_ii_tl 466
 \l__enumext_label_iii_tl 466
 \l__enumext_label_iv_tl 466
 __enumext_label_style:Nnn 23, 31, 386, 386, 401,
 471, 518, 589, 593
 \l__enumext_label_v_tl . . 69, 70, 586, 2261, 2335,
 2400, 2435, 2556, 2560, 2932, 3073, 3075
 \l__enumext_label_vi_tl . 69, 70, 586, 2258, 2332,
 3073, 3075, 3079
 \l__enumext_label_vii_tl . 513, 3419, 3444, 3451
 \l__enumext_label_viii_tl 513, 3752, 3780, 3784
 \l__enumext_label_width_by_box . . 55, 382, 383
 __enumext_label_width_by_box:Nn 31, 380, 380,
 385, 397, 654
 \l__enumext_labelsep_i_dim . . 2403, 2438, 3792,
 3807
 \l__enumext_labelsep_v_dim 2995
 \l__enumext_labelsep_vii_dim . 3164, 3173, 3215,
 3442, 3502, 3518
 \l__enumext_labelsep_viii_dim 3554, 3563, 3605,
 3846, 3869
 \l__enumext_labelwidth_i_dim . 2403, 2438, 3792,
 3807
 \l__enumext_labelwidth_v_dim 2995
 \l__enumext_labelwidth_vii_dim . . . 3164, 3172,
 3215, 3495, 3499, 3517
 \l__enumext_labelwidth_viii_dim . . 3554, 3562,
 3605, 3839, 3843, 3868
 \l__enumext_leftmargin_tmp_v_bool . 88, 3119
 \l__enumext_leftmargin_tmp_X_bool 59
 \l__enumext_leftmargin_tmp_X_dim 59
 \l__enumext_leftmargin_X_dim 59
 __enumext_level: 185, 185, 495, 498, 499, 508, 510,
 749, 753, 757, 824, 828, 832, 836, 920, 922, 924, 926,
 959, 961, 963, 965, 969, 1009, 1012, 1031, 1040, 1046,
 1051, 1055, 1066, 1070, 1071, 1076, 1112, 1116, 1289,
 1295, 1342, 1344, 1346, 1349, 1356, 1358, 1360, 1363,
 1963, 1971, 1975, 1979, 2224, 2227, 2228, 2498, 2499,
 2503, 2504, 2505, 2512, 2514, 2518, 2519, 2522, 2528,
 2529, 2584, 2587, 2589, 2596, 2597, 2598, 2601, 2604,
 2734, 2736, 2777, 2782, 2783, 2784, 2786, 2790, 2795,
 2796, 2797, 2799, 2815, 2828, 2835, 2846, 2848, 2851,
 2852, 2854, 2859, 2866, 2869, 2871, 2873, 2874, 2875,
 2876, 2879, 2885, 2890, 2896, 2899, 2901, 2907
 \l__enumext_level_h_int . . 20, 217, 239, 251, 534,
 571, 1782, 1798, 2180, 2197, 3300, 3301
 \l__enumext_level_int . 79, 20, 187, 226, 238, 252,
 932, 1083, 1276, 1776, 1825, 2157, 2167, 2173, 2179,
 2187, 2195, 2202, 2688, 2749, 2750, 2760, 2767, 2813,
 2826, 2881, 2959, 3046, 3370, 3380, 3695
 __enumext_list_arg_two_i: 2654
 __enumext_list_arg_two_ii: 2654
 __enumext_list_arg_two_iii: 2654
 __enumext_list_arg_two_iv: 2654
 __enumext_list_arg_two_v: . 75, 2654, 2934, 3120
 __enumext_list_arg_two_vii: 2694, 3282
 __enumext_list_arg_two_viii: 2694, 3670
 \l__enumext_listoffset_v_dim 2997
 \l__enumext_listparindent_vii_dim 3505
 \l__enumext_listparindent_viii_dim . . . 3849
 __enumext_make_label: 32, 73, 74, 76, 2592, 2592,
 2683
 \l__enumext_mark_answer_sym_tl . 64, 122, 1878,
 2033, 2239, 2410, 2423, 3796
 \l__enumext_mark_position_str 122, 1882, 1883,
 1906, 1907, 2031
 \l__enumext_mark_ref_sym_tl . . 122, 1892, 2137,
 2362
 __enumext_mini_addvspace: . . 46, 81, 1105, 1105,
 2856
 __enumext_mini_addvspace_vii: 49, 1258, 1258,
 3240
 __enumext_mini_addvspace_viii: 49, 1258, 1264,
 3630
 __enumext_mini_env* 999
 __enumext_mini_right_cmd:n . 49, 50, 1284, 1286,
 1286
 __enumext_mini_set_vskip: . 44, 1006, 1006, 1107
 __enumext_mini_set_vskip_vii: 48, 1201, 1201,
 1260
 __enumext_mini_set_vskip_viii: 48, 1201, 1223,
 1266
 __enumext_minipage:w . . 29, 295, 297, 1001, 3151,
 3504, 3848
 \l__enumext_minipage_active_v_bool . . 84, 85,
 2978, 3003, 3016, 3024
 \g__enumext_minipage_active_vii_bool . . . 90,
 3251, 3256, 3268
 \l__enumext_minipage_active_vii_bool . 3236,
 3247
 \g__enumext_minipage_active_viii_bool 3641,
 3646, 3658
 \l__enumext_minipage_active_viii_bool 3626,
 3637
 \g__enumext_minipage_active_X_bool . . . 152
 \l__enumext_minipage_active_X_bool 73
 \g__enumext_minipage_after_skip 73, 1205, 1217,
 3266, 3656
 \l__enumext_minipage_after_skip 44, 46, 83, 85,
 73, 1022, 1037, 1057, 1073, 1088, 1094, 1100, 1114,
 1124, 1133, 1136, 1148, 1166, 1177, 1193, 1225, 1238,
 1252, 2916, 3033
 \g__enumext_minipage_center_vii_bool . 3260,
 3269
 \g__enumext_minipage_center_viii_bool 3650,
 3659
 \g__enumext_minipage_center_X_bool . . . 152
 \l__enumext_minipage_hsep_v_dim . . . 84, 2976
 \l__enumext_minipage_hsep_vii_dim . . . 3234

```

\l__enumext_minipage_hsep_viii_dim ... 3624
\l__enumext_minipage_left_skip 44, 84, 73, 1014,
    1029, 1048, 1063, 1110, 1120, 1125, 1131, 1140, 1157,
    1169, 1189, 1199, 1203, 1208, 1212, 1226, 1230, 1244,
    1262, 1268
\l__enumext_minipage_left_v_dim 84, 2974, 2982
\l__enumext_minipage_left_vii_dim 3230, 3242
\l__enumext_minipage_left_viii_dim 3620, 3632
\l__enumext_minipage_left_X_dim ..... 73
\g__enumext_minipage_right_skip 73, 1204, 1209,
    1213, 3259, 3649
\l__enumext_minipage_right_skip . 44, 73, 1018,
    1033, 1053, 1068, 1126, 1132, 1144, 1162, 1173, 1227,
    1234, 1248, 1296, 1313
\l__enumext_minipage_right_v_dim .. 84, 1307,
    1312, 2972, 2976
\g__enumext_minipage_right_vii_dim 90, 3238,
    3258, 3271
\l__enumext_minipage_right_vii_dim 90, 3228,
    3233, 3239
\g__enumext_minipage_right_viii_dim .. 3628,
    3648, 3661
\l__enumext_minipage_right_viii_dim .. 3618,
    3623, 3629
\g__enumext_minipage_right_X_dim ..... 152
\g__enumext_minipage_right_X_skip .... 152
\g__enumext_minipage_stat_int . 81, 84, 73, 1301,
    1318, 2855, 2909, 2914, 2979, 3026, 3031
\g__enumext_miniright_code_vii_tl . 91, 3264,
    3270
\g__enumext_miniright_code_viii_tl 3654, 3660
\g__enumext_miniright_code_X_tl ..... 152
\__enumext_multi_addvspace: ... 43, 82, 954, 954,
    2887
\__enumext_multi_set_vskip: .. 42, 918, 918, 956
\l__enumext_multicols_above_ii_skip ... 937
\l__enumext_multicols_above_iii_skip .. 943
\l__enumext_multicols_above_iv_skip ... 949
\l__enumext_multicols_above_v_skip 973, 987,
    997
\l__enumext_multicols_above_X_skip .... 67
\l__enumext_multicols_below_v_skip 977, 991,
    3018
\l__enumext_multicols_below_X_skip .... 67
\__enumext_multicols_start: 82, 2861, 2863, 2863
\__enumext_multicols_stop: 82, 1291, 2893, 2893,
    2918
\__enumext_newlabel:nn 26, 30, 68, 338, 338, 2213,
    2321
\l__enumext_newlabel_arg_one_tl 26, 30, 68, 70,
    141, 2136, 2206, 2214, 2310, 2322, 2360
\l__enumext_newlabel_arg_two_tl 26, 30, 67, 141,
    2160, 2170, 2184, 2200, 2215, 2297, 2302, 2307, 2323
\__enumext_parse_keys:n ... 53, 2730, 2755, 2755
\__enumext_parse_keys_vii:n 53, 3277, 3308, 3308
\__enumext_parse_keys_viii:n . 3666, 3700, 3700
\__enumext_parse_keys_series:n .. 53, 80, 1475, 1475,
    2763, 3314
\__enumext_parse_store_keys:n . 80, 2771, 2775,
    2775
\__enumext_parse_store_keys_vii:n . 92, 3317,
    3321, 3321
\l__enumext_parsep_i_skip 935, 937, 1086, 1134
\l__enumext_parsep_ii_skip .... 941, 943, 1092
\l__enumext_parsep_iii_skip ... 947, 949, 1098
\l__enumext_parsep_vii_skip ..... 3506
\l__enumext_parsep_viii_skip ..... 3850
\l__enumext_partopsep_v_skip .. 989, 993, 1160,
    1164, 1171, 1175, 1191, 1195
\l__enumext_partopsep_viii_skip ..... 1236
\__enumext_phantomsection: 30, 302, 331, 335, 351
\__enumext_print_footnote: ... 2457, 2480, 3524,
    3875
\__enumext_print_keyans_box:NN 64, 2025, 2025,
    2038, 2226, 2402, 2437, 3792, 3807
\l__enumext_print_keyans_i_tl .... 3926, 3955
\l__enumext_print_keyans_ii_tl ... 3931, 3956
\l__enumext_print_keyans_iii_tl .. 3936, 3957
\l__enumext_print_keyans_iv_tl ... 3941, 3958
\l__enumext_print_keyans_vii_tl .. 3946, 3959
\l__enumext_print_keyans_X_tl ..... 111
\__enumext_printkeyans:nnn 104, 3960, 3963, 3963
\__enumext_redefine_item: . 74, 2531, 2531, 2682
\l__enumext_ref_key_arg_tl 33, 37, 200, 488, 489,
    502, 533, 536, 547, 553, 564, 605, 606, 617
\l__enumext_ref_the_count_tl . 33, 37, 495, 498,
    501, 541, 543, 546, 558, 560, 563, 611, 613, 616
\__enumext_regex_counter_style: .. 27, 33, 195,
    195, 496, 542, 559, 612
\__enumext_register_counter_style:Nn .. 370,
    370, 375, 376, 377, 378, 379
\__enumext_remove_extra_parsep_vii: .. 3292,
    3533, 3533
\__enumext_remove_extra_parsep_viii: . 3680,
    3884, 3884
\__enumext_renew_footnote: ... 2457, 2461, 3476,
    3829
\l__enumext_renew_the_count_v_tl 614, 623, 625
\l__enumext_renew_the_count_vii_tl 544, 573,
    575
\l__enumext_renew_the_count_viii_tl 561, 580,
    582
\l__enumext_renew_the_count_X_tl ..... 37
\l__enumext_resume_active_bool 53, 56, 48, 1479,
    1599
\__enumext_resume_counter: .. 55, 56, 1597, 1603,
    1610
\__enumext_resume_counter:n . 53, 56, 1568, 1573,
    1597, 1597, 1667, 1675
\__enumext_resume_counter_save_ans: 56, 1597,
    1608, 1640
\__enumext_resume_counter_series: . 56, 1597,
    1606, 1623
\g__enumext_resume_int ... 48, 1520, 1614, 1615
\__enumext_resume_last:n 53, 54, 1475, 1481, 1494
\l__enumext_resume_name_tl 48, 1516, 1524, 1527,
    1543, 1551, 1554, 1600, 1601, 1629, 1636
\__enumext_resume_save_counter: 54, 1507, 1507,
    2923, 3364
\__enumext_resume_series:n . 55, 1439, 1564, 1564
\__enumext_resume_starred: . 57, 1440, 1661, 1661
\g__enumext_resume_vii_int .. 93, 48, 1547, 1619,
    1620
\__enumext_safe_exec: ..... 79, 2729, 2746, 2746
\__enumext_safe_exec_vii: ... 3276, 3297, 3297
\__enumext_safe_exec_viii: ... 3665, 3685, 3685
\l__enumext_series_name_tl ..... 56
\l__enumext_series_str . 54, 80, 1437, 1477, 1485,

```


1486, 1488, 1490, 1511, 1514, 1518, 1538, 1541, 1545,
 2759, 3312
 __enumext_set_error:nn 4041, 4051, 4053
 __enumext_set_parse:n 4024, 4041, 4041
 \l__enumext_setkey_tmpa_int . . . 106, 4017, 4021
 \l__enumext_setkey_tmpa_seq . . 106, 4015, 4025,
 4031, 4033, 4035, 4048
 \l__enumext_setkey_tmpa_tl . . . 106, 4023, 4027
 \l__enumext_setkey_tmpb_seq . . 106, 4016, 4019,
 4023, 4024
 \l__enumext_setkey_tmpb_tl 106, 4043, 4045, 4046
 \l__enumext_show_answer_bool . 122, 1886, 1910,
 2233, 2377, 2391, 3070, 3790
 __enumext_show_length:nnn . . 39, 203, 203, 4093,
 4094, 4095, 4096, 4097, 4098, 4099, 4100, 4101, 4102,
 4108, 4109, 4110, 4111, 4112, 4113, 4114, 4115, 4116,
 4117
 \l__enumext_show_position_bool 122, 1889, 1913,
 2237, 2381, 2392, 3071, 3794
 \g__enumext_standar_bool . 27, 20, 216, 219, 237,
 1509, 1574, 1586, 1612, 1625, 1663, 1801, 1812, 1842
 \l__enumext_standar_bool 83, 20, 2165, 2178, 2194,
 2752, 2922
 \l__enumext_standar_first_level_bool . 79, 20,
 242, 1496, 1643, 1694, 1704, 1760
 __enumext_standar_ref: 34, 486, 506, 2684
 __enumext_standar_ref:n 33, 478, 486, 486
 \g__enumext_standar_series_tl . 48, 1498, 1499,
 1665, 1668
 \g__enumext_standard_bool 79
 \l__enumext_standard_bool 79
 \l__enumext_standard_first_level_bool . . 28
 __enumext_standard_item_vii:w 94, 3404, 3406,
 3406
 __enumext_standard_item_viii:w . . 100, 3737,
 3739, 3739
 \g__enumext_starred_bool 27, 91, 93, 20, 225, 228,
 250, 1536, 1579, 1590, 1617, 1632, 1671, 1781, 1818,
 1843, 2156, 2166, 2196, 2291, 2810, 2823, 3272
 \l__enumext_starred_bool . 91, 93, 20, 2089, 2097,
 2181, 2222, 3305, 3363
 __enumext_starred_columns_set_vii: . . 3158,
 3158, 3285
 __enumext_starred_columns_set_viii: . 3548,
 3548, 3673
 \l__enumext_starred_first_level_bool 20, 255,
 1501, 1652, 1698, 1710, 1764
 __enumext_starred_item:nn . . 2508, 2508, 2537
 __enumext_starred_item_exec: . 101, 3782, 3782,
 3833
 __enumext_starred_item_vii:w . 94, 3403, 3422,
 3422
 __enumext_starred_item_vii_aux_i:w . . 3422,
 3427, 3430
 __enumext_starred_item_vii_aux_ii:w . 3422,
 3428, 3433, 3435
 __enumext_starred_item_vii_aux_iii:w 3422,
 3438, 3447
 __enumext_starred_item_viii:w 100, 101, 3736,
 3755, 3755
 __enumext_starred_item_viii_aux_i:w . . 101,
 3755, 3760, 3763
 __enumext_starred_item_viii_aux_ii:w . 101,
 3755, 3761, 3775, 3777
 __enumext_starred_joined_item_vii:n . 89, 94,
 3177, 3177, 3401
 __enumext_starred_joined_item_viii:n . . 97,
 100, 3567, 3567, 3734
 __enumext_starred_ref: 35, 531, 569, 2712
 __enumext_starred_ref:n 34, 525, 531, 531
 \g__enumext_starred_series_tl . 48, 1503, 1504,
 1673, 1676
 __enumext_start_from:NNn 36, 628, 628, 641, 663
 \l__enumext_start_i_int 1615, 1627, 1646
 __enumext_start_item_tmp_vii: 91, 3288, 3386,
 3386
 __enumext_start_item_tmp_viii: 98, 3676, 3719,
 3719
 __enumext_start_item_vii:w . 94, 95, 3414, 3419,
 3444, 3451, 3453, 3453
 __enumext_start_item_viii:w . . 100, 3747, 3752,
 3780, 3810, 3810
 __enumext_start_list:nn 29, 76, 88, 289, 291, 2733,
 2931, 3084, 3280, 3668
 __enumext_start_mini_vii: . 92, 3226, 3226, 3355
 __enumext_start_mini_viii: 99, 3616, 3616, 3711
 __enumext_start_store_level: . 80, 2732, 2804,
 2804
 __enumext_start_store_level_vii: . 93, 3279,
 3366, 3366
 \l__enumext_start_vii_int . . . 1620, 1634, 1655
 \l__enumext_start_X_int 85, 658
 __enumext_stop_item_tmp_vii: . 91, 93, 95, 3287,
 3291, 3388, 3455
 __enumext_stop_item_tmp_viii: . 98, 100, 3675,
 3679, 3721, 3812
 __enumext_stop_item_vii: 95, 96, 3455, 3509, 3509
 __enumext_stop_item_viii: 102, 3812, 3860, 3860
 __enumext_stop_list: . . 29, 289, 292, 2742, 2941,
 3097, 3293, 3682
 __enumext_stop_mini_vii: 90, 93, 3245, 3245, 3359
 __enumext_stop_mini_viii: . 99, 3616, 3635, 3715
 __enumext_stop_store_level: . . 80, 2743, 2804,
 2833
 __enumext_stop_store_level_vii: . . 93, 3294,
 3366, 3376
 \l__enumext_store_active_bool 24, 58, 80, 92, 97,
 1644, 1653, 1720, 2051, 2769, 2808, 2821, 2946, 2954,
 3042, 3101, 3315, 3368, 3378, 3694
 __enumext_store_addto_prop:n 63, 69, 1944, 1945,
 1953, 2076, 2272, 3785
 __enumext_store_addto_seq:n 63, 71, 1954, 1954,
 1958, 1965, 1979, 1987, 1996, 2014, 2022, 2140, 2365
 \l__enumext_store_ans_bool . 58, 132, 1721, 1743,
 1961, 1985, 1992, 2020, 2064
 \l__enumext_store_anskey_arg_tl . . 24, 66, 97,
 2082, 2091, 2093, 2099, 2107, 2110, 2120, 2125, 2128,
 2134, 2140
 __enumext_store_anskey_code:nnnn . 65, 2070,
 2074, 2074
 __enumext_store_anskey_show_left:n 68, 2081,
 2231, 2231
 __enumext_store_anskey_show_wrap:n 68, 2219,
 2219, 2235, 2250
 \l__enumext_store_columns_break_bool . 2045,
 2088
 \l__enumext_store_columns_join_int 97, 2096,
 2101
 \l__enumext_store_columns_sep_vii_bool 3336

<code>\l__enumext_store_columns_sep_vii_dim</code>	3341, 3345
<code>\l__enumext_store_columns_sep_X_bool</code>	.. 111
<code>\l__enumext_store_columns_sep_X_dim</code>	... 111
<code>\l__enumext_store_columns_vii_bool</code>	... 3323
<code>\l__enumext_store_columns_vii_int</code>	3328, 3332
<code>\l__enumext_store_columns_X_bool</code> 111
<code>\l__enumext_store_columns_X_int</code> 111
<code>__enumext_store_internal_ref:</code>	.. 65, 67, 2079, 2142, 2142
<code>\l__enumext_store_item_symbol_sep_dim</code>	2043, 2117, 2122
<code>\l__enumext_store_item_symbol_tl</code>	. 2041, 2108, 2112
<code>\l__enumext_store_keyans_item_opt_sep_-tl</code> 1875, 2266, 2268, 2339, 2343, 3768, 3770
<code>\l__enumext_store_keyans_item_opt_tl</code>	... 97
<code>\l__enumext_store_keyans_label_tl</code>	24, 69–71, 101, 97, 2255, 2258, 2261, 2268, 2270, 2272, 2329, 2332, 2335, 2341, 2346, 2356, 2365, 3765, 3770, 3771, 3784, 3785, 3787
<code>__enumext_store_level_close:</code>	. 63, 1959, 1983, 2837
<code>__enumext_store_level_close_vii:</code>	1990, 2018, 3382
<code>__enumext_store_level_open:</code>	.. 62, 63, 80, 1959, 1959, 2816, 2829
<code>__enumext_store_level_open_vii:</code>	.. 92, 1990, 1990, 3372
<code>\g__enumext_store_name_tl</code>	. 24, 60, 83, 97, 1814, 1820, 1827, 1832, 1837, 1844
<code>\l__enumext_store_name_tl</code>	. 24, 58–60, 97, 1530, 1533, 1557, 1560, 1648, 1657, 1691, 1692, 1707, 1713, 1722, 1724, 1726, 1728, 1730, 1732, 1758, 1814, 1820, 1947, 1949, 1956, 2208, 2209, 2245, 2312, 2313, 2416, 2429, 3802
<code>\l__enumext_store_opt_vii_tl</code>	. 1994, 2004, 2010, 2014, 3330, 3343
<code>\l__enumext_store_opt_X_tl</code> 111
<code>\l__enumext_store_ref_key_bool</code>	65, 1895, 2077, 2131, 2276, 2353
<code>\l__enumext_store_upper_level_X_bool</code>	.. 111
<code>\l__enumext_store_write_aux_file_tl</code>	26, 68, 70, 141, 2211, 2217, 2319, 2325
<code>__enumext_storing_exec:</code>	. 58, 1689, 1708, 1714, 1718
<code>__enumext_storing_set:n</code>	.. 58, 1684, 1689, 1689
<code>\l__enumext_the_counter_v_tl</code> 613
<code>\l__enumext_the_counter_vii_tl</code> 543
<code>\l__enumext_the_counter_viii_tl</code> 560
<code>\l__enumext_the_counter_X_tl</code> 37
<code>__enumext_tmp:n</code>	32, 36, 41, 47, 59, 66, 67, 72, 79, 84, 85, 96, 111, 121, 144, 148, 152, 171, 741, 745, 1433, 1444, 1680, 1688, 1735, 1753, 1865, 1900, 1901, 1918, 2144, 2151, 2152, 2173, 2187, 2190, 2202, 2278, 2285, 2654, 2693, 2694, 2726
<code>__enumext_tmp:nn</code>	402, 423, 424, 452, 453, 465, 658, 677, 722, 740, 798, 806, 807, 821, 886, 902, 903, 917, 1322, 1338, 1919, 1943, 2441, 2456
<code>__enumext_tmp:nnn</code>	466, 482, 483, 484, 485, 513, 529, 530
<code>__enumext_tmp:nnnnnn</code>	678, 703, 706, 709, 711, 713, 716, 719
<code>__enumext_tmp:w</code> 3909, 3912
<code>\l__enumext_tmpa_vii_int</code> 3168, 3171
<code>\l__enumext_tmpa_viii_int</code> 3558, 3561
<code>\l__enumext_tmpa_X_int</code> 152
<code>\l__enumext_topsep_v_skip</code>	975, 979, 1129, 1142, 1150, 1155, 1175, 1179, 3100, 3132
<code>\l__enumext_topsep_vii_skip</code>	.. 1206, 1215, 1219
<code>\l__enumext_topsep_viii_skip</code>	. 1228, 1250, 1254
<code>\l__enumext_vspace_a_star_v_bool</code> 1371
<code>\l__enumext_vspace_a_star_vii_bool</code>	... 1393
<code>\l__enumext_vspace_a_star_viii_bool</code>	... 1404
<code>\l__enumext_vspace_a_star_X_bool</code> 85
<code>__enumext_vspace_above:</code>	.. 51, 1339, 1339, 2842
<code>__enumext_vspace_above_v:</code>	. 51, 1367, 1367, 2970
<code>\l__enumext_vspace_above_v_skip</code>	.. 1369, 1373, 1375
<code>__enumext_vspace_above_vii:</code>	.. 52, 1389, 1389, 3352
<code>\l__enumext_vspace_above_vii_skip</code>	1391, 1395, 1397
<code>__enumext_vspace_above_viii:</code>	. 52, 1389, 1400, 3709
<code>\l__enumext_vspace_above_viii_skip</code>	1402, 1406, 1408
<code>\l__enumext_vspace_b_star_v_bool</code> 1382
<code>\l__enumext_vspace_b_star_vii_bool</code>	... 1415
<code>\l__enumext_vspace_b_star_viii_bool</code>	... 1426
<code>\l__enumext_vspace_b_star_X_bool</code> 85
<code>__enumext_vspace_below:</code>	.. 51, 1353, 1353, 2921
<code>__enumext_vspace_below_v:</code>	. 51, 1378, 1378, 3038
<code>\l__enumext_vspace_below_v_skip</code>	.. 1380, 1384, 1386
<code>__enumext_vspace_below_vii:</code>	.. 52, 1411, 1411, 3362
<code>\l__enumext_vspace_below_vii_skip</code>	1413, 1417, 1419
<code>__enumext_vspace_below_viii:</code>	. 52, 1411, 1422, 3717
<code>\l__enumext_vspace_below_viii_skip</code>	1424, 1428, 1430
<code>__enumext_widest_from:nnnn</code>	.. 36, 642, 642, 657, 669
<code>\g__enumext_widest_label_tl</code>	23, 31, 55, 390, 394, 398
<code>\l__enumext_wrap_label_opt_v_bool</code> 2550
<code>\l__enumext_wrap_label_opt_vii_bool</code>	94, 3413
<code>\l__enumext_wrap_label_opt_viii_bool</code>	.. 100, 3746
<code>\l__enumext_wrap_label_opt_X_bool</code> 85
<code>\l__enumext_wrap_label_v_bool</code>	2546, 2550, 2558, 2614
<code>\l__enumext_wrap_label_vii_bool</code>	94, 3412, 3417, 3425, 3493
<code>\l__enumext_wrap_label_viii_bool</code>	. 100, 3745, 3750, 3758, 3837
<code>\l__enumext_wrap_label_X_bool</code> 85
<code>__enumext_wrapper_label_v:n</code> 2616, 3079
<code>__enumext_wrapper_label_vii:n</code> 3496
<code>__enumext_wrapper_label_viii:n</code> 3840
<code>__enumext_zero_parsep:</code>	... 46, 1026, 1081, 1081
<code>enumext*</code> 5, 3274
<code>enumXi</code> 362
<code>enumXii</code> 362
<code>enumXiii</code> 362
<code>enumXiv</code> 362
<code>enumXv</code> 362

enumXvi [362](#)
enumXvii [362](#)
enumXviii [362](#)
Environments provide by **enumext**:
 enumext* 22, 23, 25–27, 31, 34, 35, 38, 39, 41, 42, 48, 49,
 52–55, 57–67, 70, 72, 73, 78, 79, 81, 91–93, 95, 96, 98,
 100, 102, 104, 107, 109
 enumext 22, 23, 25, 27, 31–35, 37–55, 57–67, 70, 72–74, 76,
 77, 79–81, 83, 84, 87, 89, 90, 93, 104, 107, 108
 keyans* 22–28, 31, 34–36, 38, 39, 41, 42, 48, 49, 52, 58, 59,
 61–63, 69, 73, 78, 99, 107, 109
 keyanspic 22–25, 28, 31, 32, 35, 49, 58, 59, 61, 63, 69–71,
 86–88, 108
 keyans 22–25, 27, 28, 31, 32, 35, 37–41, 43, 47, 49–51, 58,
 59, 61–63, 69–71, 75–77, 83, 84, 86–88, 90, 100, 107,
 108
Environments:
 list 26, 29, 76, 77, 79
 lrbox 89, 95, 96, 102
 minipage 26, 29, 42, 44, 86–89, 95, 96, 102
 multicols 42–45, 49, 82–85
exp commands:
 \exp_after:wN 3912
 \exp_args:Ne 2766, 3900
 \exp_not:N . 45, 393, 501, 546, 563, 616, 755, 769, 770,
 781, 782, 793, 794, 2136, 2242, 2243, 2358, 2413, 2414,
 2426, 2427, 3799, 3800, 3909
 \exp_not:n 245, 258, 270, 277, 284, 501, 502, 546, 547,
 563, 564, 616, 617, 756, 1461, 1473, 1927, 1934, 2101,
 2112, 2122, 2136, 2137, 2214, 2322, 2360, 2362, 2786,
 2799, 3332, 3345
F
\fbbox 1870
file commands:
 \file_input_stop: 4232
first [807](#)
font [402](#)
\footnote 73
\footnote 73, 2465
\footnotemark 2475
\footnotesize 2243, 2414, 2427, 3800
\footnotetext 2459
G
\getkeyans 14, 103, [3898](#)
group commands:
 \group_begin: . . 2063, 2241, 2412, 2425, 3472, 3491,
 3798, 3825, 3835, 3920, 3954
 \group_end: 2072, 2248, 2419, 2432, 3501, 3513, 3805,
 3845, 3864, 3922, 3961
H
\hbadness 3520, 3871
hbox commands:
 \hbox_set:Nn 382
\hfill 432, 436, 441, 442, 1293, 1311, 2136, 2358, 3250, 3640
hook commands:
 \hook_gput_code:nnn 9, 179, 183, 300
 \hook_gset_rule:nnnn 301
\hspace 3531, 3882
\hyperlink 66, 71
\hyperlink 2136, 2358
\hypertarget 30
\hypertarget 330

I

\IfHyperBoolean 308
\IfPackageLoadedTF 11, 304, 318
\ignorespaces 758
\inputlineno 245, 258, 270, 277, 284
int commands:
 \int_add:Nn 3210, 3600
 \int_case:nn 932, 1083, 1776, 1798
 \int_compare:nNnTF . 534, 551, 571, 578, 1008, 1127,
 1272, 1276, 1280, 1825, 1829, 1850, 1856, 2055, 2059,
 2256, 2295, 2300, 2305, 2330, 2408, 2750, 2760, 2813,
 2826, 2865, 2881, 2895, 2909, 2955, 2959, 2988, 3013,
 3026, 3046, 3050, 3106, 3180, 3190, 3206, 3301, 3370,
 3380, 3526, 3535, 3570, 3580, 3596, 3688, 3695, 3877,
 3886, 4021
 \int_compare_p:nNn . . . 217, 226, 238, 239, 251, 252,
 1782, 2157, 2167, 2179, 2180, 2195, 2197
 \int_decr:N 3209, 3599
 \int_eval:n 1949, 2209, 2243, 2313, 2414, 2427, 2669,
 2711, 3198, 3588, 3800
 \int_from_alph:n 636, 650
 \int_from_roman:n 638, 652
 \int_gadd:Nn 3211, 3601
 \int_gdecr:N 1785, 1789, 1792, 1795, 1803
 \int_gincr:N 1614, 1619, 2068, 2368, 2496, 2526, 2564,
 2855, 2979, 3068, 3390, 3468, 3723, 3789
 \int_gset:Nn 2473
 \int_gset_eq:NN 1513, 1520, 1526, 1532, 1540, 1547,
 1553, 1559, 2470
 \int_gzero:N 1301, 1318, 1840, 1841, 1862, 2914, 3031,
 3544, 3895
 \int_if_exist:NnTF 1488, 1524, 1530, 1551, 1557, 1730
 \int_incr:N 2749, 2950, 3105, 3300, 3389, 3687, 3722
 \int_mod:nn 3537, 3888
 \int_new:N . 20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 108,
 116, 129, 130, 136, 137, 138, 149, 155, 156, 157, 158,
 159, 1490, 1732
 \int_set:Nn 632, 636, 638, 1627, 1634, 1646, 1655, 1924,
 2096, 3145, 3146, 3168, 3179, 3185, 3201, 3520, 3558,
 3569, 3575, 3591, 3871, 4017
 \int_set_eq:NN . 1615, 1620, 2781, 3208, 3327, 3598
 \int_step_function:nnN 2173, 2187, 2202
 \int_step_inline:nnn 3147, 4044
 \int_to_roman:n 187, 2153, 2191
 \int_use:N . 1009, 1629, 1636, 1648, 1657, 2669, 2688,
 2711, 2767, 2866, 2875, 2890, 2896, 3183, 3184, 3196,
 3573, 3574, 3586
 \int_zero:N 3529, 3880
 \c_one_int . 3168, 3187, 3193, 3199, 3203, 3206, 3558,
 3577, 3583, 3589, 3593, 3596
 \c_zero_int 2157, 2167, 2179, 2180, 2195, 2197, 3370,
 3380, 3540, 3891
\item 29, 40, 41, 64, 73, 86, 88, 89, 91, 98
\item 73, 75, 93, 95, 100, 102, 293, 1967, 1973, 1998, 2006, 2093,
 2332, 2335, 2533, 2568, 3286, 3288, 3674, 3676, 3787
\item* 6, 12, 61, [2566](#)
item-pos* [2441](#)
item-sym* [2441](#)
\itemindent 23, 77
\itemindent 77
itemindent [722](#)
\itemsep 87, 88
\itemsep 3121, 3127
\itemwidth 3175, 3219, 3223, 3565, 3609, 3613

K

keyans 12, 2926

keyans* 12, 3663

keyanspic 13, 3081

Keys for environments provide by [enumext](#):

 above* 24, 50–52

 above 24, 50–52, 81, 84, 92, 99

 after 39–41, 83, 86, 93, 99

 align 24, 32, 76, 95

 before* 39–41, 81, 92, 99

 before 39–41, 84

 below* 24, 50–52

 below 24, 50–52, 83, 86, 93, 99

 check-ans 24, 25, 27, 28, 58–61, 65, 71, 73, 74, 81, 83, 96, 107

 columns-sep* 25, 62, 80, 92

 columns-sep 41, 63, 80, 82, 85, 92

 columns* 25, 62, 80, 92

 columns 24, 41, 42, 44, 50, 63, 80, 82, 85, 92

 first 39–41, 95

 font 32, 76, 95

 item-pos* 65, 66, 72

 item-sym* 23, 65, 66, 72, 74

 item*-sep 74

 itemindent 24, 38, 75, 95

 itemsep 37, 78

 labelsep 32, 74, 77, 95

 labelwidth 31–36, 77

 label 23, 31, 33, 36, 89

 lisparindent 78

 list-indent 23, 38, 88

 list-offset 38

 listparindent 38, 95

 mark-ans 25, 61, 68

 mark-pos 61, 62

 mark-ref 25, 61, 67

 mini-env .. 24, 41, 44, 49, 50, 73, 81, 84, 90, 92, 98, 99

 mini-sep 24, 41, 81, 84

 miniright* 24, 42

 miniright 24, 42, 49, 91

 minirigth* 27

 minirigth 27

 no-store 25, 58, 59

 noitemsep 37, 46

 nosep 37, 46

 parindent 78

 parsep 37, 78, 95

 partopsep 37

 ref 23, 27, 33–35, 107

 resume* 23, 52, 53, 57, 58, 83

 resume 23, 52–58, 83, 93

 rightmargin 38

 save-ans 24, 53–59, 63, 65, 69, 70, 75, 80, 84, 86, 93, 100, 101, 103, 104, 107

 save-key 25, 53, 80

 save-ref 26, 30, 61, 65–67, 69, 71, 75, 101

 save-sep 61, 101

 series 23, 52–57, 80, 83

 show-ans 25, 61, 62, 64, 65, 68, 75, 101

 show-length 27, 39, 107

 show-pos 25, 61, 62, 64, 65, 68, 71, 75, 101

 start 24, 27, 36, 37, 53

 store-brk 65, 66

 topsep 37

 widest 23, 27, 36, 37

wrap-ans 61, 64, 68

wrap-label* 32, 73, 76, 94, 95, 100

wrap-label 32, 76, 94, 95, 100

wrap-opt 61

keys commands:

 \keys_define:nn 404, 426, 455, 468, 515, 586, 660, 680, 724, 743, 800, 809, 888, 905, 1324, 1435, 1682, 1737, 1867, 1903, 1921, 2039, 2443, 3924, 3987

 \l_keys_key_str 4078

 \keys_set:nn . 418, 912, 1329, 1334, 1576, 1581, 1668, 1676, 2085, 2762, 2766, 2966, 3313, 3704, 3989, 3990, 3991, 3992, 3993, 3994, 3995, 3996, 3997, 3998, 3999, 4000, 4038

keyval commands:

 \keyval_parse:NNn 1449

L

label 466, 513, 586

Labels provide by [enumext](#):

 \Alph* 31

 \Roman* 31

 \alph* 31

 \arabic* 27, 31

 \roman* 31

 \labelsep 88

 \labelsep 3122, 3125

 labelsep 402

 \labelwidth 31, 88

 \labelwidth 3122, 3123

 labelwidth 402

 \leftmargin 23, 77

 \leftmargin 77, 3122

legacy commands:

 \legacy_if:nTF 3456, 3459, 3813, 3816

 \legacy_if_gset_false:n 1002

 \legacy_if_set_false:n 3458, 3815

 \legacy_if_set_true:n 3418, 3443, 3450, 3463, 3751, 3779, 3820

 \linewidth 81, 84

 \linewidth 2850, 2976, 3144, 3171, 3232, 3561, 3622

 \list 29

 \list 291

 list-indent 722

 list-offset 722

 \listparindent 3124

 listparindent 722

 \lrbbox 3473, 3826

M

\makebox 89

\makebox .. 2029, 2031, 2588, 3487, 3495, 3499, 3839, 3843

\makelabel 73, 76, 89

\makelabel 76, 2594, 2610

\makesavenoteenv 324

mark-ans 1865

mark-pos 1865, 1901

mark-ref 1865

mini-env 886

mini-sep 886

\minipage 29

\minipage 297

\miniright 10, 49, 1270, 2912, 3029

\miniright* 10

mode commands:

 \mode_if_vertical:TF 957, 985, 1108, 1187

\mode_leave_vertical: .. 755, 769, 781, 793, 1998, 2006, 2027, 2586, 3485

msg commands:

\msg_error:nn .. 2957, 2961, 3048, 3108, 3303, 3690, 3697, 4001

\msg_error:nnn 491, 538, 555, 608, 1274, 1278, 1303, 1320, 1588, 1592, 1696, 1700, 3914, 3919, 3984, 4054

\msg_error:nnnn 1762, 1766, 2053, 2057, 2061, 2948, 3044, 3052

\msg_fatal:nn 2751

\msg_fatal:nnn 356

\msg_info:nnn 13, 16, 306, 320

\msg_line_context: .. 4082, 4087, 4092, 4107, 4122, 4126, 4130, 4134, 4160, 4165, 4170, 4175, 4180, 4184, 4189, 4194, 4198, 4203, 4207, 4212, 4217, 4222, 4226, 4230

\msg_new:nnn 4055, 4059, 4063, 4067, 4072, 4076, 4080, 4085, 4090, 4105, 4120, 4124, 4128, 4132, 4136, 4145, 4150, 4154, 4158, 4163, 4168, 4173, 4178, 4182, 4187, 4192, 4196, 4201, 4205, 4210, 4215, 4220, 4224, 4228

\msg_note:nnnn 1706, 1712

\msg_term:nnn 1832

\msg_term:nnnn 2678, 2688, 2717, 2722

\msg_warning:nn 2911, 3028

\msg_warning:nnnn 1835, 1853, 1859, 2626, 2631, 3182, 3195, 3572, 3585

\multicolsep 82, 85

\multicolsep 2880, 3001

N

\NeedsTeXFormat 3

\newcounter 359

\NewDocumentCommand 1270, 2049, 3040, 3898, 3952, 4008

\NewDocumentEnvironment . 2727, 2926, 3081, 3274, 3663

\newlabel 30

\newlabel 342

no-store 1735

\noindent 91, 98

\noindent . 2857, 2981, 3241, 3287, 3528, 3631, 3675, 3879

\nointerlineskip 2857, 2981, 3241, 3631

noitemsep 678

\nopagebreak 968, 996, 1119, 1198, 1261, 1267

\normalfont 2242, 2413, 2426, 3799

nosep 678

P

Packages:

enumext 22, 33, 57, 77, 86, 106

enumitem 30, 31

expl3 89

footnotehyper 30

hyperref 25, 26, 29, 30, 66, 71, 95, 106

lua-visual-debug 44

multicol 22, 106

shortlst 89

\par 968, 996, 1119, 1198, 1261, 1267, 1296, 1313, 2221, 2901, 2916, 3018, 3033, 3156, 3259, 3266, 3528, 3542, 3649, 3656, 3879, 3893

\parindent 3505, 3849

\parsep 43, 46, 87, 88

\parsep 1999, 2007, 2708, 3121, 3128, 3133

parsep 678

\parskip 3506, 3850

\partopsep 88

\partopsep 2709, 3126

partopsep 678

peek commands:

\peek_meaning:NTF 3395, 3409, 3426, 3437, 3728, 3742, 3759

\peek_meaning_remove:NTF 3402, 3735

\peek_remove_spaces:n 2572

\phantomsection 30

\phantomsection 331

prg commands:

\prg_do_nothing: 335

\prg_new_protected_conditional:Npnn ... 189

\prg_replicate:nn 206, 4142

\prg_return_false: 193

\prg_return_true: 192

\printkeyans 14, 104, 3952

prop commands:

\prop_count:N 1949, 2209, 2245, 2313, 2416, 2429, 3802

\prop_gput_if_not_in:Nnn 1944, 1947

\prop_if_exist:NTF 1722, 3918

\prop_item:Nn 3921

\prop_new:N 1724

\ProvidesExplPackage 4

R

\raggedcolumns 2889, 3007

\ref 67, 69

ref 466, 513, 586

\refstepcounter 3465, 3822

regex commands:

\regex_match:nnTF 191, 635, 637, 649, 651, 2779, 2792, 3325, 3338

\regex_replace_once:nnN 199

\renewcommand 501, 546, 563, 616

\RenewDocumentCommand ... 2465, 2533, 2568, 2594, 2610

\RequirePackage 17

resume 1433

resume* 1433

rightmargin 722

\Roman 31, 36

\Roman 378

\roman 31, 36

\roman 379, 484, 3940

S

save-ans 1680

save-ref 1865

save-sep 1865

scan commands:

\scan_stop: 88, 3135, 3286, 3674, 3909, 3912

seq commands:

\seq_clear:N 4015

\seq_const_from_clist:Nn 4003

\seq_count:N 3094, 4019

\seq_gclear:N 2463, 2464

\seq_gput_right:Nn 1956, 2476, 2477

\seq_if_empty:NTF 2482, 3967, 4033

\seq_if_exist:NTF 1726, 3965

\seq_item:Nn 3153

\seq_map_function:NN 4024

\seq_map_inline:Nn .. 3973, 3978, 4012, 4034, 4035

\seq_map_pairwise_function:NNN 2484

\seq_new:N 109, 110, 127, 150, 151, 1728

\seq_pop_left:NN 4023

\seq_put_right:Nn 3054, 4031, 4048

\seq_set_from_clist:Nn 4016

\seq_set_map_e:NNn 4025

\seq_show:N	3969
series	1433
\setcounter	646, 650, 652, 2669, 2711, 3099
\setenumext	6–9, 105, 3928, 3933, 3938, 3943, 3948, 4008
\setlength	2000, 2008
show-ans	1865, 1901
show-length	798
show-pos	1901
skip commands:	
\skip_add:Nn	937, 943, 949, 959, 963, 987, 991, 1088, 1094, 1100, 1110, 1114, 1136, 1189, 1193, 3121
\skip_eval:n	1999, 2007
\skip_gset:Nn	1209, 1213, 1217
\skip_gzero_new:N	1204, 1205
\skip_horizontal:N	770, 782, 794, 3488, 3502, 3846
\skip_horizontal:n	756, 2028, 2036, 2587, 2589, 3486, 3854
\skip_if_eq:nnTF	935, 941, 947, 1011, 1045, 1086, 1092, 1098, 1129, 1134, 1155, 1206, 1228, 1341, 1355, 1369, 1380, 1391, 1402, 1413, 1424
\skip_new:N	69, 70, 74, 75, 76, 77, 78, 131, 169
\skip_set:Nn	920, 924, 973, 977, 1014, 1018, 1022, 1029, 1033, 1037, 1048, 1053, 1057, 1063, 1068, 1073, 1131, 1132, 1133, 1140, 1144, 1148, 1157, 1162, 1166, 1169, 1173, 1177, 1208, 1212, 1230, 1234, 1238, 1244, 1248, 1252, 3115, 3129
\skip_set_eq:NN	2667, 2707, 2708, 3505, 3506, 3849, 3850
\skip_use:N	922, 926, 961, 965, 969, 989, 993, 1012, 1031, 1040, 1046, 1051, 1055, 1066, 1070, 1071, 1076, 1112, 1116, 1142, 1342, 1346, 1349, 1356, 1360, 1363, 2901
\skip_zero:N	2709, 2880, 3001, 3126, 3127
\skip_zero_new:N	1124, 1125, 1126, 1203, 1225, 1226, 1227
\c_zero_skip	935, 941, 947, 1012, 1046, 1086, 1092, 1098, 1129, 1134, 1155, 1206, 1228, 1342, 1356, 1369, 1380, 1391, 1402, 1413, 1424
\small	3930, 3935, 3940, 3945, 3950
\star	2447
start	658
\stepcounter	2469, 3061
str commands:	
\c_backslash_str	4152, 4156, 4165, 4166, 4170, 4171, 4175, 4176, 4207, 4208, 4212, 4217, 4218
\c_colon_str	2208, 2312, 3909
\str_case:nn	211, 264
\str_case:nnTF	1456, 1465
\str_clear:N	2759, 3312
\str_count:n	206, 4142
\str_if_empty:N	1477, 1518, 1545
\str_if_eq:nnTF	2670, 2713
\str_if_in:nnTF	3905
\str_new:N	126, 164
\str_set:Nn	458, 459, 460, 1882, 1883, 1906, 1907
\string	324
\strutbox	1016, 1020, 1024, 1035, 1039, 1050, 1059, 1065, 1075, 1088, 1094, 1100, 1131, 1132, 1133, 1136, 1146, 1150, 1159, 1166, 1171, 1179, 1208, 1209, 1212, 1219, 1232, 1240, 1246, 1254, 3131

T

TeX and \LaTeX commands:

\@auxout	340
\@currenvir	211, 264

\protected@write	340
text commands:	
\text_expand:n	3901
\textasteriskcentered	1879, 1893
\thepage	346
tl commands:	
\c_space_tl	2394, 4092, 4107
\tl_clear:N	431, 437, 1863, 2082, 2255, 2329, 3765
\tl_clear_new:N	388
\tl_const:Nn	37, 372
\tl_gclear:N	1498, 1503, 1844, 1845, 2605, 3270, 3489, 3660
\tl_gclear_new:N	1485
\tl_gput_right:Nn	373
\tl_greplace_all:Nnn	394
\tl_gset:Nn	243, 256, 1486, 1499, 1504, 1814, 1820, 3432
\tl_gset_eq:NN	390, 2514, 3482
\tl_if_blank:nTF	3480
\tl_if_empty:N	489, 508, 536, 553, 573, 580, 606, 623, 1511, 1516, 1538, 1543, 1601, 1665, 1673, 1692, 1758, 1827, 1963, 1994, 2108, 2266, 2339, 2388, 2584, 3768, 4046
\tl_if_empty:nTF	1566
\tl_if_exist:N	1571
\tl_if_novalue:nTF	2083, 2094, 2263, 2337, 2373, 2467, 2492, 2510, 2515, 2544, 2757, 3092, 3310, 3702, 3766, 4010
\tl_map_inline:Nn	197, 391
\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, 134, 135, 141, 142, 143, 146, 163, 166
\tl_put_left:Nn	1971, 2004, 2091, 2400, 2435, 3784, 3787
\tl_put_right:Nn	389, 499, 544, 561, 614, 1925, 1932, 1975, 2010, 2093, 2099, 2107, 2110, 2120, 2125, 2128, 2134, 2160, 2170, 2184, 2200, 2206, 2211, 2258, 2261, 2268, 2270, 2297, 2302, 2307, 2310, 2319, 2332, 2335, 2341, 2346, 2356, 2784, 2797, 3330, 3343, 3770, 3771, 3926, 3931, 3936, 3941, 3946
\tl_remove_all:Nn	4045
\tl_remove_once:Nn	2148, 2282
\tl_replace_all:Nnn	393
\tl_reverse:N	2147, 2149, 2281, 2283
\tl_set:Nn	45, 268, 275, 282, 358, 432, 436, 441, 442, 488, 533, 605, 753, 767, 779, 791, 1600, 1691, 2239, 2375, 2410, 2423, 2512, 3773, 3796, 4043
\tl_set_eq:NN	399, 494, 497, 541, 543, 558, 560, 611, 613, 2146, 2280, 2293, 2556, 2560, 3073, 3075
\tl_to_str:n	1571, 1577, 1582, 3901
\tl_trim_spaces:n	389, 4031, 4043, 4049
\tl_use:N	395, 398, 510, 575, 582, 625, 824, 828, 832, 836, 840, 844, 848, 852, 856, 860, 864, 868, 872, 876, 880, 884, 2033, 2153, 2161, 2172, 2186, 2191, 2203, 2499, 2505, 2529, 2547, 2551, 2559, 2596, 2597, 2604, 2612, 2613, 2619, 2734, 2932, 3078, 3264, 3492, 3503, 3507, 3654, 3836, 3847, 3853, 3857, 3955, 3956, 3957, 3958, 3959, 4027
token commands:	
\token_to_str:N	342
\topsep	2000, 2008
topsep	678
\typeout	310, 313, 323, 324

U

\u	200
----	-----

use commands:		1395, 1397, 1406, 1408, 1417, 1419, 1428, 1430, 1999,
\use:N	207, 2601, 2736	2007, 3089, 3100, 3543, 3894
\use:n	1447, 3907	
\use_none:nn	334	
\usecounter	2668, 2710	
V		
\value	1514, 1520, 1527, 1533, 1541, 1547, 1554, 1560	
\vspace	1003, 1346, 1349, 1360, 1363, 1373, 1375, 1384, 1386,	
W		
widest	658	
wrap-ans	1865	
wrap-label	402	
wrap-label*	402	
wrap-opt	1865	