

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 multicol	4	4.5	The environment <code>keyans</code>	11
1.3.3	Support for minipage	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 label and ref	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 add code	8	10	Implementation	21
3.4	Keys for start, series and resume	8	11	Index of Implementation	109
3.5	Keys for multicol	9			
3.6	Keys for minipage	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 multicol, 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 (l^{pp}l), version 1.3 or later (<https://www.latex-project.org/lppl.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[3] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TE_X. It has been tested with the standard classes provided by L^AT_EX: `book`, `report`, `article` and `letter` on 10pt, 11pt and 12pt.

1 Introduction

In the \LaTeX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like `exam`[1] and packages like `xsim`[2] do the job perfectly, but they don’t always fit the basic day to day needs.

In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

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

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

Or we are interested in referring to a specific question and its “answer”, for example:

The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.

Or we are interested in printing all the “answers”:

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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

(A) value

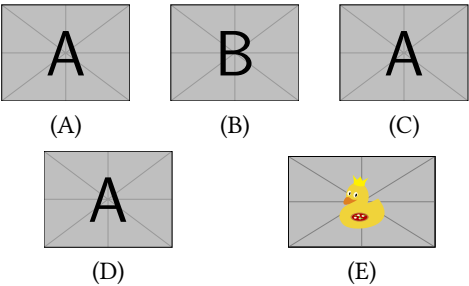
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
-

Where what we are interested in the `<label>` and a “short note” that we leave as an explanation, and then print them:

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

These “simple worksheets” or “multiple choice questions” appear to be easy to obtain using a combination of the `enumerate`, `minipage` and `multicols` environments, but like many things, what “looks simple” is not so simple.

The `enumext` package was created and designed to meet these small requirements in the creation of “simple worksheets” and “multiple choice questions”.

1.1 Description and usage

The `enumext` package defines enumerated environments using the `list` environment provided by \LaTeX , but “does not redefine” any internal commands associated with it such as `\list`, `\endlist` or `\item` outside of the “scope” in which they are defined.

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[5], the approach is intended to work without hindering either of them.
This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex>dvips>ps2pdf` and is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktxlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[5] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.



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

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



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

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



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

1.3 User interface

The user interface consists in `enumext`, `enumext*`, `keyans`, `keyans*` and `keyanspic` environments, `\anskey`, `\item*` and `\anspic*` commands to \langle stored content \rangle , `\getkeyans` command to get the individual \langle stored content \rangle , `\printkeyans` to print all \langle stored content \rangle , `\miniright` for `minipage` and `\setenumext` to config all $[(key = val)]$ options.

1.3.1 Internal counters

The package `enumext` uses internally the `enumXi`, `enumXii`, `enumXiii`, `enumXiv` counters for the four nesting levels of the `enumext` environment, the `enumXv` counter for the `keyans` environment, the `enumXvi` counter for the `keyanspic` environment, the counter `enumXvii` for `enumext*` environment and the counter `enumXviii` for `keyans*` environment.

- If any package defines these counters or they are user-defined in the document, the package will return a missing error and abort the load.

1.3.2 Support for multicol

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

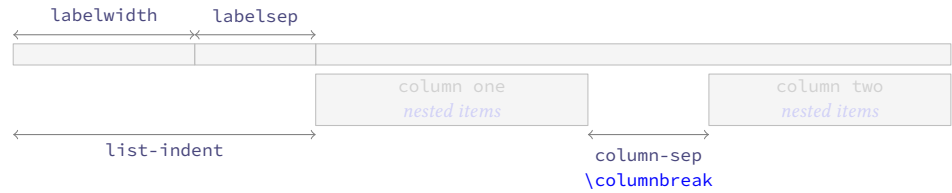


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §3.5).

- The `\columnseprule` command is not available as a key and is set to “zero” for the inner levels and the `keyans` environment. If the value of this is set inside the document, it will affect “all environments” that use the `columns` key.

1.3.3 Support for minipage

The package provides direct support for `minipage` environment, this allows you to obtain an output like the one shown in figure 5.



Figure 5: Representation of the `mini-env` output for a nested level `enumext` environment.

The `minipage` environments (left and right) is always used with “aligned on top” [`t`], the `minipage` environment on the “right side” always starts with `\centering`. It can be used at all nesting levels and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the “left” side `minipage` environment to the “right” side one must use the command `\miniright` (see §3.6).

1.3.4 The \label and \ref system

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§3.1), the standard \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. T_EX will enter *vertical mode* and apply this value to the “top” and “bottom” the environment or nested level.

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

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

`noitemsep` *<value forbidden>* default: *not used*

This is a “meta-key” that does not receive an argument. Set `itemsep` and `parsep` equal to 0pt the entire level of environment.

`nosep` *<value forbidden>* default: *not used*

This is a “meta-key” that does not receive an argument. Sets all keys for vertical spacing equal to 0pt the entire level of environment.

- The following *<keys>* should be used with “caution”, they are intended to be used at the “top” and “bottom” of the environment when the `columns` or `mini-env` keys do not provide adequate *vertical spaces*. The values passed can be *rubber* or *rigid* lengths, the way they are applied is the way you differ, using the star ‘*’ *<keys>* applies `\vspace*` so that T_EX does *not discard* this space at page break.

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

Set the *extra vertical space* added, beyond `topsep`, to the top of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space on the “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace` so is “discardable”.

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

Set the *extra vertical space* added, beyond `topsep`, to the top of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space on the “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace*` so is “not discardable”.

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

Set the *extra vertical space* 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* 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 {content}`

The `\anskey` command takes a mandatory argument and is triggered by `save-ans` key. The “content” are “stored” in *store name* set by `save-ans` key. The command does “not support” verbatim content and must NOT be nested. By design it is assumed that each `\item` or `\item*` will have a “single” occurrence of the command unless a nested level is opened or the `no-store` key is used. If `save-ref` key are active and the `hyperref`[7] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by L^AT_EX will be used.

Example

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

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

4.5 The environment keyans

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

The `keyans` is an “*enumerated list*” environment designed for “*multiple choice*” questions activated by the `save-ans` key. This environment can NOT be nested and must always be at the “*first level*” of the `enumext` environment, the commands `\item` and `\item[\langle custom \rangle]` work in the usual.

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

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

4.5.1 The `\item*` in `keyans`

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

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

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

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

Example

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

  \item Text containing a question and image.
    \begin{keyans}[nosep,mini-env={0.4\linewidth}]
      \item Choice
      \item Choice
      \item Choice
      \item Choice
      \item*[\note] Correct choice
      \miniright
      \includegraphics[scale=0.25]{example-image-a}

      Some text
    \end{keyans}
\end{enumext}
```

1. Text containing a question.

(A) Choice

* (B) Correct choice

(C) Choice

(D) Choice
2. Text containing a question and image.

(A) Choice

(B) Choice

(C) Choice

(D) Choice

* (E) [note] Correct choice



Some text

4.6 The environment keyanspic

keyanspic

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

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

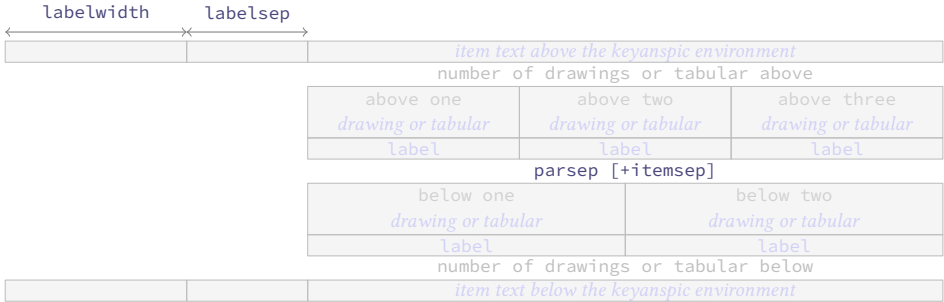


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

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

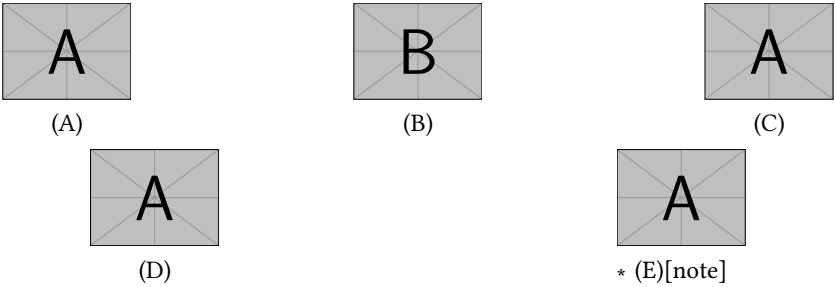
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

The “content” can only be accessed “after” it is stored, if the `<store name>` does not exist the command will return an error. The form taken by the argument `<store name> : <position>` is the same as that used to generate the internal “label and ref” system when `save-ref` key are active, so to refer to a stored “content”. For example `\getkeyans{test:4}` will return the “stored content” at position 4 of the environment in which the key `save-ans=test` was set.

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

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

(a) ~~LaTeXe~~ is cool?

[2] Very True!
3. Related to Linux

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

(b) Rate the following package and class

i. xsim

[4] very good

ii. exsheets

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

5 Full examples

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

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

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

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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


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

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

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

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

Example 2

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

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

A 36 km/h.

☒ B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

A 36 km/h.

☒ B 360 km/h.

C 27,8 km/h.

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

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

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

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

D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
1. B
2. A
3. B
4. A
- *
- *
- *
- *

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

©2024 by Pablo González L

15 / 121

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

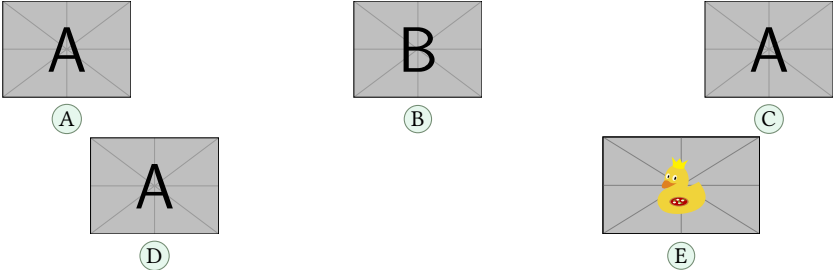
 II only
- C

 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:



- D

 I and III only
- E

 I, II, and III

- D

 value
- E

 value

5. Question with image on left side:
- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

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

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

*

*

*

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

 True False
- (a) $\alpha > \delta$
- (b) ~~ETX~~ze is cool?
- 4

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

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

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

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

*

*

*

*

*

Example 5

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

<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 nosep key to reduce the vertical spaces in the left side example and set the label key in mathematical mode for the right side as \ast, \diamond, \circ and \star for the four levels together with the nosep key

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

Fake description environment

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

Something long A much longer description text may take more than one line or more than one paragraph.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

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

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

LoNg vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[4]` package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in [Create a fake label ref using list](#) and the answer given by David Carlisle in [Change the use of label ref by data save in an array \(list\)](#) I managed to create a more solid code than the original version, now using the `l3prop`[10] and `l3seq`[10] modules together with the `hyperref`[7] and `enumitem`[5] packages, which did the job, but with some limitations.

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

Why list environments?

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

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

Why not random questions and other utilities

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

7 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [5] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [6] BERRY, KARL. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [7] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [8] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [9] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [10] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.
- [12] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [13] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [14] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

8 Change history

v1.0 2024-05-22 – First public release.

9 Index of Documentation

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

C

Document class:

article

book

exam

letter

report

\columnbreak

\columnsep

Commands provide by enumext:

\anskey

\anspic*

\anspic

\getkeyans

\item*

\item

\miniright

\printkeyans

\setenumext

Counters defined by enumext:

enumXiii

enumXii

enumXiv

enumXi

enumXviii

enumXvii

enumXvi

enumXv

E

Environments provide by enumext:

enumext*

enumext

keyans*

keyanspic

keyans

Environments:

enumerate

list

minipage

multicols

I

\item

\itemsep

K

Keys for environments provide by enumext:

above*

above

after

align

before*

before

below*

below

check-ans

columns-sep

columns

first

font

item-pos*

item-sym*

itemindent

itemsep

labelsep

labelwidth

label

list-indent

list-offset

listparindent

mark-ans

mark-pos

mark-ref

mini-env

mini-sep

miniright*

miniright

no-store

noitemsep

nosep

parsep

partopsep

ref

resume*

resume*

resume

rightmargin

save-ans

save-key

save-ref

save-sep

series

show-ans

show-length

show-pos

start

topsep

widest

wrap-ans

wrap-label*

wrap-label

wrap-opt

L

\label

Labels provide by enumext:

\Alph*

\Roman*

\alph*

\arabic*

\roman*

\labelsep

\labelwidth

\linewidth

\listparindent

P

Packages:

enumerate

enumext

enumitem

©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		
\partopsep	8	\topsep	8

10 Implementation

The most recent publicly released version of `enumext` is available at CTAN: <https://www.ctan.org/pkg/enumext>. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: <https://github.com/pablgonz/enumext/issues>.

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a T_EXpert are superfluous, but, after so many years developing this project is the only way to remember what does what.

10.1 General conventions

Variables containing `i`, `ii`, `iii` and `iv` are associated by level with the `enumext` environment, variables containing `v` are associated with the `keyans` environment, variables containing `vi` are associated with the `keyanspic` environment, variables containing `vii` are associated with the `enumext*` environment and variables containing `viii` are associated with the `keyans*` environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital “X”.

The temporary function `__enumext_tmp:n` is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

10.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

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

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

10.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-22}
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
\g__enumext_check_ans_bool
\l__enumext_check_start_line_env_tl
\g__enumext_start_line_tl
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int
```

Internal variables used by “check answer” mechanism (§10.23) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
132 \bool_new:N \l__enumext_store_ans_bool
133 \bool_new:N \l__enumext_check_ans_bool
134 \bool_new:N \g__enumext_check_ans_bool
135 \tl_new:N \l__enumext_check_start_line_env_tl
136 \tl_new:N \g__enumext_start_line_tl
137 \tl_new:N \g__enumext_envir_name_tl
138 \int_new:N \g__enumext_check_starred_cmd_int
139 \int_new:N \g__enumext_item_anskey_int
140 \int_new:N \g__enumext_item_number_int
```

(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`.

```
141 \bool_new:N \l__enumext_hyperref_bool
142 \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.

```

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

```

151 \int_new:N \g__enumext_footnote_int
152 \seq_new:N \g__enumext_footnote_arg_seq
153 \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.

```

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

```

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

```

(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.

```

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

```

(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.

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

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

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

```
187 \cs_new:Nn \__enumext_level:
188 {
189     \int_to_roman:n { \l__enumext_level_int }
190 }
```

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

(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.

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

(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.

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

(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.

```
211 \cs_new_protected:Nn \__enumext_is_not_nested:
212 {
213     \str_case:en { \@currenvir }
214     {
215         {enumext}
216         {
217             \bool_lazy_and:nnT
218             { \bool_not_p:n { \g__enumext_standar_bool } }
219             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
220             {
221                 \bool_gset_true:N \g__enumext_standar_bool
222             }
223         }
224     }
```

```

223     }
224     {enumext*}
225     {
226         \bool_lazy_and:nnT
227         { \bool_not_p:n { \g__enumext_starred_bool } }
228         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
229         {
230             \bool_gset_true:N \g__enumext_starred_bool
231         }
232     }
233 }
234 }

```

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

```

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

```

(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.

```

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

```

```

282         }
283     }
284     {keyanspic}
285     {
286         \tl_set:Nx \l__enumext_check_start_line_env_tl
287         {
288             in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
289         }
290     }
291 }
292 }

```

(End of definition for `__enumext_keyans_save_start_line:`.)

`__enumext_execute_after_env:`

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

```

293 \cs_new_protected:Nn \__enumext_execute_after_env:
294 {
295     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
296     {
297         \tl_if_empty:NF \g__enumext_store_name_tl
298         {
299             \msg_note:nnV
300             { enumext } { save-ans-hook } \g__enumext_store_name_tl
301             \msg_log:nnVV
302             { enumext } { save-ans-log-hook } \g__enumext_envir_name_tl \g__enumext_store_name_
303             \bool_if:NT \g__enumext_check_ans_bool
304             {
305                 \__enumext_check_ans_show:
306             }
307             % Aquí cargamos los mensajes para .log y terminal
308         }
309         \int_gzero:N \g__enumext_item_number_int
310         \int_gzero:N \g__enumext_item_anskey_int
311         \bool_gset_false:N \g__enumext_check_ans_bool
312         \bool_gset_false:N \g__enumext_standar_bool
313         \bool_gset_false:N \g__enumext_starred_bool
314         \tl_gclear:N \g__enumext_store_name_tl
315         \tl_gclear:N \g__enumext_start_line_tl
316         \tl_gclear:N \g__enumext_envir_name_tl
317     }
318 }

```

(End of definition for `__enumext_execute_after_env:`.)

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.

```

319 \__enumext_at_begin_document:n
320 {
321     \cs_new_eq:NN \__enumext_start_list:nn \list
322     \cs_new_eq:NN \__enumext_stop_list: \endlist
323     \cs_new_eq:NN \__enumext_item_std:w \item
324 }

```

(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:
325 \__enumext_at_begin_document:n
326 {
327     \cs_new_eq:NN \__enumext_minipage:w \minipage
328     \cs_new_eq:NN \__enumext_endminipage: \endminipage
329 }

```

(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.

```

330 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
331 \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 `\l__enumext_footnotes_key_bool` to “true”.

```

332 \cs_new_protected:Nn \__enumext_after_hyperref:
333 {
334     \IfPackageLoadedTF { hyperref }
335     {
336         \msg_info:nnn { enumext } { package-load } { hyperref }
337         \bool_set_true:N \l__enumext_hyperref_bool
338         \IfHyperBoolean{hyperfootnotes}
339         {
340             \typeout{hyperfootnotes=true}
341             \bool_set_true:N \l__enumext_footnotes_key_bool
342         }
343         { \typeout{hyperfootnotes=false} }
344     }
345     { }

```

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`.

```

346 \bool_if:NT \l__enumext_footnotes_key_bool
347 {
348     \IfPackageLoadedTF { footnotehyper }
349     {
350         \msg_info:nnn { enumext } { package-load } { footnotehyper }
351     }
352     {
353         \typeout{No ~ footnotehyper ~ load}
354         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
355         \bool_set_false:N \l__enumext_footnotes_key_bool
356     }
357 }

```

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.

```

358 \bool_if:NTF \l__enumext_hyperref_bool
359 {
360     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
361     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
362 }
363 {
364     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
365     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
366 }
367 }

```

(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: `__enumext_newlabel_arg_one_tl`
 #2: `__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.

```

368 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
369 {
370   \protected@write \@auxout { }
371   {
372     \token_to_str:N \newlabel {#1}
373     {
374       {#2}
375       \bool_if:NT \__enumext_hyperref_bool
376       { { \thepage } {#2} {#1} }
377       { }
378     }
379   }
380   \__enumext_hypertarget:nn {#1} { }
381   \__enumext_phantomsection:
382 }

```

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

10.8 Definition of counters

`__enumext_define_counters:Nn` 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 `__enumext_counter_X_tl` for “store” the counter’s name.
 #2: The counter’s name.

```

383 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
384 {
385   \cs_if_exist:cTF { c@ #2 }
386   { \msg_fatal:nnn { enumext } { counters } { #2 } }
387   {
388     \tl_set:Nn #1 { #2 }
389     \newcounter { #2 }
390   }
391 }

```

(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 392 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii 393 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii 394 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv 395 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv 396 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvi 397 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
enumXvii 398 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii 399 \__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.

```

400 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
401 {
402   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
403   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
404 }

```

```

405 \__enumext_register_counter_style:Nn \arabic { 0 }
406 \__enumext_register_counter_style:Nn \Alph { M }
407 \__enumext_register_counter_style:Nn \alph { m }
408 \__enumext_register_counter_style:Nn \Roman { VIII }
409 \__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.

```

410 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
411 {
412   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
413   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
414 }
415 \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.

```

416 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
417 {
418   \tl_clear_new:N #1
419   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
420   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
421   \tl_map_inline:Nn \g__enumext_counter_styles_tl
422   {
423     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
424     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
425     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
426   }
427   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
428   { \tl_use:N \g__enumext_widest_label_tl }
429   \tl_set_eq:cN { the #2 } #1
430 }
431 \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
labelsep
labelwidth
wrap-label
wrap-label*

```

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

```

432 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
433 {
434   \keys_define:nn { enumext / #1 }
435   {
436     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
437     font      .value_required:n = true,
438     labelsep   .dim_set:c   = { l__enumext_labelsep_#2_dim },
439     labelsep   .initial:n   = {0.3333em},
440     labelsep   .value_required:n = true,
441     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
442     labelwidth .value_required:n = true,
443     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
444     wrap-label .initial:n   = {##1},
445     wrap-label .value_required:n = true,
446     wrap-label* .code:n = {
447       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
448       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
449     },
450     wrap-label* .value_required:n = true,
451   }
452 }
453 \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.

```

454 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
455 {
456   \keys_define:nn { enumext / #1 }
457   {
458     align .choice:,
459     align / left .code:n =
460       {
461         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
462         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
463       },
464     align / right .code:n =
465       {
466         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
467         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
468       },
469     align / center .code:n =
470       {
471         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
472         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
473       },
474     align .initial:n = left,
475     align .value_required:n = true,
476   }
477 }
478 \clist_map_inline:nn
479 {
480   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
481 }
482 { \__enumext_tmp:nn #1 }

483 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
484 {
485   \keys_define:nn { enumext / #1 }
486   {
487     align .choice:,
488     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
489     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
490     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
491     align .initial:n = left,
492     align .value_required:n = true,
493   }
494 }
495 \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

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

496 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
497 {
498   \keys_define:nn { enumext / #1 }
499   {
500     label .code:n = {
501       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
502       { l__enumext_counter_#2_tl } {##1}
503       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
504       \l__enumext_current_widest_dim
505     },
```

```

506     label .initial:n = #3,
507     label .value_required:n = true,
508     ref .code:n = \__enumext_standar_ref:n {##1},
509     ref .value_required:n = true,
510   }
511 }
512 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
513 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
514 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
515 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

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

```

516 \cs_new_protected:Npn \__enumext_standar_ref:n #1
517 {
518   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
519   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
520   {
521     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
522   }
523   {
524     \tl_set_eq:Nc
525       \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
526     \__enumext_regex_counter_style:
527     \tl_set_eq:Nc
528       \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
529     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
530     {
531       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
532       { \exp_not:V \l__enumext_ref_key_arg_tl }
533     }
534   }
535 }

```

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

```

536 \cs_new_protected:Nn \__enumext_standar_ref:
537 {
538   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
539   {
540     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
541   }
542 }

```

(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.

```

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
543 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
544 {
545   \keys_define:nn { enumext / #1 }
546   {
547     label .code:n = {
548       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
549       { \l__enumext_counter_#2_tl } {##1}
550       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
551       \l__enumext_current_widest_dim
552     },
553     label .initial:n = #3,
554     label .value_required:n = true,
555     ref .code:n = \__enumext_starred_ref:n {##1},
556     ref .value_required:n = true,
557   }
558 }

```

```

559 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
560 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

```

\__enumext_starred_ref:n
\__enumext_starred_ref:

```

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

```

561 \cs_new_protected:Npn \__enumext_starred_ref:n #1
562 {
563   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
564   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
565   {
566     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
567     {
568       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
569     }
570     {
571       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
572       \__enumext_regex_counter_style:
573       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
574       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
575       {
576         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
577         { \exp_not:V \l__enumext_ref_key_arg_tl }
578       }
579     }
580   }
581   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
582   {
583     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
584     {
585       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
586     }
587     {
588       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
589       \__enumext_regex_counter_style:
590       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
591       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
592       {
593         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
594         { \exp_not:V \l__enumext_ref_key_arg_tl }
595       }
596     }
597   }
598 }

```

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.

```

599 \cs_new_protected:Nn \__enumext_starred_ref:
600 {
601   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
602   {
603     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
604     {
605       \tl_use:N \l__enumext_renew_the_count_vii_tl
606     }
607   }
608   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
609   {
610     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
611     {
612       \tl_use:N \l__enumext_renew_the_count_viii_tl
613     }
614   }
615 }

```

(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

`label` 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

```

```

616 \keys_define:nn { enumext / keyans }
617 {
618   label .code:n = {
619     \__enumext_label_style:cnv { \__enumext_label_v_tl }
620     { \__enumext_counter_v_tl } {#1}
621     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
622     \__enumext_current_widest_dim
623     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
624     { \__enumext_counter_vi_tl } {#1}
625     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
626     \__enumext_current_widest_dim
627   },
628   label .initial:n = (\Alph*),
629   label .value_required:n = true,
630   ref .code:n = \__enumext_keyans_ref:n {#1},
631   ref .value_required:n = true,
632 }

```

(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`.
`__enumext_keyans_ref:`

```

633 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
634 {
635   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
636   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
637   {
638     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
639   }
640   {
641     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
642     \__enumext_regex_counter_style:
643     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
644     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
645     {
646       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
647       { \exp_not:V \l__enumext_ref_key_arg_tl }
648     }
649   }
650 }

```

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

```

651 \cs_new_protected:Npn \__enumext_keyans_ref:
652 {
653   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
654   {
655     \tl_use:N \l__enumext_renew_the_count_v_tl
656   }
657 }

```

(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.

```

658 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
659 {
660   \__enumext_if_is_int:nTF { #3 }
661   {
662     \int_set:Nn #2 {#3}
663   }
664   {
665     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
666     { \int_set:Nn #2 { \int_from_alph:n {#3} } }

```

```

667         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
668         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
669     }
670 }
671 \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 `\Alpha`, `\alpha`, `\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.

```

672 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
673 {
674     \__enumext_if_is_int:nTF {#4}
675     {
676         \setcounter{enumX#1} { #4 }
677     }
678     {
679         \regex_match:nVT { \c{Alpha} | \c{alpha} } {#2}
680         { \setcounter{enumX#1} { \int_from_alpha:n {#4} } }
681         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
682         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
683     }
684     \__enumext_label_width_by_box:cv
685     { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
686 }
687 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

`start`

`widest`

`\l__enumext_start_X_int`

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

```

688 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
689 {
690     \keys_define:nn { enumext / #1 }
691     {
692         start .code:n = {
693             \__enumext_start_from:ccn
694             { \__enumext_label_#2_tl }
695             { \__enumext_start_#2_int } {##1}
696         },
697         start .initial:n = 1,
698         widest .code:n = {
699             \__enumext_widest_from:nccn {#2}
700             { \__enumext_label_#2_tl }
701             { \__enumext_labelwidth_#2_dim } {##1}
702         },
703         widest .value_required:n = true,
704         start .value_required:n = true,
705     }
706 }
707 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.13 Setting keys for vertical spaces

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

`parsep`
`noitemsep`
`nosep`

```

708 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
709 {
710     \keys_define:nn { enumext / #1 }
711     {
712         topsep .skip_set:c = { \__enumext_topsep_#2_skip },
713         topsep .initial:n = {#3},

```

```

714     topsep      .value_required:n = true,
715     partopsep   .skip_set:c = { l__enumext_partopsep_#2_skip },
716     partopsep   .initial:n = {#4},
717     partopsep   .value_required:n = true,
718     parsep      .skip_set:c = { l__enumext_parsep_#2_skip },
719     parsep      .initial:n = {#5},
720     parsep      .value_required:n = true,
721     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
722     itemsep     .initial:n = {#6},
723     itemsep     .value_required:n = true,
724     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
725     noitemsep   .value_forbidden:n = true,
726     noseop      .meta:n = {
727                             itemsep = 0pt, parsep= 0pt,
728                             topsep = 0pt, partopsep = 0pt,
729                             },
730     noseop      .value_forbidden:n = true,
731   }
732 }

```

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

```

733 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
734 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
735 { 4.0pt plus 2.0pt minus 1.0pt }
736 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
737 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
738 { 2.0pt plus 1.0pt minus 1.0pt }
739 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
740 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
741 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
742 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
743 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
744 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
745 { 2.0pt plus 1.0pt minus 1.0pt }
746 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
747 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
748 { 4.0pt plus 2.0pt minus 1.0pt }
749 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
750 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
751 { 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.

```

752 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
753 {
754   \keys_define:nn { enumext / #1 }
755   {
756     itemindent      .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
757     itemindent      .value_required:n = true,
758     rightmargin     .dim_set:c = { l__enumext_rightmargin_#2_dim },
759     rightmargin     .value_required:n = true,
760     listparindent   .dim_set:c = { l__enumext_listparindent_#2_dim },
761     listparindent   .value_required:n = true,
762     list-offset     .dim_set:c = { l__enumext_listoffset_#2_dim },
763     list-offset     .value_required:n = true,
764     list-indent     .code:n =
765                     \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
766                     \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
767     list-indent     .value_required:n = true,
768   }
769 }
770 \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.

```

771 \cs_set_protected:Npn \__enumext_tmp:n #1
772 {

```

```

773     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
774   }
775   \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.

```

776   \cs_set_protected:Nn \__enumext_fake_item:
777   {
778     \dim_compare:nNnT
779     { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
780     >
781     { \c_zero_dim }
782     {
783       \tl_set:ce { \__enumext_fake_item_indent_ \__enumext_level: _tl }
784       {
785         \exp_not:N \mode_leave_vertical:
786         \exp_not:n { \skip_horizontal:n
787           { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
788         \ignorespaces
789       }
790     }
791   }
792   \cs_set_protected:Nn \__enumext_keyans_fake_item:
793   {
794     \dim_compare:nNnT
795     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
796     {
797       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
798       {
799         \exp_not:N \mode_leave_vertical:
800         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
801       }
802     }
803   }
804   \cs_set_protected:Nn \__enumext_fake_item_vii:
805   {
806     \dim_compare:nNnT
807     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
808     {
809       \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
810       {
811         \exp_not:N \mode_leave_vertical:
812         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
813       }
814     }
815   }
816   \cs_set_protected:Nn \__enumext_fake_item_viii:
817   {
818     \dim_compare:nNnT
819     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
820     {
821       \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
822       {
823         \exp_not:N \mode_leave_vertical:
824         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
825       }
826     }
827   }

```

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

10.15 Setting show-length key

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.

```

828   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2

```



```

829   {
830     \keys_define:nn { enumext / #1 }
831     {
832       show-length .bool_set:c = { l__enumext_show_length_#2_bool },
833       show-length .initial:n = false,
834     }
835   }
836   \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

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

```

before* 837 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    838 {
first    839   \keys_define:nn { enumext / #1 }
          840   {
841     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
842     before .value_required:n = true,
843     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
844     before* .value_required:n = true,
845     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
846     after .value_required:n = true,
847     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
848     first .value_required:n = true,
849   }
850 }
851 \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 $\{\langle code \rangle\}$ set by the `before*` key “before” the `enumext` environment is started. The $\{\langle code \rangle\}$ is executed “without” knowing any definition of the *second argument* of the list.

```

\__enumext_after_stop_list:
\__enumext_after_args_exec:
852 \cs_new_protected:Nn \__enumext_before_args_exec:
853 {
854   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
855 }

```

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

```

856 \cs_new_protected:Nn \__enumext_before_keys_exec:
857 {
858   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
859 }

```

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

```

860 \cs_new_protected:Nn \__enumext_after_stop_list:
861 {
862   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
863 }

```

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

```

864 \cs_new_protected:Nn \__enumext_after_args_exec:
865 {
866   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
867 }

```

(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:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_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.

```

868 \cs_new_protected:Nn __enumext_before_args_exec_v:
869 {
870     \tl_use:N \l__enumext_before_starred_key_v_tl
871 }

```

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⟩`.

```

872 \cs_new_protected:Nn __enumext_before_keys_exec_v:
873 {
874     \tl_use:N \l__enumext_before_no_starred_key_v_tl
875 }

```

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

```

876 \cs_new_protected:Nn __enumext_after_stop_list_v:
877 {
878     \tl_use:N \l__enumext_after_stop_list_v_tl
879 }

```

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`.

```

880 \cs_new_protected:Nn __enumext_after_args_exec_v:
881 {
882     \tl_use:N \l__enumext_after_list_args_v_tl
883 }

```

(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.

```

884 \cs_new_protected:Nn __enumext_before_args_exec_vii:
885 {
886     \tl_use:N \l__enumext_before_starred_key_vii_tl
887 }
888 \cs_new_protected:Nn __enumext_before_args_exec_viii:
889 {
890     \tl_use:N \l__enumext_before_starred_key_viii_tl
891 }

```

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⟩`.

```

892 \cs_new_protected:Nn __enumext_before_keys_exec_vii:
893 {
894     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
895 }
896 \cs_new_protected:Nn __enumext_before_keys_exec_viii:
897 {
898     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
899 }

```

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

```

900 \cs_new_protected:Nn __enumext_after_stop_list_vii:
901 {
902     \tl_use:N \l__enumext_after_stop_list_vii_tl
903 }
904 \cs_new_protected:Nn __enumext_after_stop_list_viii:
905 {
906     \tl_use:N \l__enumext_after_stop_list_viii_tl
907 }

```

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`.

```

908 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
909 {
910   \tl_use:N \l__enumext_after_list_args_vii_tl
911 }
912 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
913 {
914   \tl_use:N \l__enumext_after_list_args_viii_tl
915 }

```

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

10.17 Setting keys for multicols and minipage

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.

```

916 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
917 {
918   \keys_define:nn { enumext / #1 }
919   {
920     mini-env    .dim_set:c = { l__enumext_minipage_right_#2_dim },
921     mini-env    .value_required:n = true,
922     mini-sep    .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
923     mini-sep    .initial:n = 0.3333em,
924     mini-sep    .value_required:n = true,
925     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
926     columns-sep .value_required:n = true,
927     columns     .int_set:c = { l__enumext_columns_#2_int },
928     columns     .initial:n = 1,
929     columns     .value_required:n = true,
930   }
931 }
932 \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`.

```

933 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
934 {
935   \keys_define:nn { enumext / #1 }
936   {
937     columns     .initial:n = 2,
938     miniright   .tl_gset:c = { g__enumext_miniright_code_#2_tl },
939     miniright   .value_required:n = true,
940     miniright*  .code:n = {
941       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
942       \keys_set:nn { enumext / #1 } { miniright = {##1} }
943     },
944     miniright*  .value_required:n = true,
945   }
946 }
947 \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 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 `__enumext_multicols_above_X_skip` and `__enumext_multicols_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

948 \cs_new_protected:Nn \__enumext_multi_set_vskip:
949 {
950   \skip_set:cn { \__enumext_multicols_above_ \__enumext_level: } _skip {
951     {
952       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
953       }
954     }
955   \skip_set:cn { \__enumext_multicols_below_ \__enumext_level: } _skip {
956     {
957       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
958     }
959   }

```

(End of definition for `__enumext_multi_set_vskip:`.)

`__enumext_add_pre_parsep:` The function `__enumext_add_pre_parsep:` “adjusted” the value of `__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*.

```

960 \cs_new_protected:Nn \__enumext_add_pre_parsep:
961 {
962   \int_case:nn { \__enumext_level_int }
963   {
964     { 2 } {
965       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
966       {
967         \skip_add:Nn \__enumext_multicols_above_ii_skip { \__enumext_parsep_i_skip }
968       }
969     }
970     { 3 } {
971       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip }
972       {
973         \skip_add:Nn \__enumext_multicols_above_iii_skip { \__enumext_parsep_ii_skip }
974       }
975     }
976     { 4 } {
977       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip }
978       {
979         \skip_add:Nn \__enumext_multicols_above_iv_skip { \__enumext_parsep_iii_skip }
980       }
981     }
982   }
983 }

```

(End of definition for `__enumext_add_pre_parsep:`.)

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

```

984 \cs_new_protected:Nn \__enumext_multi_addvspace:
985 {
986   \__enumext_multi_set_vskip:
987   \mode_if_vertical:T
988   {

```

```

989     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
990     {
991         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
992     }
993     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
994     {
995         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
996     }
997 }
998 \par\nopagebreak
999 \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1000 }

```

(End of definition for `__enumext_multi_addvspace:`)

10.18.2 Adjustment of vertical spaces for multicols in keyans

`__enumext_keyans_multi_set_vskip:`
`__enumext_keyans_multi_addvspace:`

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

```

1001 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1002 {
1003     \skip_set:Nn \l__enumext_multicols_above_v_skip
1004     {
1005         \l__enumext_topsep_v_skip
1006     }
1007     \skip_set:Nn \l__enumext_multicols_below_v_skip
1008     {
1009         \l__enumext_topsep_v_skip
1010     }
1011 }
1012 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1013 {
1014     \__enumext_keyans_multi_set_vskip:
1015     \mode_if_vertical:T
1016     {
1017         \skip_add:Nn \l__enumext_multicols_above_v_skip
1018         {
1019             \skip_use:N \l__enumext_partopsep_v_skip
1020         }
1021         \skip_add:Nn \l__enumext_multicols_below_v_skip
1022         {
1023             \skip_use:N \l__enumext_partopsep_v_skip
1024         }
1025     }
1026     \par\nopagebreak
1027     \addvspace{ \l__enumext_multicols_above_v_skip }
1028 }

```

(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.

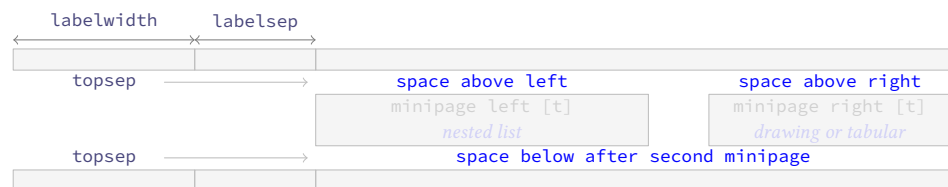


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```

1029 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
1030 {
1031     \__enumext_minipage:w [ t ] { #1 }
1032     \legacy_if_gset_false:n { @minipage }
1033     \vspace { 0pt }
1034 }
1035 { \__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*.

```

1036 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1037 {
1038     \int_compare:nNnTF
1039     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1040     {

```

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.

```

1041     \skip_if_eq:nnTF
1042     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1043     {
1044         \skip_set:Nn \l__enumext_minipage_left_skip
1045         {
1046             -0.15\box_dp:N \strutbox
1047         }
1048         \skip_set:Nn \l__enumext_minipage_right_skip
1049         {
1050             0.695\box_dp:N \strutbox
1051         }
1052         \skip_set:Nn \l__enumext_minipage_after_skip
1053         {
1054             \box_dp:N \strutbox
1055         }
1056         \__enumext_zero_parsep:
1057     }
1058     {
1059         \skip_set:Nn \l__enumext_minipage_left_skip
1060         {
1061             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1062         }
1063         \skip_set:Nn \l__enumext_minipage_right_skip
1064         {
1065             0.695\box_dp:N \strutbox
1066         }
1067         \skip_set:Nn \l__enumext_minipage_after_skip
1068         {
1069             1.85\box_dp:N \strutbox
1070             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1071         }
1072     }

```

```

1073     }
1074   {

```

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.

```

1075     \skip_if_eq:nnTF
1076     { \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip } } { \c_zero_skip }
1077     {
1078       \skip_set:Nn \l__enumext_minipage_left_skip
1079       {
1080         0.5\box_dp:N \strutbox
1081         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1082       }
1083       \skip_set:Nn \l__enumext_minipage_right_skip
1084       {
1085         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1086       }
1087       \skip_set:Nn \l__enumext_minipage_after_skip
1088       {
1089         1.6\box_dp:N \strutbox
1090       }
1091     }
1092   {
1093     \skip_set:Nn \l__enumext_minipage_left_skip
1094     {
1095       0.5875\box_dp:N \strutbox
1096       - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1097     }
1098     \skip_set:Nn \l__enumext_minipage_right_skip
1099     {
1100       + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1101       + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1102     }
1103     \skip_set:Nn \l__enumext_minipage_after_skip
1104     {
1105       0.325\box_dp:N \strutbox
1106       + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1107     }
1108   }
1109 }
1110 }

```

(End of definition for `__enumext_mini_set_vskip:`)

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

```

1111 \cs_new_protected:Nn \__enumext_zero_parsep:
1112 {
1113   \int_case:nn { \l__enumext_level_int }
1114   {
1115     { 2 }{
1116       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1117       {
1118         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1119       }
1120     }
1121     { 3 }{
1122       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1123       {
1124         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1125       }
1126     }
1127     { 4 }{
1128       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1129       {
1130         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1131       }
1132     }
1133   }
1134 }

```


(End of definition for `__enumext_zero_parsep:`)

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

```

1135 \cs_new_protected:Nn \__enumext_mini_addvspace:
1136 {
1137   \__enumext_mini_set_vskip:
1138   \mode_if_vertical:T
1139   {
1140     \skip_add:Nn \l__enumext_minipage_left_skip
1141     {
1142       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1143     }
1144     \skip_add:Nn \l__enumext_minipage_after_skip
1145     {
1146       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1147     }
1148   }
1149   \par\nopagebreak
1150   \addvspace { \l__enumext_minipage_left_skip }
1151 }

```

(End of definition for `__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`.

```

1152 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1153 {
1154   \skip_zero_new:N \l__enumext_minipage_after_skip
1155   \skip_zero_new:N \l__enumext_minipage_left_skip
1156   \skip_zero_new:N \l__enumext_minipage_right_skip
1157   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1158   {
1159     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1160     {
1161       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1162       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1163       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1164       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1165       {
1166         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1167       }
1168     }
1169     {
1170       \skip_set:Nn \l__enumext_minipage_left_skip
1171       {
1172         \skip_use:N \l__enumext_topsep_v_skip
1173       }
1174       \skip_set:Nn \l__enumext_minipage_right_skip
1175       {
1176         0.705\box_dp:N \strutbox
1177       }
1178       \skip_set:Nn \l__enumext_minipage_after_skip
1179       {
1180         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1181       }
1182     }
1183   }
1184   {
1185     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1186     {
1187       \skip_set:Nn \l__enumext_minipage_left_skip
1188       {
1189         0.5\box_dp:N \strutbox
1190         + \l__enumext_partopsep_v_skip
1191       }

```

```

1192         \skip_set:Nn \l__enumext_minipage_right_skip
1193         {
1194             \l__enumext_partopsep_v_skip
1195         }
1196     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1197 }
1198 {
1199     \skip_set:Nn \l__enumext_minipage_left_skip
1200     {
1201         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1202     }
1203     \skip_set:Nn \l__enumext_minipage_right_skip
1204     {
1205         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1206     }
1207     \skip_set:Nn \l__enumext_minipage_after_skip
1208     {
1209         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1210     }
1211 }
1212 }
1213 }

```

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

```

1214 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1215 {
1216     \__enumext_keyans_mini_set_vskip:
1217     \mode_if_vertical:T
1218     {
1219         \skip_add:Nn \l__enumext_minipage_left_skip
1220         {
1221             \l__enumext_partopsep_v_skip
1222         }
1223         \skip_add:Nn \l__enumext_minipage_after_skip
1224         {
1225             \l__enumext_partopsep_v_skip
1226         }
1227     }
1228     \par\nopagebreak
1229     \addvspace { \l__enumext_minipage_left_skip }
1230 }

```

(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*`.

```

1231 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1232 {
1233     \skip_zero_new:N \l__enumext_minipage_left_skip
1234     \skip_gzero_new:N \g__enumext_minipage_right_skip
1235     \skip_gzero_new:N \g__enumext_minipage_after_skip
1236     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1237     {
1238         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1239         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1240     }
1241     {
1242         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1243         \skip_gset:Nn \g__enumext_minipage_right_skip
1244         {
1245             \l__enumext_topsep_vii_skip
1246         }
1247     }
1248 }

```

```

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

```

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

`\l__enumext_mini_addvspace_vii:`
`\l__enumext_mini_addvspace_viii:`

The functions `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `\l__enumext_mini_env*` environment on the *left side* when the `\miniright` key is active in the `enumext*` and `keyans*` environments.

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

```

1288 \cs_new_protected:Nn \l__enumext_mini_addvspace_vii:
1289 {
1290     \l__enumext_mini_set_vskip_vii:
1291     \par\nopagebreak
1292     \addvspace { \l__enumext_minipage_left_skip }
1293 }
1294 \cs_new_protected:Nn \l__enumext_mini_addvspace_viii:
1295 {
1296     \l__enumext_mini_set_vskip_viii:
1297     \par\nopagebreak
1298     \addvspace { \l__enumext_minipage_left_skip }
1299 }

```

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

10.19.4 The command `\miniright`

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

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

```

1300 \NewDocumentCommand \miniright { s }
1301 {
1302   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1303   {
1304     \msg_error:nnn { enumext } { wrong-miniright-place }
1305   }
1306   \int_compare:nNt { \__enumext_level_int } = { 0 }
1307   {
1308     \msg_error:nnn { enumext } { wrong-miniright-place }
1309   }
1310   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1311   {
1312     \__enumext_keyans_mini_right_cmd:n {#1}
1313   }
1314   { \__enumext_mini_right_cmd:n {#1} }
1315 }

```

(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 `__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.

```

1316 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1317 {
1318   \dim_compare:nNtF
1319   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1320   {
1321     \__enumext_multicols_stop:
1322     \end{\__enumext_mini_env*}
1323     \hfill
1324     \begin{\__enumext_mini_env*}
1325     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1326     \par\addvspace { \__enumext_minipage_right_skip }
1327     \bool_if:nF {#1}
1328     {
1329       \centering
1330     }
1331     \int_gzero:N \g__enumext_minipage_stat_int
1332   }
1333   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1334 }

```

(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.

```

1335 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1336 {
1337   \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1338   {
1339     \__enumext_keyans_multicols_stop:
1340     \end{\__enumext_mini_env*}
1341     \hfill
1342     \begin{\__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1343     \par\addvspace { \__enumext_minipage_right_skip }
1344     \bool_if:nF {#1}
1345     {
1346       \centering
1347     }
1348     \int_gzero:N \g__enumext_minipage_stat_int
1349   }
1350   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1351 }

```

(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.

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

```

1352 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1353 {
1354   \keys_define:nn { enumext / #1 }
1355   {
1356     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1357     above .value_required:n = true,
1358     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1359               \keys_set:nn { enumext / #1 } { above = {##1} },
1360     above* .value_required:n = true,
1361     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1362     below .value_required:n = true,
1363     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1364               \keys_set:nn { enumext / #1 } { below = {##1} },
1365     below* .value_required:n = true,
1366   }
1367 }
1368 \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.

```

1369 \cs_new_protected:Nn \__enumext_vspace_above:
1370 {
1371   \skip_if_eq:nnF
1372   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1373   {
1374     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1375     {
1376       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1377     }
1378     {
1379       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1380     }
1381   }
1382 }

```

(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.

```

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

```

(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.

```

1397 \cs_new_protected:Nn \__enumext_vspace_above_v:
1398 {
1399   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1400   {
1401     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1402     {
1403       \vspace*{ \l__enumext_vspace_above_v_skip }
1404     }
1405     { \vspace { \l__enumext_vspace_above_v_skip } }
1406   }
1407 }

```

(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.

```

1408 \cs_new_protected:Nn \__enumext_vspace_below_v:
1409 {
1410   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1411   {
1412     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1413     {
1414       \vspace*{ \l__enumext_vspace_below_v_skip }
1415     }
1416     { \vspace { \l__enumext_vspace_below_v_skip } }
1417   }
1418 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

`__enumext_vspace_above_vii:`

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

`__enumext_vspace_above_viii:`

```

1419 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1420 {
1421   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1422   {
1423     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1424     {
1425       \vspace*{ \l__enumext_vspace_above_vii_skip }
1426     }
1427     { \vspace { \l__enumext_vspace_above_vii_skip } }
1428   }
1429 }
1430 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1431 {
1432   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1433   {
1434     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1435     {
1436       \vspace*{ \l__enumext_vspace_above_viii_skip }
1437     }
1438     { \vspace { \l__enumext_vspace_above_viii_skip } }
1439   }
1440 }

```

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

`__enumext_vspace_below_vii:`

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

`__enumext_vspace_below_viii:`

```

1441 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1442 {
1443   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1444   {
1445     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1446     {
1447       \vspace*{ \l__enumext_vspace_below_vii_skip }

```

```

1448     }
1449     { \vspace { \l__enumext_vspace_below_vii_skip } }
1450   }
1451 }
1452 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1453 {
1454   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1455   {
1456     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1457     {
1458       \vspace*{ \l__enumext_vspace_below_viii_skip }
1459     }
1460     { \vspace { \l__enumext_vspace_below_viii_skip } }
1461   }
1462 }

```

(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*
1463 \cs_set_protected:Npn \__enumext_tmp:n #1
1464 {
1465   \keys_define:nn { enumext / #1 }
1466   {
1467     series .str_set:N = \l__enumext_series_str,
1468     series .value_required:n = true,
1469     resume .code:n = \__enumext_resume_series:n {##1},
1470     resume* .code:n = \__enumext_resume_starred:,
1471     resume* .value_forbidden:n = true,
1472   }
1473 }
1474 \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

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

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1475 \cs_new:Npn \__enumext_filter_series:n #1
1476 {
1477   \use:e
1478   {
1479     \keyval_parse:NNn
1480     \__enumext_filter_series_key:n
1481     \__enumext_filter_series_pair:nn {#1}
1482   }
1483 }

```

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

```

1484 \cs_new:Npn \__enumext_filter_series_key:n #1
1485 {
1486   \str_case:nnF {#1}
1487   {
1488     { resume } {}
1489     { resume* } {}
1490   }
1491   { , { \exp_not:n {#1} } }
1492 }

```

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

```

1493 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1494 {
1495   \str_case:nnF {#1}
1496   {
1497     { series } {}

```



```

1498     { resume } {}
1499     { start } {}
1500     { save-ans } {}
1501     { save-key } {}
1502   }
1503   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1504 }

```

(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).

```

1505 \cs_new_protected:Npn \__enumext_parse_series:n #1
1506 {
1507   \str_if_empty:NTF \l__enumext_series_str
1508   {
1509     \bool_if:NF \l__enumext_resume_active_bool
1510     {
1511       \__enumext_resume_last:n {#1}
1512     }
1513   }
1514   {
1515     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1516     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1517       { \__enumext_filter_series:n {#1} }
1518     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1519     {
1520       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1521     }
1522   }
1523 }

```

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.

```

1524 \cs_new_protected:Npn \__enumext_resume_last:n #1
1525 {
1526   \bool_if:NT \l__enumext_standar_first_level_bool
1527   {
1528     \tl_gclear:N \g__enumext_standar_series_tl
1529     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1530   }
1531   \bool_if:NT \l__enumext_starred_first_level_bool
1532   {
1533     \tl_gclear:N \g__enumext_starred_series_tl
1534     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1535   }
1536 }

```

(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).

```

1537 \cs_new_protected:Nn \__enumext_resume_save_counter:
1538 {
1539   \bool_if:NT \__enumext_standar_bool
1540   {
1541     \tl_if_empty:NF \__enumext_series_str
1542     {
1543       \int_gset_eq:cN
1544       { g__enumext_series_ \__enumext_series_str_int } \value{enumXi}
1545     }
1546     \tl_if_empty:NTF \__enumext_resume_name_tl
1547     {
1548       \str_if_empty:NT \__enumext_series_str
1549       {
1550         \int_gset_eq:NN \__enumext_resume_int \value{enumXi}
1551       }
1552     }
1553     {
1554       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1555       {
1556         \int_gset_eq:cN
1557         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXi}
1558       }
1559     }
1560     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1561     {
1562       \int_gset_eq:cN
1563       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXi}
1564     }
1565   }
1566   \bool_if:NT \__enumext_starred_bool
1567   {
1568     \tl_if_empty:NF \__enumext_series_str
1569     {
1570       \int_gset_eq:cN
1571       { g__enumext_series_ \__enumext_series_str_int } \value{enumXvii}
1572     }
1573     \tl_if_empty:NTF \__enumext_resume_name_tl
1574     {
1575       \str_if_empty:NT \__enumext_series_str
1576       {
1577         \int_gset_eq:NN \__enumext_resume_vii_int \value{enumXvii}
1578       }
1579     }
1580     {
1581       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1582       {
1583         \int_gset_eq:cN
1584         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXvii}
1585       }
1586     }
1587     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1588     {
1589       \int_gset_eq:cN
1590       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXvii}
1591     }
1592   }
1593 }

```

(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.

```

1594 \cs_new_protected:Npn \__enumext_resume_series:n #1
1595 {
1596   \tl_if_empty:NTF {#1}
1597   {
1598     \__enumext_resume_counter:n { }
1599   }
1600   {
1601     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1602     {
1603       \__enumext_resume_counter:n {#1}
1604       \bool_if:NT \g__enumext_standar_bool
1605       {
1606         \keys_set:nv { enumext / level-1 }
1607         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1608       }
1609       \bool_if:NT \g__enumext_starred_bool
1610       {
1611         \keys_set:nv { enumext / enumext* }
1612         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1613       }
1614     }
1615     {
1616       \bool_if:NT \g__enumext_standar_bool
1617       {
1618         \msg_error:nnn { enumext } { unknown-series } {#1}
1619       }
1620       \bool_if:NT \g__enumext_starred_bool
1621       {
1622         \msg_error:nnn { enumext } { unknown-series } {#1}
1623       }
1624     }
1625   }
1626 }

```

(End of definition for `__enumext_resume_series:n`)

```

\__enumext_resume_counter:n
\__enumext_resume_counter:
  \__enumext_resume_counter_series:
  \__enumext_resume_counter_save_ans:

```

The function `__enumext_resume_counter:n` will set the variable `\l__enumext_resume_active_bool` to true and pass the value of the key `resume` to the variable `\l__enumext_series_name_tl` which will contain the `{⟨series name⟩}`. If the variable `\l__enumext_series_name_tl` is empty, that is, we are passing the key `resume` *without value*, we will execute the function `__enumext_resume_counter:` otherwise, when we pass `resume={⟨series name⟩}` we will execute the function `__enumext_resume_counter_series:`, finally we will execute the function `__enumext_resume_counter_save_ans:` which is associated with the key `save-ans`.

```

1627 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1628 {
1629   \bool_set_true:N \l__enumext_resume_active_bool
1630   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1631   \tl_if_empty:NTF \l__enumext_resume_name_tl
1632   {
1633     \__enumext_resume_counter:
1634   }
1635   {
1636     \__enumext_resume_counter_series:
1637   }
1638   \__enumext_resume_counter_save_ans:
1639 }

```

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.

```

1640 \cs_new_protected:Nn \__enumext_resume_counter:
1641 {
1642   \bool_if:NT \g__enumext_standar_bool
1643   {
1644     \int_gincr:N \g__enumext_resume_int
1645     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1646   }
1647   \bool_if:NT \g__enumext_starred_bool
1648   {
1649     \int_gincr:N \g__enumext_resume_vii_int

```

```

1650         \int_set_eq:Nn \l__enumext_start_vii_int \g__enumext_resume_vii_int
1651     }
1652 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume={⟨series name⟩}` key is active, setting the counters for the “first level” of the environments according to the value of the integer variables created by the `series` key.

```

1653 \cs_new_protected:Nn \__enumext_resume_counter_series:
1654 {
1655     \bool_if:NT \g__enumext_standar_bool
1656     {
1657         \int_set:Nn \l__enumext_start_i_int
1658         {
1659             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1660         }
1661     }
1662     \bool_if:NT \g__enumext_starred_bool
1663     {
1664         \int_set:Nn \l__enumext_start_vii_int
1665         {
1666             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1667         }
1668     }
1669 }

```

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.

```

1670 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1671 {
1672     \bool_lazy_and:nnT
1673     { \bool_if_p:N \l__enumext_standar_first_level_bool }
1674     { \bool_if_p:N \l__enumext_store_active_bool }
1675     {
1676         \int_set:Nn \l__enumext_start_i_int
1677         {
1678             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1679         }
1680     }
1681     \bool_lazy_and:nnT
1682     { \bool_if_p:N \l__enumext_starred_first_level_bool }
1683     { \bool_if_p:N \l__enumext_store_active_bool }
1684     {
1685         \int_set:Nn \l__enumext_start_vii_int
1686         {
1687             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1688         }
1689     }
1690 }

```

(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.

```

1691 \cs_new_protected:Nn \__enumext_resume_starred:
1692 {
1693     \bool_if:NT \g__enumext_standar_bool
1694     {
1695         \tl_if_empty:NF \g__enumext_standar_series_tl
1696         {
1697             \__enumext_resume_counter:n { }
1698             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1699         }
1700     }
1701     \bool_if:NT \g__enumext_starred_bool
1702     {
1703         \tl_if_empty:NF \g__enumext_starred_series_tl
1704         {

```

```

1705         \__enumext_resume_counter:n { }
1706         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1707     }
1708 }
1709 }

```

(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*`.

```

1710 \cs_set_protected:Npn \__enumext_tmp:n #1
1711 {
1712     \keys_define:nn { enumext / #1 }
1713     {
1714         save-ans .code:n = \__enumext_storing_set:n {##1},
1715         save-ans .value_required:n = true,
1716     }
1717 }
1718 \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.

```

1719 \cs_new_protected:Npn \__enumext_storing_set:n #1
1720 {
1721     \tl_set:Nx \l__enumext_store_name_tl {#1}
1722     \tl_if_empty:NTF \l__enumext_store_name_tl
1723     {
1724         \bool_lazy_or:nnT
1725         { \l__enumext_standar_first_level_bool }
1726         { \l__enumext_starred_first_level_bool }
1727         {
1728             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1729         }
1730     }
1731     {
1732         \bool_lazy_or:nnT
1733         { \l__enumext_standar_first_level_bool }
1734         { \l__enumext_starred_first_level_bool }
1735         {
1736             \msg_note:nnV
1737             { enumext } { save-ans-ok } \l__enumext_store_name_tl
1738             \msg_log:nnVV
1739             { enumext } { save-ans-log } \g__enumext_envir_name_tl \l__enumext_store_name_tl
1740             \__enumext_storing_exec:
1741         }
1742     }
1743 }

```

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*`.

```

1744 \cs_new_protected:Nn \__enumext_storing_exec:
1745 {
1746     \bool_set_true:N \l__enumext_store_active_bool
1747     \bool_set_true:N \l__enumext_store_ans_bool
1748     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1749     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1750     {

```

```

1751     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1752     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1753   }
1754   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1755   {
1756     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1757     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1758   }
1759   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1760   {
1761     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1762     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1763   }
1764 }

```

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

10.23 The check answer mechanism

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

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

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

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

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

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

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

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

10.23.1 Setting check-ans key

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

```

check-ans no-store
1765 \cs_set_protected:Npn \__enumext_tmp:n #1
1766 {
1767   \keys_define:nn { enumext / #1 }
1768   {
1769     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1770     check-ans .initial:n = false,
1771     check-ans .value_required:n = true,
1772     no-store .code:n = {
1773       \bool_set_false:N \l__enumext_store_ans_bool
1774       \bool_set_false:N \l__enumext_check_ans_bool
1775     },
1776     no-store .value_forbidden:n = true,
1777   }
1778 }
1779 \clist_map_inline:nn
1780 {
1781   level-1, level-2, level-3, level-4, enumext*
1782 }
1783 { \__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.

```

1784 \cs_new_protected:Nn \__enumext_check_ans:
1785 {
1786   \bool_if:NT \l__enumext_check_ans_bool
1787   {
1788     \tl_if_empty:NTF \l__enumext_store_name_tl
1789     {
1790       \bool_if:NT \l__enumext_standar_first_level_bool
1791       {
1792         \msg_error:nnnn { enumext } { need-save-ans } { check-ans } { enumext }
1793       }
1794       \bool_if:NT \l__enumext_starred_first_level_bool
1795       {
1796         \msg_error:nnnn { enumext } { need-save-ans } { check-ans } { enumext* }
1797       }
1798     }
1799     {
1800       \__enumext_check_ans_level:
1801     }
1802   }
1803 }

```

The function `__enumext_check_ans_level:` will decrement by “*one*” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment `enumext`, taking into account whether it is nested within `enumext*` or the opposite.

```

1804 \cs_new_protected:Nn \__enumext_check_ans_level:
1805 {
1806   \int_case:nn { \l__enumext_level_int }
1807   {
1808     { 1 } {
1809       \bool_lazy_all:nT
1810       {
1811         { \bool_if_p:N \g__enumext_starred_bool }
1812         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1813       }
1814       {
1815         \int_gdecr:N \g__enumext_item_number_int
1816       }
1817     }
1818     { 2 } {
1819       \int_gdecr:N \g__enumext_item_number_int
1820     }
1821     { 3 } {
1822       \int_gdecr:N \g__enumext_item_number_int
1823     }
1824     { 4 } {
1825       \int_gdecr:N \g__enumext_item_number_int
1826     }
1827   }
1828   \int_case:nn { \l__enumext_level_h_int }
1829   {
1830     { 1 } {
1831       \bool_if:NT \g__enumext_standar_bool
1832       {
1833         \int_gdecr:N \g__enumext_item_number_int
1834       }
1835     }
1836   }
1837 }

```

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

`__enumext_check_ans_to_hook:`

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


```

1838 \cs_new_protected:Nn \__enumext_check_ans_to_hook:
1839 {
1840   \bool_lazy_and:nnT
1841     { \bool_if_p:N \l__enumext_check_ans_bool }
1842     { \bool_if_p:N \g__enumext_standar_bool }
1843     {
1844       \bool_gset_true:N \g__enumext_check_ans_bool
1845     }
1846   \bool_lazy_and:nnT
1847     { \bool_if_p:N \l__enumext_check_ans_bool }
1848     { \bool_if_p:N \g__enumext_starred_bool }
1849     {
1850       \bool_gset_true:N \g__enumext_check_ans_bool
1851     }
1852 }

```

(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.

```

1853 \cs_new_protected:Nn \__enumext_check_ans_show:
1854 {
1855   \int_compare:nNnTF
1856     { \g__enumext_item_number_int } = { \g__enumext_item_anskey_int }
1857     {
1858       \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1859     }
1860     {
1861       \msg_warning:nnVV
1862         { enumext } { item-different-answer }
1863       \g__enumext_store_name_tl \g__enumext_start_line_tl
1864     }
1865 }

```

(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.

```

1866 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1867 {
1868   \int_compare:nNnT
1869     { \g__enumext_check_starred_cmd_int } = { 0 }
1870     {
1871       \msg_warning:nnnV
1872         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1873     }
1874   \int_compare:nNnT
1875     { \g__enumext_check_starred_cmd_int } > { 1 }
1876     {
1877       \msg_warning:nnnV
1878         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1879     }
1880   \int_gzero:N \g__enumext_check_starred_cmd_int
1881   \tl_clear:N \l__enumext_check_start_line_env_tl
1882 }

```

(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*`.

```

1883 \cs_set_protected:Npn \__enumext_tmp:n #1
1884 {
1885   \keys_define:nn { enumext / #1 }
1886   {

```

```

1887     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1888     wrap-ans .initial:n = \fbox{##1},
1889     wrap-ans .value_required:n = true,
1890     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1891     wrap-opt .initial:n = [{##1}],
1892     wrap-opt .value_required:n = true,
1893     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1894     save-sep .initial:n = {, ~ },
1895     save-sep .value_required:n = true,
1896     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
1897     mark-ans .initial:n = \textasteriskcentered,
1898     mark-ans .value_required:n = true,
1899     mark-pos .choice:,
1900     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1901     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1902     mark-pos .initial:n = right,
1903     mark-pos .value_required:n = true,
1904     show-ans .bool_set:N = \__enumext_show_answer_bool,
1905     show-ans .initial:n = false,
1906     show-ans .value_required:n = true,
1907     show-pos .bool_set:N = \__enumext_show_position_bool,
1908     show-pos .initial:n = false,
1909     show-pos .value_required:n = true,
1910     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
1911     mark-ref .initial:n = \textasteriskcentered,
1912     mark-ref .value_required:n = true,
1913     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
1914     save-ref .initial:n = false,
1915     save-ref .value_required:n = true,
1916   }
1917 }
1918 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1919 \cs_set_protected:Npn \__enumext_tmp:n #1
1920 {
1921   \keys_define:nn { enumext / #1 }
1922   {
1923     mark-pos .choice:,
1924     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1925     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1926     mark-pos .initial:n = right,
1927     mark-pos .value_required:n = true,
1928     show-ans .bool_set:N = \__enumext_show_answer_bool,
1929     show-ans .initial:n = false,
1930     show-ans .value_required:n = true,
1931     show-pos .bool_set:N = \__enumext_show_position_bool,
1932     show-pos .initial:n = false,
1933     show-pos .value_required:n = true,
1934   }
1935 }
1936 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1937 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1938 {
1939   \keys_define:nn { enumext / #1 }
1940   {
1941     columns* .code:n = \bool_set_true:c { \__enumext_store_columns_#2_bool }
1942     \int_set:cn { \__enumext_store_columns_#2_int } {##1}
1943     \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1944     {
1945       columns = \exp_not:v { \__enumext_store_columns_#2_int },

```

```

1946         },
1947         columns* .value_required:n = true,
1948         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1949         \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1950         \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1951         {
1952             columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim }
1953         },
1954         columns-sep* .value_required:n = true,
1955     }
1956 }
1957 \clist_map_inline:nn
1958 {
1959     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1960 }
1961 { \__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.

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.

```

1962 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1963 {
1964     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1965     {
1966         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1967     }
1968     { #1 }
1969 }
1970 \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.

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

```

1971 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1972 {
1973     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1974 }
1975 \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 `opt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```

1976 \cs_new_protected:Nn \__enumext_store_level_open:
1977 {
1978     \bool_if:NT \l__enumext_store_ans_bool
1979     {
1980         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1981         {
1982             \__enumext_store_addto_seq:n
1983             {
1984                 \item \begin{enumext}
1985             }

```

```

1986     }
1987     {
1988         \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1989         {
1990             \item \begin{enumext} [
1991             ]
1992             \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1993             {
1994             ]
1995             }
1996             \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1997         }
1998     }
1999 }
2000 \cs_new_protected:Nn \__enumext_store_level_close:
2001 {
2002     \bool_if:NT \l__enumext_store_ans_bool
2003     {
2004         \__enumext_store_addto_seq:n { \end{enumext} }
2005     }
2006 }

```

(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.

```

2007 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2008 {
2009     \bool_if:NT \l__enumext_store_ans_bool
2010     {
2011         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
2012         {
2013             \__enumext_store_addto_seq:n
2014             {
2015                 \item \mode_leave_vertical:
2016                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2017                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2018             }
2019         }
2020         {
2021             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
2022             {
2023                 \item \mode_leave_vertical:
2024                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2025                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2026                 ]
2027             }
2028             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
2029             {
2030             ]
2031             }
2032             \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
2033         }
2034     }
2035 }
2036 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2037 {
2038     \bool_if:NT \l__enumext_store_ans_bool
2039     {
2040         \__enumext_store_addto_seq:n { \end{enumext*} }
2041     }
2042 }

```

(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

2042 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2043 {
2044   \mode_leave_vertical:
2045   \skip_horizontal:n { -\dim_use:N #2 }
2046   \makebox[0pt][ r ]
2047   {
2048     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2049     {
2050       \tl_use:N \l__enumext_mark_answer_sym_tl
2051     }
2052   }
2053   \skip_horizontal:n { \dim_use:N #2 }
2054 }
2055 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for `__enumext_print_keyans_box:NN`.)

10.25 The command `\anskey` and internal label and ref

Since we will be “*storing content*” in a list environment within *⟨sequences⟩* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing. The `\anskey` command will cover this point and give it very similar behaviour to that of `\item` in the `enumext` and `enumext*` environments.

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

```

2056 \keys_define:nn { enumext / anskey }
2057 {
2058   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2059   item-sym* .value_required:n = true,
2060   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2061   item-pos* .value_required:n = true,
2062   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2063   store-brk .default:n = true,
2064   store-brk .value_forbidden:n = true,
2065 }

```

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

```

2066 \NewDocumentCommand \anskey { d() s o +m }
2067 {
2068   \bool_if:NF \l__enumext_store_active_bool
2069   {
2070     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2071   }
2072   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2073   {
2074     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2075   }
2076   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2077   {
2078     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2079   }
2080   \group_begin:
2081     \bool_if:NT \l__enumext_store_ans_bool
2082     {
2083       \bool_if:NT \l__enumext_check_ans_bool
2084       {
2085         \int_gincr:N \g__enumext_item_anskey_int
2086       }
2087       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2088     }
2089   \group_end:
2090 }

```

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

```
\__enumext_store_anskey_code:nnnn
```

The internal function `__enumext_store_anskey_code:nnnn` first we pass the command *⟨argument⟩* to the *⟨prop list⟩*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `save-ref` key and will call the function `__enumext_store_internal_ref:` for the internal “*label and ref*” system. Followed by this if the `show-ans` or `show-pos` keys are active we will show the “*wrapped*” *⟨argument⟩* passed to the command.

```
2091 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2092 {
2093   \__enumext_store_addto_prop:n {#4}
2094   \bool_if:NT \l__enumext_store_ref_key_bool
2095   {
2096     \__enumext_store_internal_ref:
2097   }
2098   \__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⟩*.

```
2099   \tl_clear:N \l__enumext_store_anskey_arg_tl
2100   \tl_if_novalue:nF {#3}
2101   {
2102     \keys_set:nn { enumext / anskey } {#3}
2103   }
2104   \bool_lazy_and:nnT
2105   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2106   { \bool_not_p:n { \l__enumext_starred_bool } }
2107   {
2108     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2109   }
2110   \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).

```
2111   \tl_if_novalue:nF {#1}
2112   {
2113     \int_set:Nn \l__enumext_store_columns_join_int {#1}
2114     \bool_if:NT \l__enumext_starred_bool
2115     {
2116       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2117       {
2118         ( \exp_not:V \l__enumext_store_columns_join_int )
2119       }
2120     }
2121   }
```

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`.

```
2122   \bool_if:nTF {#2}
2123   {
2124     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2125     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2126     {
2127       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2128       {
2129         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2130       }
2131     }
2132     \dim_compare:nT
2133     {
2134       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2135     }
2136     {
2137       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2138       {
2139         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2140       }
2141     }
2142     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2143   }
2144 }
```

```

2145     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2146   }

```

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`.

```

2147   \bool_lazy_and:nnT
2148   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2149   { \bool_if_p:N \l__enumext_hyperref_bool }
2150   {
2151     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2152     {
2153       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2154       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2155     }
2156   }
2157   \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2158 }

```

(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`.

```

2159 \cs_new_protected:Nn \__enumext_store_internal_ref:
2160 {
2161   \cs_set_protected:Npn \__enumext_tmp:n ##1
2162   {
2163     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2164     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2165     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2166     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2167   }
2168   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2169   \cs_set:Npn \__enumext_tmp:n ##1
2170   { . \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.

```

2171   \bool_lazy_all:nT
2172   {
2173     { \bool_if_p:N \g__enumext_starred_bool }
2174     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2175   }
2176   {
2177     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2178     { \tl_use:N \l__enumext_label_copy_vii_tl }
2179   }
2180   \bool_lazy_all:nT
2181   {
2182     { \bool_if_p:N \l__enumext_standar_bool }
2183     { \bool_if_p:N \g__enumext_starred_bool }
2184     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2185   }
2186   {
2187     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2188     {
2189       \tl_use:N \l__enumext_label_copy_vii_tl
2190       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2191     }
2192   }

```

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.

```

2193   \bool_lazy_all:nT
2194   {
2195     { \bool_if_p:N \l__enumext_standar_bool }
2196     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2197     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2198     { \bool_not_p:n { \l__enumext_starred_bool } }
2199   }

```



```

2200     {
2201         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2202         {
2203             \tl_use:N \l__enumext_label_copy_i_tl
2204             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2205         }
2206     }
2207 \cs_set:Npn \l__enumext_tmp:n ##1
2208 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2209 \bool_lazy_all:nT
2210 {
2211     { \bool_if_p:N \l__enumext_standar_bool }
2212     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2213     { \bool_not_p:n { \g__enumext_starred_bool } }
2214     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2215 }
2216 {
2217     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2218     {
2219         \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2220         . \tl_use:N \l__enumext_label_copy_vii_tl
2221     }
2222 }

```

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

```

2223 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2224 {
2225     \l__enumext_store_name_tl \c_colon_str
2226     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2227 }

```

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.

```

2228 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2229 {
2230     \l__enumext_newlabel:nn
2231     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2232     { \l__enumext_newlabel_arg_two_tl }
2233 }
2234 \l__enumext_store_write_aux_file_tl
2235 }

```

(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.

```

2236 \cs_new_protected:Npn \l__enumext_store_anskey_show_wrap:n #1
2237 {
2238     \par
2239     \bool_if:NT \l__enumext_starred_bool
2240     {
2241         \cs_set:Nn \l__enumext_level: { vii }
2242     }
2243     \l__enumext_print_keyans_box:cc
2244     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2245     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2246     \l__enumext_anskey_wrapper:n { #1 }
2247 }

```

(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.

```

2248 \cs_new_protected:Npn \l__enumext_store_anskey_show_left:n #1
2249 {
2250     \bool_if:NT \l__enumext_show_answer_bool
2251     {
2252         \l__enumext_store_anskey_show_wrap:n { #1 }

```

```

2253     }
2254     \bool_if:NT \l__enumext_show_position_bool
2255     {
2256         \tl_set:Nx \l__enumext_mark_answer_sym_tl
2257         {
2258             \group_begin:
2259             \exp_not:N \normalfont
2260             \exp_not:N \footnotesize [ \int_eval:n
2261             {
2262                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2263             }
2264             ]
2265             \group_end:
2266         }
2267         \__enumext_store_anskey_show_wrap:n { #1 }
2268     }
2269 }

```

(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`.

```

2270 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2271 {
2272     \tl_clear:N \l__enumext_store_keyans_label_tl
2273     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2274     {
2275         \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2276     }
2277     {
2278         \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2279     }
2280     \tl_if_novalue:nF { #1 }
2281     {
2282         % Set save-sep
2283         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2284         {
2285             \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2286         }
2287         \tl_put_right:Nx \l__enumext_store_keyans_label_tl { #1 }
2288     }
2289     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2290 }

```

(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.

```

2291 \cs_new_protected:Npn \__enumext_keyans_store_ref:
2292 {
2293     \bool_if:NT \l__enumext_store_ref_key_bool
2294     {
2295         \cs_set_protected:Npn \__enumext_tmp:n #1
2296         {
2297             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2298             \tl_reverse:c { \l__enumext_label_copy_##1_tl }

```

```

2299         \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2300         \tl_reverse:c { \__enumext_label_copy_##1_tl }
2301     }
2302     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2303     \__enumext_keyans_store_ref_aux_i:
2304 }
2305 }

```

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

```

2306 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2307 {
2308     \bool_if:NT \g__enumext_starred_bool
2309     {
2310         \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2311     }
2312     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2313     {
2314         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2315         { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2316     }
2317     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2318     {
2319         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2320         { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2321     }
2322     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2323     {
2324         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2325         { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
2326     }
2327     \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
2328     {
2329         \__enumext_store_name_tl \c_colon_str
2330         \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
2331     }
2332     \__enumext_keyans_store_ref_aux_ii:
2333 }

```

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

```

2334 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2335 {
2336     \tl_put_right:Ne \__enumext_store_write_aux_file_tl
2337     {
2338         \__enumext_newlabel:nn
2339         { \exp_not:V \__enumext_newlabel_arg_one_tl }
2340         { \__enumext_newlabel_arg_two_tl }
2341     }
2342     \__enumext_store_write_aux_file_tl
2343 }

```

(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 $\langle \textit{label} \rangle$ `__enumext_label_v_tl` for the `keyans` environment and the `__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the $\langle \textit{contents} \rangle$ of the optional argument of both commands to the `__enumext_store_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

2344 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2345 {
2346     \tl_clear:N \__enumext_store_keyans_label_tl
2347     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2348     {
2349         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_vi_tl }
2350     }
2351     {
2352         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_v_tl }

```

```

2353     }
2354     \tl_if_novalue:nF { #1 }
2355     {
2356         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2357         {
2358             \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2359             {
2360                 \l__enumext_store_keyans_item_opt_sep_tl
2361             }
2362         }
2363         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2364     }
2365     \__enumext_keyans_addto_seq_link:
2366 }

```

Checks if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store using the `__enumext_store_addto_seq:V` function. Finally, copy the contents of the variable `\l__enumext_store_keyans_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `__enumext_check_starred_cmd:n` and increment the value of the integer variable `\g__enumext_item_anskey_int` handled by the `check-ans` key.

```

2367 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2368 {
2369     \bool_lazy_and:nnT
2370     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2371     { \bool_if_p:N \l__enumext_hyperref_bool }
2372     {
2373         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2374         {
2375             \hfill \exp_not:N \hyperlink
2376             {
2377                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2378             }
2379             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2380         }
2381     }
2382     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2383     \bool_if:NT \l__enumext_check_ans_bool
2384     {
2385         \int_gincr:N \g__enumext_item_anskey_int
2386     }
2387 }

```

(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.

```

2388 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2389 {
2390     \tl_if_novalue:nF { #1 }
2391     {
2392         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2393     }
2394     \bool_if:NT \l__enumext_show_answer_bool
2395     {
2396         \__enumext_keyans_show_ans:
2397     }
2398     \bool_if:NT \l__enumext_show_position_bool
2399     {
2400         \__enumext_keyans_show_pos:
2401     }
2402 }
2403 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2404 {
2405     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2406     {
2407         \bool_lazy_or:nnT

```

```

2408         { \bool_if_p:N \l__enumext_show_answer_bool }
2409         { \bool_if_p:N \l__enumext_show_position_bool }
2410         {
2411             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2412         }
2413     }
2414 }
2415 \cs_new_protected:Nn \l__enumext_keyans_show_ans:
2416 {
2417     \tl_put_left:Nn \l__enumext_label_v_tl
2418     {
2419         \__enumext_print_keyans_box:NN
2420         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2421     }
2422 }
2423 \cs_new_protected:Nn \l__enumext_keyans_show_pos:
2424 {
2425     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2426     {
2427         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2428         {
2429             \group_begin:
2430             \exp_not:N \normalfont
2431             \exp_not:N \footnotesize [ \int_eval:n
2432             {
2433                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2434             }
2435             ]
2436             \group_end:
2437         }
2438     }
2439     {
2440         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2441         {
2442             \group_begin:
2443             \exp_not:N \normalfont
2444             \exp_not:N \footnotesize [ \int_eval:n
2445             {
2446                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2447             }
2448             ]
2449             \group_end:
2450         }
2451     }
2452     \tl_put_left:Nn \l__enumext_label_v_tl
2453     {
2454         \__enumext_print_keyans_box:NN
2455         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2456     }
2457 }

```

(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*
2458 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
2459 {
2460     \keys_define:nn { enumext / #1 }
2461     {
2462         item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2463         item-sym* .value_required:n = true,
2464         item-sym* .initial:n = { $\star$ },
2465         item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2466         item-pos* .value_required:n = true,
2467     }
2468 }
2469 \clist_map_inline:nn
2470 {

```

```

2471     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2472   }
2473   { \__enumext_tmp:n #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](#).

```

2474 \cs_new_protected:Nn \__enumext_footnotetext:nn
2475 {
2476   \footnotetext[#1]{#2}
2477 }
2478 \cs_new_protected:Nn \__enumext_renew_footnote:
2479 {
2480   \seq_gclear:N \g__enumext_footnote_arg_seq
2481   \seq_gclear:N \g__enumext_footnote_int_seq
2482   \RenewDocumentCommand \footnote { o +m }
2483   {
2484     \tl_if_novalue:nTF {##1}
2485     {
2486       \stepcounter{footnote}
2487       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2488     }
2489     {
2490       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2491     }
2492     \footnotemark [ \g__enumext_footnote_int ]
2493     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2494     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2495   }
2496 }
2497 \cs_new_protected:Nn \__enumext_print_footnote:
2498 {
2499   \seq_if_empty:NF \g__enumext_footnote_int_seq
2500   {
2501     \seq_map_pairwise_function:NNN
2502     \g__enumext_footnote_int_seq
2503     \g__enumext_footnote_arg_seq
2504     \__enumext_footnotetext:nn
2505   }
2506 }

```

(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).

```

2507 \cs_new_protected:Npn \__enumext_default_item:n #1
2508 {
2509   \tl_if_novalue:nTF {#1}
2510   {
2511     \bool_if:NT \l__enumext_check_ans_bool
2512     {
2513       \int_gincr:N \g__enumext_item_number_int
2514     }
2515     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }

```

```

2516     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2517   }
2518   {
2519     \bool_set_eq:cc
2520     { l__enumext_wrap_label_ \__enumext_level: _bool }
2521     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2522     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2523   }
2524 }

```

(End of definition for __enumext_default_item:n.)

__enumext_starred_item:nn

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

#1: \l__enumext_item_symbol_X_tl

#2: \l__enumext_item_symbol_sep_X_dim

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label:` (§10.30).

```

2525 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2526 {
2527   \tl_if_novalue:nF {#1}
2528   {
2529     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2530   }
2531   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2532   \tl_if_novalue:nTF {#2}
2533   {
2534     \dim_set_eq:cc
2535     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2536     { l__enumext_labelsep_ \__enumext_level: _dim }
2537   }
2538   {
2539     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2540   }
2541   \bool_if:NT \l__enumext_check_ans_bool
2542   {
2543     \int_gincr:N \g__enumext_item_number_int
2544   }
2545   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2546   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2547 }

```

(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).

```

2548 \cs_new_protected:Nn \__enumext_redefine_item:
2549 {
2550   \RenewDocumentCommand \item { s o o }
2551   {
2552     \bool_if:nTF {##1}
2553     {
2554       \__enumext_starred_item:nn {##2} {##3}
2555     }
2556   }

```



```

2556         { \__enumext_default_item:n {##2} }
2557     }
2558 }

```

(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`.

```

2559 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2560 {
2561     \tl_if_novalue:nTF { #1 }
2562     {
2563         \bool_set_true:N \__enumext_wrap_label_v_bool
2564         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2565     }
2566     {
2567         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2568         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2569     }
2570 }

```

(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 $\langle label \rangle$.

```

2571 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2572 {
2573     \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2574     \__enumext_keyans_show_left:n { #1 }
2575     \bool_set_true:N \__enumext_wrap_label_v_bool
2576     \__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 $\langle sequence \rangle$.

```

2577     \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2578     \__enumext_keyans_addto_prop:n { #1 }
2579     \__enumext_keyans_store_ref:
2580     \__enumext_keyans_addto_seq:n { #1 }
2581     \int_gincr:N \g__enumext_check_starred_cmd_int
2582 }

```

(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).

```

2583 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2584 {
2585     \RenewDocumentCommand \item { s o }
2586     {
2587         \bool_if:nTF {##1}
2588         {
2589             \peek_remove_spaces:n
2590             {
2591                 \__enumext_keyans_starred_item:n {##2}
2592             }
2593         }
2594         {
2595             \__enumext_keyans_default_item:n {##2}
2596         }
2597     }
2598 }

```

(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.

```

2599 \cs_new_protected:Nn \__enumext_item_starred:
2600 {
2601   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2602   {
2603     \mode_leave_vertical:
2604     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2605     \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
2606     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2607   }
2608 }
```

(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).

```

2609 \cs_new_protected:Nn \__enumext_make_label:
2610 {
2611   \RenewDocumentCommand \makeLabel { m }
2612   {
2613     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2614     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2615     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2616     {
2617       \__enumext_item_starred:
2618       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2619     }
2620     { ##1 }
2621     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2622     \tl_gclear:N \g__enumext_item_symbol_tl
2623   }
2624 }
```

(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).

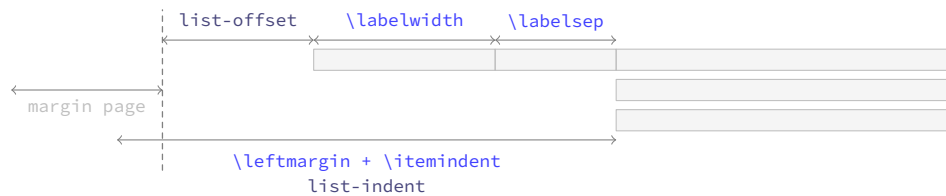
```

2625 \cs_new_protected:Nn \__enumext_keyans_make_label:
2626 {
2627   \RenewDocumentCommand \makeLabel { m }
2628   {
2629     \tl_use:N \l__enumext_label_fill_left_v_tl
2630     \tl_use:N \l__enumext_label_font_style_v_tl
2631     \bool_if:NTF \l__enumext_wrap_label_v_bool
2632     {
2633       \__enumext_wrapper_label_v:n { ##1 }
2634     }
2635     { ##1 }
2636     \tl_use:N \l__enumext_label_fill_right_v_tl
2637   }
2638 }
```

(End of definition for __enumext_keyans_make_label:.)

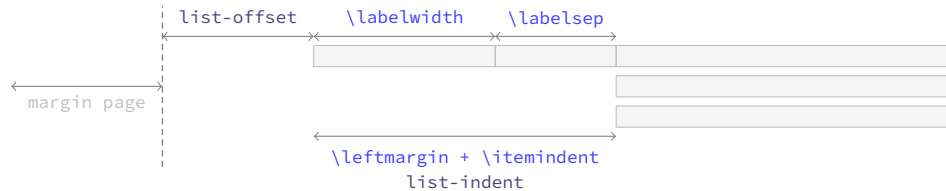
10.31 Second argument of the lists

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.

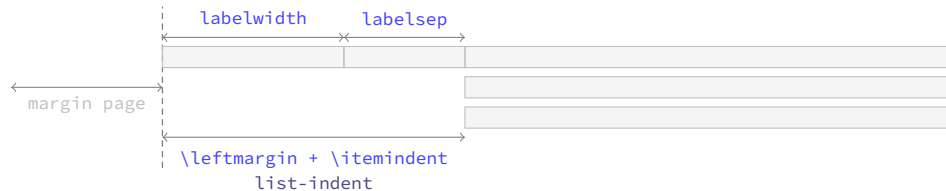
Figure 9: Representation of standard horizontal lengths in `list` environment.

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).

```
2639 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2640 {
2641   \dim_compare:nNt { #1 } < { \c_zero_dim }
2642   {
2643     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2644     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2645   }
2646   \dim_compare:nNt { #2 } < { \c_zero_dim }
2647   {
2648     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2649     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2650   }
2651 }
```

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

```
2651   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2652   \dim_compare:nNtF { #4 } < { \c_zero_dim }
2653   {
2654     \dim_set:Nn #6 { #1 + #2 - #4 }
2655     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2656   }
```

```

2657     {
2658         \dim_compare:nNnT { #4 } = { #1 + #2 }
2659         { \dim_set:Nn #6 { \c_zero_dim } }
2660         \dim_compare:nNnT { #4 } < { #1 + #2 }
2661         { \dim_set:Nn #6 { #1 + #2 - #4 } }
2662         \dim_compare:nNnT { #4 } > { #1 + #2 }
2663         {
2664             \dim_set:Nn #6 { -#1 - #2 + #4 }
2665             \dim_set:Nn #6 { #6*-1 }
2666         }
2667         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2668     }
2669 }
2670 \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:
2671 \cs_set_protected:Npn \__enumext_tmp:n #1
2672 {
2673     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2674     {
2675         \__enumext_calc_hspace:ccccc
2676         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2677         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2678         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2679         { \__enumext_leftmargin_tmp_#1_bool }
2680         \clist_map_inline:nn
2681         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2682         { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2683         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2684         { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2685         \usecounter { enumX#1 }
2686         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2687         \str_if_eq:nnTF {#1} { v }
2688         {
2689             \__enumext_keyans_redefine_item:
2690             \__enumext_keyans_make_label:
2691             \__enumext_keyans_ref:
2692             \__enumext_keyans_fake_item:
2693             \bool_if:cT { \__enumext_show_length_#1_bool }
2694             {
2695                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2696             }
2697         }
2698         {
2699             \__enumext_redefine_item:
2700             \__enumext_make_label:
2701             \__enumext_standar_ref:
2702             \__enumext_fake_item:
2703             \bool_if:cT { \__enumext_show_length_#1_bool }
2704             {
2705                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2706             }
2707         }
2708     }
2709 }
2710 \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.

```

2711 \cs_set_protected:Npn \__enumext_tmp:n #1
2712 {
2713     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2714     {

```

```

2715 \__enumext_calc_hspace:ccccc
2716 { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2717 { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2718 { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2719 { \__enumext_leftmargin_tmp_#1_bool }
2720 \clist_map_inline:nn
2721 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2722 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2723 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2724 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2725 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2726 \skip_zero:N \partopsep
2727 \usecounter { enumX#1 }
2728 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2729 \__enumext_starred_ref:
2730 \str_if_eq:nnTF {#1} { vii }
2731 {
2732   \__enumext_fake_item_vii:
2733   \bool_if:cT { \__enumext_show_length_vii_bool }
2734     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2735 }
2736 {
2737   \__enumext_fake_item_viii:
2738   \bool_if:cT { \__enumext_show_length_#1_bool }
2739     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2740 }
2741 }
2742 }
2743 \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.

```

2744 \NewDocumentEnvironment{enumext}{0} {
2745   {
2746     \__enumext_safe_exec:
2747     \__enumext_parse_keys:n {#1}
2748     \__enumext_before_list:
2749     \__enumext_start_store_level:
2750     \__enumext_start_list:nn
2751     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2752     {
2753       \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2754       \__enumext_before_keys_exec:
2755     }
2756     \__enumext_after_args_exec:
2757   }
2758   {
2759     \__enumext_stop_list:
2760     \__enumext_stop_store_level:
2761     \__enumext_after_list:
2762   }

```

(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:`.

```

2763 \cs_new_protected:Nn \__enumext_safe_exec:
2764 {
2765   \__enumext_is_not_nested:
2766   \int_incr:N \l__enumext_level_int
2767   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2768     { \msg_fatal:nn { enumext } { list-too-deep } }
2769   \bool_set_true:N \l__enumext_standar_bool
2770   \__enumext_is_on_first_level:
2771 }

```

(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`.

```

2772 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2773 {
2774   \tl_if_novalue:nF {#1}
2775   {
2776     \str_clear:N \l__enumext_series_str
2777     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2778     {
2779       \keys_set:nn { enumext / level-1 } {#1}
2780       \__enumext_parse_series:n {#1}
2781     }
2782     {
2783       \exp_args:Ne \keys_set:nn
2784         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2785     }
2786     \bool_if:NT \l__enumext_store_active_bool
2787     {
2788       \__enumext_parse_store_keys:n {#1}
2789     }
2790   }
2791 }

```

(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:`.

```

2792 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2793 {
2794   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2795   {
2796     \regex_match:nnT { \b columns\b } {#1}
2797     {
2798       \int_set_eq:cc
2799         { \l__enumext_store_columns_ \__enumext_level: _int }
2800         { \l__enumext_columns_ \__enumext_level: _int }
2801       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2802         {
2803           columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2804         }
2805     }
2806   }
2807   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2808   {
2809     \regex_match:nnT { \b columns-sep\b } {#1}
2810     {
2811       \dim_set_eq:cc
2812         { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2813         { \l__enumext_columns_sep_ \__enumext_level: _dim }
2814       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2815         {
2816           columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2817         }
2818     }
2819   }
2820 }

```

(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.

```

2821 \cs_new_protected:Nn \__enumext_start_store_level:
2822 {
2823   \bool_lazy_all:nT
2824   {
2825     { \bool_if_p:N \l__enumext_store_active_bool }
2826     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2827     { \bool_not_p:n { \g__enumext_starred_bool } }
2828   }
2829   {
2830     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2831     {
2832       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2833       \__enumext_store_level_open:
2834     }
2835   }
2836   \bool_lazy_all:nT
2837   {
2838     { \bool_if_p:N \l__enumext_store_active_bool }
2839     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2840     { \bool_if_p:N \g__enumext_starred_bool }
2841   }
2842   {
2843     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2844     {
2845       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2846       \__enumext_store_level_open:
2847     }
2848   }
2849 }
2850 \cs_new_protected:Nn \__enumext_stop_store_level:
2851 {
2852   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2853   {
2854     \__enumext_store_level_close:
2855   }
2856 }

```

(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.

```

2857 \cs_new_protected:Nn \__enumext_before_list:
2858 {
2859   \__enumext_vspace_above:
2860   \__enumext_before_args_exec:

```

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

```

2861 \__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.

```

2862 \dim_compare:nNnT
2863 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2864 {
2865   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2866   {
2867     \linewidth
2868     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2869     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2870   }

```

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*.

```

2871         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2872         \int_gincr:N \g__enumext_minipage_stat_int
2873         \__enumext_mini_addvspace:
2874         \nointerlineskip\noindent
2875         \begin{__enumext_mini_env*}
2876         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2877     }
2878     \__enumext_multicols_start:
2879 }

```

(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.

```

2880 \cs_new_protected:Nn \__enumext_multicols_start:
2881 {
2882     \int_compare:nNt
2883     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2884     {
2885         \dim_compare:nNt
2886         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2887         {
2888             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2889             {
2890                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2891                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2892                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2893                   - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2894             }
2895         }
2896         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2897         \skip_zero:N \multicolsep
2898         \int_compare:nNt { \l__enumext_level_int } > { 1 }
2899         {
2900             \dim_zero:N \columnseprule
2901         }
2902     }
2903 }

```

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.

```

2902         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2903         {
2904             \__enumext_multi_addvspace:
2905         }
2906         \raggedcolumns
2907         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2908     }
2909 }

```

(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.

```

2910 \cs_new_protected:Nn \__enumext_multicols_stop:
2911 {
2912     \int_compare:nNt
2913     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2914     {
2915         \end{multicols}
2916         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2917         {
2918             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2919         }
2920     }
2921 }

```

```

2920     }
2921 }

```

(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.

```

2922 \cs_new_protected:Nn \__enumext_after_list:
2923 {
2924   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2925   {
2926     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2927     {
2928       \msg_warning:nn { enumext } { missing-miniright }
2929       \miniright
2930     }
2931     \int_gzero:N \g__enumext_minipage_stat_int
2932     \end{__enumext_mini_env*}
2933     \par\addvspace { \l__enumext_minipage_after_skip }
2934   }
2935   { \__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`.

```

2936   \__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.

```

2937   \__enumext_after_stop_list:
2938   \__enumext_vspace_below:
2939   \bool_set_false:N \l__enumext_standar_bool
2940   \__enumext_resume_save_counter:
2941 }

```

(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`.

```

2942 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

10.33 The environment `keyans`

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

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

```

2943 \NewDocumentEnvironment{keyans}{ 0{ } }
2944 {
2945   \__enumext_keyans_safe_exec:
2946   \__enumext_keyans_parse_keys:n {#1}
2947   \__enumext_before_list_v:
2948   \__enumext_start_list:nn
2949   { \tl_use:N \l__enumext_label_v_tl }
2950   {
2951     \__enumext_list_arg_two_v:
2952     \__enumext_before_keys_exec_v:
2953   }
2954   \__enumext_after_args_exec_v:
2955 }
2956 {
2957   \__enumext_check_starred_cmd:n { item }
2958   \__enumext_stop_list:
2959   \__enumext_after_list_v:
2960 }

```

(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.

```

2961 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2962 {
2963   \bool_if:NF \l__enumext_store_active_bool
2964   {
2965     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
2966   }
2967   \int_incr:N \l__enumext_keyans_level_int
2968   \bool_set_true:N \l__enumext_keyans_env_bool
2969   \__enumext_keyans_save_start_line:
2970   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2971   \bool_set_false:N \l__enumext_store_active_bool
2972   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2973   {
2974     \msg_error:nn { enumext } { keyans-nested }
2975   }
2976   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2977   {
2978     \msg_error:nn { enumext } { keyans-wrong-level }
2979   }
2980 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

2981 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2982 {
2983   \keys_set:nn { enumext / keyans } { #1 }
2984 }

```

(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.

```

2985 \cs_new_protected:Nn \__enumext_before_list_v:
2986 {
2987   \__enumext_vspace_above_v:
2988   \__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.

```

2989   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2990   {
2991     \dim_set:Nn \l__enumext_minipage_left_v_dim
2992     {
2993       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2994     }

```

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*.

```

2995     \bool_set_true:N \l__enumext_minipage_active_v_bool
2996     \int_gincr:N \g__enumext_minipage_stat_int
2997     \__enumext_keyans_mini_addvspace:
2998     \nointerlineskip\noindent
2999     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3000   }

```

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

```

3001   \__enumext_keyans_multicols_start:
3002 }

```

(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.

```

3003 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3004 {
3005   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3006   {
3007     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3008     {
3009       \dim_set:Nn \__enumext_columns_sep_v_dim
3010       {
3011         (
3012           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3013         ) / \__enumext_columns_v_int
3014         - \__enumext_listoffset_v_dim
3015       }
3016     }
3017     \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).

```

3018     \skip_zero:N \multicolsep
3019     \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.

```

3020     \bool_if:NF \__enumext_minipage_active_v_bool
3021     {
3022       \__enumext_keyans_multi_addvspace:
3023     }
3024     \raggedcolumns
3025     \begin{multicols}{ \__enumext_columns_v_int }
3026   }
3027 }

```

(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.

```

3028 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3029 {
3030   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3031   {
3032     \end{multicols}
3033     \bool_if:NF \__enumext_minipage_active_v_bool
3034     {
3035       \par\addvspace{ \__enumext_multicols_below_v_skip }
3036     }
3037   }
3038 }

```

(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.

```

3039 \cs_new_protected:Nn \__enumext_after_list_v:
3040 {
3041   \bool_if:NTF \__enumext_minipage_active_v_bool
3042   {
3043     \int_compare:nNt { \__enumext_minipage_stat_int } = { 1 }
3044     {
3045       \msg_warning:nn { enumext } { missing-miniright }
3046       \miniright

```

```

3047     }
3048     \int_gzero:N \g__enumext_minipage_stat_int
3049     \end{__enumext_mini_env*}
3050     \par\addvspace{ \l__enumext_minipage_after_skip }
3051   }
3052   { __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.

```

3053     \bool_set_false:N \l__enumext_keyans_env_bool
3054     __enumext_after_stop_list_v:
3055     __enumext_vspace_below_v:
3056   }

```

(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.

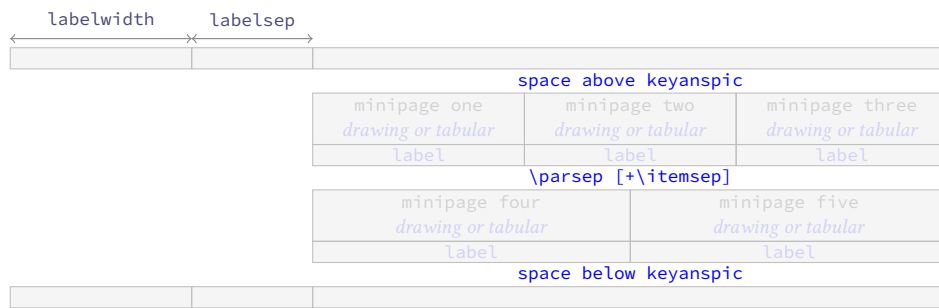


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

```

3057 \NewDocumentCommand \anspic { s o +m }
3058 {

```

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

```

3059     \bool_if:NF \l__enumext_store_active_bool
3060     {
3061       \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3062     }
3063     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
3064     {
3065       \msg_error:nn { enumext } { keyanspic-wrong-level }
3066     }
3067     \int_compare:nNtT { \l__enumext_keyans_level_int } = { 1 }
3068     {
3069       \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3070     }

```

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.

```

3071     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3072     {
3073       __enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3074     }
3075   }

```

(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.

```

3076 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3077 {
3078   \stepcounter { enumXvi }
3079   #3 \\\
3080   \bool_if:nT { #1 }
3081   {
3082     \__enumext_keyans_addto_prop:n { #2 }
3083     \__enumext_keyans_store_ref:
3084     \__enumext_keyans_addto_seq:n { #2 }
3085     \int_gincr:N \g__enumext_check_starred_cmd_int
3086     \bool_lazy_or:nnT
3087     { \bool_if_p:N \l__enumext_show_answer_bool }
3088     { \bool_if_p:N \l__enumext_show_position_bool }
3089     {
3090       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3091       \__enumext_keyans_show_left:n { #2 }
3092       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3093     }
3094   }
3095   \tl_use:N \l__enumext_label_font_style_v_tl
3096   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3097 }

```

(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.

```

3098 \NewDocumentEnvironment{keyanspic}{o}
3099 {
3100   \__enumext_keyans_pic_safe_exec:
3101   \__enumext_start_list:nn
3102   { }
3103   {
3104     \__enumext_keyans_pic_arg_two:
3105   }

```

We apply the “adjusted” vertical spacing above the environment

```

3106   \vspace { \l__enumext_keyans_pic_above_skip }
3107 }

```

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.

```

3108 {
3109   \tl_if_novalue:nTF { #1 }
3110   {
3111     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3112   }
3113   { \__enumext_keyans_pic_do:n { #1 } }
3114   \__enumext_stop_list:
3115   \__enumext_check_starred_cmd:n { anspic }
3116   \setcounter { enumXvi } { 0 }
3117   \vspace { \l__enumext_topsep_v_skip }
3118   %\bool_set_false:N \l__enumext_store_active_bool
3119 }

```

(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.

```

3120 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3121 {
3122   \int_incr:N \l__enumext_keyans_pic_level_int
3123   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3124   {
3125     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3126     }
3127     \__enumext_keyans_save_start_line:
3128 }

```

(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.

```

3129 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3130 {
3131     \dim_compare:nNnT { #1 } < { 0pt }
3132     { \skip_set:Nn #1 { -#1 } }
3133 }

```

(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.

```

3134 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3135 {

```

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.

```

3136     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3137     \__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.

```

3138     \skip_add:Nn \parsep { \itemsep }
3139     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3140     \dim_zero:N \labelwidth
3141     \dim_zero:N \listparindent
3142     \dim_zero:N \labelsep
3143     \skip_zero:N \partopsep
3144     \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.

```

3145     \__enumext_keyans_pic_skip_abs:N \parsep
3146     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3147     {
3148         \box_dp:N \strutbox
3149         + \l__enumext_topsep_v_skip
3150         - \parsep
3151     }
3152     \__enumext_item_std:w \scan_stop:
3153 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

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

```

3154 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3155 {
3156     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3157 }
3158 \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.

```

3159 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3160 {
3161     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3162     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3163     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }

```



```

3164 \int_step_inline:nnn
3165 { \l__enumext_keyans_pic_above_int + 1 }
3166 { \l__enumext_keyans_pic_below_int }
3167 {
3168   \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3169   \centering
3170   \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3171   \__enumext_endminipage:
3172 }
3173 \par
3174 }

```

(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

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

```

3175 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3176 {
3177   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3178   {
3179     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3180     {
3181       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3182       / \l__enumext_columns_vii_int
3183     }
3184   }
3185   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3186   \dim_set:Nn \l__enumext_item_width_vii_dim
3187   {
3188     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3189     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3190     - \l__enumext_labelsep_vii_dim
3191   }
3192   \dim_zero_new:N \itemwidth
3193 }

```

(End of definition for `__enumext_starred_columns_set_vii:`.)

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`.

```

3194 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3195 {
3196   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3197   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3198   {
3199     \msg_warning:nnee { enumext } { item-joined }
3200     { \int_use:N \l__enumext_joined_item_vii_int }
3201     { \int_use:N \l__enumext_columns_vii_int }
3202     \int_set:Nn \l__enumext_joined_item_vii_int
3203     {
3204       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3205     }
3206   }
3207   \int_compare:nNnT
3208   { \l__enumext_joined_item_vii_int }
3209   >
3210   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```

```

3211 {
3212   \msg_warning:nnee { enumext } { item-joined-columns }
3213   { \int_use:N \l__enumext_joined_item_vii_int }
3214   {
3215     \int_eval:n
3216     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3217   }
3218   \int_set:Nn \l__enumext_joined_item_vii_int
3219   {
3220     \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3221   }
3222 }

```

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

```

3223   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3224   {
3225     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3226     \int_decr:N \l__enumext_joined_item_aux_vii_int
3227     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3228     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3229     \dim_set:Nn \l__enumext_joined_width_vii_dim
3230     {
3231       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3232       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3233         + \l__enumext_columns_sep_vii_dim
3234         ) * \l__enumext_joined_item_aux_vii_int
3235     }
3236     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3237   }
3238   {
3239     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3240     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3241   }
3242 }

```

(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`.

```

3243 \cs_new_protected:Nn \__enumext_start_mini_vii:
3244 {
3245   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3246   {
3247     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3248     {
3249       \linewidth
3250       - \l__enumext_minipage_right_vii_dim
3251       - \l__enumext_minipage_hsep_vii_dim
3252     }
3253     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3254     \dim_gset_eq:NN
3255     \g__enumext_minipage_right_vii_dim
3256     \l__enumext_minipage_right_vii_dim
3257     \__enumext_mini_addvspace_vii:
3258     \nointerlineskip\noindent
3259     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3260   }
3261 }

```

(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”.

```

3262 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3263 {
3264   \bool_if:NT \l__enumext_minipage_active_vii_bool

```

```

3265     {
3266         \end{__enumext_mini_env*}
3267         \hfill
3268         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3269     }
3270 }

```

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”.

```

3271 \__enumext_after_env:nn {enumext*}
3272 {
3273     \bool_if:NT \g__enumext_minipage_active_vii_bool
3274     {
3275         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3276         \par\addvspace { \g__enumext_minipage_right_skip }
3277         \bool_if:NF \g__enumext_minipage_center_vii_bool
3278         {
3279             \centering
3280         }
3281         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3282         \end{__enumext_mini_env*}
3283         \par\addvspace{ \g__enumext_minipage_after_skip }
3284     }
3285     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3286     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3287     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3288     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3289     \bool_gset_false:N \g__enumext_starred_bool
3290 }

```

(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.

```

3291 \NewDocumentEnvironment{enumext*}{ o }
3292 {
3293     \__enumext_safe_exec_vii:
3294     \__enumext_parse_keys_vii:n {#1}
3295     \__enumext_before_list_vii:
3296     \__enumext_start_store_level_vii:
3297     \__enumext_start_list:nn { }
3298     {
3299         \__enumext_list_arg_two_vii:
3300         \__enumext_before_keys_exec_vii:
3301     }
3302     \__enumext_starred_columns_set_vii:
3303     \item[] \scan_stop:
3304     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3305     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3306 }
3307 {
3308     \__enumext_stop_item_tmp_vii:
3309     \__enumext_remove_extra_parsep_vii:
3310     \__enumext_stop_list:
3311     \__enumext_stop_store_level_vii:
3312     \__enumext_after_list_vii:
3313 }

```

(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`.

```

3314 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3315 {
3316     \__enumext_is_not_nested:
3317     \int_incr:N \l__enumext_level_h_int
3318     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3319     {
3320         \msg_error:nn { enumext } { nested }
3321     }

```

```

3322     \bool_set_true:N \l__enumext_starred_bool
3323     \__enumext_is_on_first_level:
3324 }

```

(End of definition for __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n

Parse [*key* = *val*] for *enumext**. If the variable \l__enumext_store_active_bool is true it will call the function __enumext_parse_store_keys_vii:n and reprocess the keys to pass them to the storage sequence.

```

3325 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3326 {
3327     \tl_if_novalue:nF {#1}
3328     {
3329         \str_clear:N \l__enumext_series_str
3330         \keys_set:nn { enumext / enumext* } {#1}
3331         \__enumext_parse_series:n {#1}
3332         \bool_if:NT \l__enumext_store_active_bool
3333         {
3334             \__enumext_parse_store_keys_vii:n {#1}
3335         }
3336     }
3337 }

```

(End of definition for __enumext_parse_keys_vii:n.)

__enumext_parse_store_keys_vii:n

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

```

3338 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3339 {
3340     \bool_if:NF \l__enumext_store_columns_vii_bool
3341     {
3342         \regex_match:nnT { \b columns\b } {#1}
3343         {
3344             \int_set_eq:NN
3345             \l__enumext_store_columns_vii_int
3346             \l__enumext_columns_vii_int
3347             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3348             {
3349                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3350             }
3351         }
3352     }
3353     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3354     {
3355         \regex_match:nnT { \b columns-sep\b } {#1}
3356         {
3357             \dim_set_eq:NN
3358             \l__enumext_store_columns_sep_vii_dim
3359             \l__enumext_columns_sep_vii_dim
3360             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3361             {
3362                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3363             }
3364         }
3365     }
3366 }

```

(End of definition for __enumext_parse_store_keys_vii:n.)

__enumext_before_list_vii:

The function __enumext_before_list_vii: will add the vertical spacing on the environment if the *above* key is active next to the {*code*} defined by the *before** key if it is active, the call the function __enumext_start_mini_vii: handle by *mini-env*.

```

3367 \cs_new_protected:Npn \__enumext_before_list_vii:
3368 {
3369     \__enumext_vspace_above_vii:
3370     \__enumext_check_ans: % need by chek-ans
3371     \__enumext_before_args_exec_vii:
3372     \__enumext_start_mini_vii:
3373 }

```

(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.

```

3374 \cs_new_protected:Nn \__enumext_after_list_vii:
3375 {
3376   \__enumext_stop_mini_vii:
3377   \__enumext_after_stop_list_vii:
3378   \__enumext_check_ans_to_hook:
3379   \__enumext_vspace_below_vii:
3380   \bool_set_false:N \l__enumext_starred_bool
3381   \__enumext_resume_save_counter:
3382 }

```

(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:`

```

3383 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3384 {
3385   \bool_if:NT \l__enumext_store_active_bool
3386   {
3387     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3388     {
3389       \__enumext_store_level_open_vii:
3390     }
3391   }
3392 }
3393 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3394 {
3395   \bool_if:NT \l__enumext_store_active_bool
3396   {
3397     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3398     {
3399       \__enumext_store_level_close_vii:
3400     }
3401   }
3402 }

```

(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`.

```

3403 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3404 {
3405   \__enumext_stop_item_tmp_vii:
3406   \int_incr:N \l__enumext_item_column_pos_vii_int
3407   \int_gincr:N \g__enumext_item_count_all_vii_int
3408   \__enumext_item_peek_args_vii:
3409 }

```

(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).

```

3410 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3411 {
3412   \peek_meaning:NTF (
3413     { \__enumext_joined_item_vii:w }

```

```

3414     { \__enumext_joined_item_vii:w (1) }
3415 }

```

(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.

```

3416 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3417 {
3418   \__enumext_starred_joined_item_vii:n {#1}
3419   \peek_meaning_remove:NTF *
3420   { \__enumext_starred_item_vii:w }
3421   { \__enumext_standard_item_vii:w }
3422 }

```

(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].

```

3423 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3424 {
3425   \bool_set_false:N \l__enumext_item_starred_vii_bool
3426   \peek_meaning:NTF [
3427   {
3428     \bool_set_eq:NN
3429     \l__enumext_wrap_label_vii_bool
3430     \l__enumext_wrap_label_opt_vii_bool
3431     \__enumext_start_item_vii:w
3432   }
3433   {
3434     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3435     \legacy_if_set_true:n { @noitemarg }
3436     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3437   }
3438 }

```

(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*⟩].

```

3439 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3440 {
3441   \bool_set_true:N \l__enumext_item_starred_vii_bool
3442   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3443   \peek_meaning:NTF [
3444   { \__enumext_starred_item_vii_aux_i:w }
3445   { \__enumext_starred_item_vii_aux_ii:w }
3446 }
3447 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3448 {
3449   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3450   \__enumext_starred_item_vii_aux_ii:w
3451 }
3452 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3453 {
3454   \peek_meaning:NTF [
3455   { \__enumext_starred_item_vii_aux_iii:w }
3456   {
3457     \dim_set_eq:NN
3458     \l__enumext_item_symbol_sep_vii_dim
3459     \l__enumext_labelsep_vii_dim
3460     \legacy_if_set_true:n { @noitemarg }
3461     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]

```

```

3462     }
3463   }
3464   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3465   {
3466     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3467     \legacy_if_set_true:n { @noitemarg }
3468     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3469   }

```

(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.

```

3470 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3471 {
3472   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3473   \legacy_if:nT { @noitemarg }
3474   {
3475     \legacy_if_set_false:n { @noitemarg }
3476     \legacy_if:nT { @nmbrrlist }
3477     {
3478       \bool_if:NT \l__enumext_hyperref_bool
3479       {
3480         \legacy_if_set_true:n { @hyper@item }
3481       }
3482       \refstepcounter{enumXvii}
3483       \bool_if:NT \l__enumext_check_ans_bool
3484       {
3485         \int_gincr:N \g__enumext_item_number_int
3486       }
3487     }
3488   }

```

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.

```

3489   \group_begin:
3490   \lrbox{ \l__enumext_item_text_vii_box }
3491   \bool_if:NF \l__enumext_footnotes_key_bool
3492   {
3493     \__enumext_renew_footnote:
3494   }
3495   \bool_if:NT \l__enumext_item_starred_vii_bool
3496   {
3497     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3498     {
3499       \tl_gset_eq:NN
3500       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3501     }
3502     \mode_leave_vertical:
3503     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3504     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3505     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3506     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3507   }
3508   \group_begin:
3509   \tl_use:N \l__enumext_label_font_style_vii_tl
3510   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3511   {
3512     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]

```

```

3513         { \__enumext_wrapper_label_vii:n {#1} }
3514     }
3515     {
3516         \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3517     }
3518     \group_end:
3519     \skip_horizontal:N \__enumext_labelsep_vii_dim
3520     \tl_use:N \__enumext_after_list_args_vii_tl
3521     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3522     \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3523     \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3524     \tl_use:N \__enumext_fake_item_indent_vii_tl
3525 }

```

(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.

```

3526 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3527 {
3528     \__enumext_endminipage:
3529     \endlrbox
3530     \group_end:
3531     \box_set_wd:Nn \__enumext_item_text_vii_box
3532     {
3533         \__enumext_joined_width_vii_dim
3534         + \__enumext_labelwidth_vii_dim
3535         + \__enumext_labelsep_vii_dim
3536     }
3537     \int_set:Nn \hbadness { 10000 }
3538     \box_use:N \__enumext_item_text_vii_box
3539     \bool_if:NF \__enumext_footnotes_key_bool
3540     {
3541         \__enumext_print_footnote:
3542     }
3543     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3544     {
3545         \par\noindent
3546         \int_zero:N \__enumext_item_column_pos_vii_int
3547     }
3548     { \hspace{ \__enumext_columns_sep_vii_dim } }
3549 }

```

(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.

```

3550 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3551 {
3552     \int_compare:nNnT
3553     {
3554         \int_mod:nn { \__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3555     }
3556     =
3557     { \c_zero_int }
3558     {
3559         \par
3560         \vspace{ -\__enumext_itemsep_vii_skip }
3561         \int_gzero:N \__enumext_item_count_all_vii_int
3562     }
3563 }

```

(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.

```

3564 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```


10.36 The environment keyans*

10.36.1 Functions for item box width

_enumext_starred_columns_set_viii:

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

```

3565 \cs_new_protected:Nn \_enumext_starred_columns_set_viii:
3566 {
3567   \dim_compare:nNnT { \_enumext_columns_sep_viii_dim } = { \c_zero_dim }
3568   {
3569     \dim_set:Nn \_enumext_columns_sep_viii_dim
3570     {
3571       ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim )
3572       / \_enumext_columns_viii_int
3573     }
3574   }
3575   \int_set:Nn \_enumext_tmpa_viii_int { \_enumext_columns_viii_int - \c_one_int }
3576   \dim_set:Nn \_enumext_item_width_viii_dim
3577   {
3578     ( \linewidth - \_enumext_columns_sep_viii_dim * \_enumext_tmpa_viii_int )
3579     / \_enumext_columns_viii_int - \_enumext_labelwidth_viii_dim
3580     - \_enumext_labelsep_viii_dim
3581   }
3582   \dim_zero_new:N \itemwidth
3583 }

```

(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`.

```

3584 \cs_new_protected:Npn \_enumext_starred_joined_item_viii:n #1
3585 {
3586   \int_set:Nn \_enumext_joined_item_viii_int {#1}
3587   \int_compare:nNnT { \_enumext_joined_item_viii_int } > { \_enumext_columns_viii_int }
3588   {
3589     \msg_warning:nnee { enumext } { item-joined }
3590     { \int_use:N \_enumext_joined_item_viii_int }
3591     { \int_use:N \_enumext_columns_viii_int }
3592     \int_set:Nn \_enumext_joined_item_viii_int
3593     {
3594       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3595     }
3596   }
3597   \int_compare:nNnT
3598   { \_enumext_joined_item_viii_int }
3599   >
3600   { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3601   {
3602     \msg_warning:nnee { enumext } { item-joined-columns }
3603     { \int_use:N \_enumext_joined_item_viii_int }
3604     {
3605       \int_eval:n
3606       { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3607     }
3608     \int_set:Nn \_enumext_joined_item_viii_int
3609     {
3610       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3611     }
3612   }
3613   \int_compare:nNnTF { \_enumext_joined_item_viii_int } > { \c_one_int }
3614   {
3615     \int_set_eq:NN \_enumext_joined_item_aux_viii_int \_enumext_joined_item_viii_int
3616     \int_decr:N \_enumext_joined_item_aux_viii_int
3617     \int_add:Nn \_enumext_item_column_pos_viii_int { \_enumext_joined_item_aux_viii_int }
3618     \int_gadd:Nn \_enumext_item_count_all_viii_int { \_enumext_joined_item_aux_viii_int }
3619     \dim_set:Nn \_enumext_joined_width_viii_dim
3620     {
3621       \_enumext_item_width_viii_dim * \_enumext_joined_item_viii_int
3622       + ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim
3623         + \_enumext_columns_sep_viii_dim
3624         ) * \_enumext_joined_item_aux_viii_int
3625     }

```

```

3626     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3627   }
3628   {
3629     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3630     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3631   }
3632 }

```

(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:
3633 \cs_new_protected:Nn \__enumext_start_mini_viii:
3634 {
3635   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3636   {
3637     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3638     {
3639       \linewidth
3640       - \l__enumext_minipage_right_viii_dim
3641       - \l__enumext_minipage_hsep_viii_dim
3642     }
3643     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3644     \dim_gset_eq:NN
3645       \g__enumext_minipage_right_viii_dim
3646       \l__enumext_minipage_right_viii_dim
3647     \__enumext_mini_addvspace_viii:
3648     \nointerlineskip\noindent
3649     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3650   }
3651 }
3652 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3653 {
3654   \bool_if:NT \l__enumext_minipage_active_viii_bool
3655   {
3656     \end{__enumext_mini_env*}
3657     \hfill
3658     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3659   }
3660 }
3661 \__enumext_after_env:nn {keyans*}
3662 {
3663   \bool_if:NT \g__enumext_minipage_active_viii_bool
3664   {
3665     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3666     \par\addvspace { \g__enumext_minipage_right_skip }
3667     \bool_if:NF \g__enumext_minipage_center_viii_bool
3668     {
3669       \centering
3670     }
3671     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3672     \end{__enumext_mini_env*}
3673     \par\addvspace{ \g__enumext_minipage_after_skip }
3674   }
3675   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3676   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3677   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3678   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3679 }

```

(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.

```

3680 \NewDocumentEnvironment{keyans*}{ o }
3681 {
3682   \__enumext_safe_exec_viii:
3683   \__enumext_parse_keys_viii:n {#1}
3684   \__enumext_before_list_viii:
3685   \__enumext_start_list:nn { }

```

```

3686     {
3687         \__enumext_list_arg_two_viii:
3688         \__enumext_before_keys_exec_viii:
3689     }
3690     \__enumext_starred_columns_set_viii:
3691     \item[] \scan_stop:
3692     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3693     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3694 }
3695 {
3696     \__enumext_stop_item_tmp_viii:
3697     \__enumext_remove_extra_parsep_viii:
3698     \__enumext_check_starred_cmd:n { item }
3699     \__enumext_stop_list:
3700     \__enumext_after_list_viii:
3701 }

```

(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.

```

3702 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3703 {
3704     \int_incr:N \__enumext_keyans_level_h_int
3705     \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
3706     {
3707         \msg_error:nn { enumext } { nested }
3708     }
3709     \__enumext_keyans_save_start_line:
3710     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3711     \bool_set_false:N \__enumext_store_active_bool
3712     \int_compare:nNnT { \__enumext_level_int } > { 1 }
3713     {
3714         \msg_error:nn { enumext } { keyans-wrong-level }
3715     }
3716 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

3717 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3718 {
3719     \tl_if_novalue:nF {#1}
3720     {
3721         \keys_set:nn { enumext / keyans* } {#1}
3722     }
3723 }

```

(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`.

```

3724 \cs_new_protected:Nn \__enumext_before_list_viii:
3725 {
3726     \__enumext_vspace_above_viii:
3727     \__enumext_before_args_exec_viii:
3728     \__enumext_start_mini_viii:
3729 }

```

(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.

```

3730 \cs_new_protected:Nn \__enumext_after_list_viii:
3731 {
3732     \__enumext_stop_mini_viii:
3733     \__enumext_after_stop_list_viii:
3734     \__enumext_vspace_below_viii:
3735 }

```

(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`.

```
3736 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3737 {
3738   \__enumext_stop_item_tmp_viii:
3739   \int_incr:N \l__enumext_item_column_pos_viii_int
3740   \int_gincr:N \g__enumext_item_count_all_viii_int
3741   \__enumext_item_peek_args_viii:
3742 }
```

(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).

```
3743 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3744 {
3745   \peek_meaning:NTF (
3746     { \__enumext_joined_item_viii:w }
3747     { \__enumext_joined_item_viii:w (1) }
3748 }
```

(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`.

```
3749 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3750 {
3751   \__enumext_starred_joined_item_viii:n {#1}
3752   \peek_meaning_remove:NTF *
3753     { \__enumext_starred_item_viii:w }
3754     { \__enumext_standard_item_viii:w }
3755 }
```

(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]`.

```
3756 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3757 {
3758   \bool_set_false:N \l__enumext_item_starred_viii_bool
3759   \peek_meaning:NTF [
3760     {
3761       \bool_set_eq:NN
3762         \l__enumext_wrap_label_viii_bool
3763         \l__enumext_wrap_label_opt_viii_bool
3764       \__enumext_start_item_viii:w
3765     }
3766     {
3767       \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

3768         \legacy_if_set_true:n { @noitemarg }
3769         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3770     }
3771 }

```

(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].

```

3772 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3773 {
3774     \bool_set_true:N \__enumext_item_starred_viii_bool
3775     \bool_set_true:N \__enumext_wrap_label_viii_bool
3776     \peek_meaning:NTF [
3777         { \__enumext_starred_item_viii_aux_i:w }
3778         { \__enumext_starred_item_viii_aux_ii:w }
3779     }

```

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

```

3780 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3781 {
3782     \tl_clear:N \__enumext_store_keyans_label_tl
3783     \tl_if_no_value:nF { #1 }
3784     {
3785         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3786         {
3787             \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
3788             \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
3789         }
3790         \tl_set:Ne \__enumext_keyans_item_opt_tl { #1 }
3791     }
3792     \__enumext_starred_item_viii_aux_ii:w
3793 }
3794 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3795 {
3796     \legacy_if_set_true:n { @noitemarg }
3797     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3798 }

```

(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.

```

3799 \cs_new_protected:Nn \__enumext_starred_item_exec:
3800 {
3801     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3802     \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3803     \__enumext_keyans_store_ref:
3804     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3805     \__enumext_keyans_addto_seq_link:
3806     \int_gincr:N \g__enumext_check_starred_cmd_int
3807     \bool_if:NT \__enumext_show_answer_bool
3808     {
3809         \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3810     }
3811     \bool_if:NT \__enumext_show_position_bool
3812     {
3813         \tl_set:Ne \__enumext_mark_answer_sym_tl
3814         {
3815             \group_begin:
3816             \exp_not:N \normalfont
3817             \exp_not:N \footnotesize [ \int_eval:n
3818                 {
3819                     \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3820                 }

```

```

3821         ]
3822         \group_end:
3823     }
3824     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
3825 }
3826 }

```

(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.

```

3827 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3828 {
3829     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3830     \legacy_if:nT { @noitemarg }
3831     {
3832         \legacy_if_set_false:n { @noitemarg }
3833         \legacy_if:nT { @nmbrlist }
3834         {
3835             \bool_if:NT \l__enumext_hyperref_bool
3836             {
3837                 \legacy_if_set_true:n { @hyper@item }
3838             }
3839             \refstepcounter{enumXviii}
3840         }
3841     }

```

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

```

3842     \group_begin:
3843     \lrbox{ \l__enumext_item_text_viii_box }
3844     \bool_if:NF \l__enumext_footnotes_key_bool
3845     {
3846         \__enumext_renew_footnote:
3847     }
3848     \bool_if:NT \l__enumext_item_starred_viii_bool
3849     {
3850         \__enumext_starred_item_exec:
3851     }
3852     \group_begin:
3853     \tl_use:N \l__enumext_label_font_style_viii_tl
3854     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3855     {
3856         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3857         { \__enumext_wrapper_label_viii:n {#1} }
3858     }
3859     {
3860         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3861     }
3862     \group_end:
3863     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3864     \tl_use:N \l__enumext_after_list_args_viii_tl
3865     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3866     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3867     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3868     \bool_if:NT \l__enumext_item_starred_viii_bool
3869     {
3870         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3871         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3872     }
3873     {
3874         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3875     }
3876 }

```

(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.

```

3877 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3878 {

```

```

3879     \__enumext_endminipage:
3880     \endlrbox
3881     \group_end:
3882     \box_set_wd:Nn \l__enumext_item_text_viii_box
3883     {
3884         \l__enumext_joined_width_viii_dim
3885         + \l__enumext_labelwidth_viii_dim
3886         + \l__enumext_labelsep_viii_dim
3887     }
3888     \int_set:Nn \hbadness { 10000 }
3889     \box_use:N \l__enumext_item_text_viii_box
3890     \bool_if:NF \l__enumext_footnotes_key_bool
3891     {
3892         \__enumext_print_footnote:
3893     }
3894     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3895     {
3896         \par\noindent
3897         \int_zero:N \l__enumext_item_column_pos_viii_int
3898     }
3899     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3900 }

```

(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.

```

3901 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3902 {
3903     \int_compare:nNnT
3904     {
3905         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3906     }
3907     =
3908     { \c_zero_int }
3909     {
3910         \par
3911         \vspace{ -\l__enumext_itemsep_viii_skip }
3912         \int_gzero:N \g__enumext_item_count_all_viii_int
3913     }
3914 }

```

(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.

```

3915 \NewDocumentCommand \getkeyans { m }
3916 {
3917     \exp_args:Ne \__enumext_getkeyans_aux:n
3918     { \tl_to_str:e { \text_expand:n {#1} } }
3919 }

```

(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.

```

3920 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3921 {
3922     \str_if_in:nnTF {#1} { : }
3923     {
3924         \use:e
3925         {
3926             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3927             { {##1} {##2} }
3928         }
3929         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3930     }
3931     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3932 }

```

(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⟩*.

```

3933 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3934 {
3935   \prop_if_exist:cF { g__enumext_#1_prop }
3936   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3937   \group_begin:
3938   \prop_item:cn { g__enumext_#1_prop }{#2}
3939   \group_end:
3940 }

```

(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.

```

3941 \keys_define:nn { keyanskey / print }
3942 {
3943   level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3944                       {
3945                         \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3946                       },
3947   level-1 .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3948   level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3949                       {
3950                         \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3951                       },
3952   level-2 .initial:n   = { nosep, label=(\alph*), first=\small, font=\small },
3953   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3954                       {
3955                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3956                       },
3957   level-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
3958   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3959                       {
3960                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3961                       },
3962   level-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
3963   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3964                       {
3965                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3966                       },
3967   level-* .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3968 }

```

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

```

3969 \NewDocumentCommand \printkeyans { s O{} m }
3970 {
3971   \group_begin:
3972   \tl_use:N \l__enumext_print_keyans_i_tl
3973   \tl_use:N \l__enumext_print_keyans_ii_tl
3974   \tl_use:N \l__enumext_print_keyans_iii_tl
3975   \tl_use:N \l__enumext_print_keyans_iv_tl
3976   \tl_use:N \l__enumext_print_keyans_vii_tl
3977   \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3978   \group_end:
3979 }

```

(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

```

3980 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3981 {
3982   \seq_if_exist:cTF { g__enumext_#3_seq }
3983   {
3984     \seq_if_empty:cF { g__enumext_#3_seq }
3985     {
3986       %%\seq_show:c { g__enumext_#3_seq }
3987       \bool_if:nTF {#1}
3988       {
3989         \begin{enumext*}[#2]
3990         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3991         \end{enumext*}
3992       }
3993       {
3994         \begin{enumext}[#2]
3995         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3996         \end{enumext}
3997       }
3998     }
3999   }
4000   {
4001     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4002   }
4003 }

```

(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.

```

4004 \keys_define:nn { enumext / meta-families }
4005 {
4006   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4007   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4008   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4009   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4010   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4011   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4012   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4013   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
4014   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
4015   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
4016   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
4017   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
4018   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4019 }

```

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

```

4020 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4021 {
4022   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
4023   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
4024 }

```

\setenumext Now we define the user command \setenumext.

```

4025 \NewDocumentCommand \setenumext { o +m }
4026 {
4027   \tl_if_novalue:nTF {#1}
4028   {
4029     \seq_map_inline:Nn \c__enumext_all_families_seq
4030     {
4031     {
4032       \seq_clear:N \l__enumext_setkey_tmpa_seq
4033       \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4034       \int_set:Nn \l__enumext_setkey_tmpa_int
4035       {
4036         \seq_count:N \l__enumext_setkey_tmpb_seq
4037       }
4038       \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4039       {

```

```

4040         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4041         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4042         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4043         {
4044             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4045         }
4046     }
4047     {
4048         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4049     }
4050     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4051     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4052     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4053 }
4054 {
4055     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4056 }
4057 }

```

(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.

```

4058 \cs_new_protected:Npn \__enumext_set_parse:n #1
4059 {
4060     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4061     \int_step_inline:nnn { 0 } { 4 } % <- max level
4062     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4063     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4064     {
4065         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4066         { \tl_trim_spaces:n {#1} }
4067     }
4068     { \__enumext_set_error:nn {#1} { } }
4069 }
4070 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4071 { \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.

```

4072 \msg_new:nnn { enumext } { package-load }
4073 {
4074     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4075 }
4076 \msg_new:nnn { enumext } { package-not-load }
4077 {
4078     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4079 }
4080 \msg_new:nnn { enumext } { package-load-foot }
4081 {
4082     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4083 }

```

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

```

4084 \msg_new:nnn { enumext } { counters }
4085 {
4086     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4087     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4088 }

```

Message used in the creation of *⟨prop list⟩* by `enumext` package.

```

4089 \msg_new:nnn { enumext } { store-prop }
4090 {
4091     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4092 }
4093 \msg_new:nnn { enumext } { store-seq }
4094 {
4095     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4096 }
4097 \msg_new:nnn { enumext } { store-int }
4098 {

```

```

4099     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_int ~ \msg_line_context:.
4100 }

```

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

```

4101 \msg_new:nnn { enumext } { invalid-key }
4102 {
4103     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4104 }
4105 \msg_new:nnn { enumext } { unknown-key-family }
4106 {
4107     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4108 }

```

Messages used in length calculation.

```

4109 \msg_new:nnn { enumext } { width-negative }
4110 {
4111     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4112     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4113 }
4114 \msg_new:nnn { enumext } { width-zero }
4115 {
4116     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4117     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4118 }

```

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

```

4119 \msg_new:nnn { enumext } { list-lengths }
4120 {
4121     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4122     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4123     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4124     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4125     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4126     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4127     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4128     \__enumext_show_length:nnn { skip } { topsep } {#1}
4129     \__enumext_show_length:nnn { skip } { parsep } {#1}
4130     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4131     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4132     ****
4133 }

```

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

```

4134 \msg_new:nnn { enumext } { list-lengths-not-nested }
4135 {
4136     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4137     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4138     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4139     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4140     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4141     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4142     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4143     \__enumext_show_length:nnn { skip } { topsep } {#1}
4144     \__enumext_show_length:nnn { skip } { parsep } {#1}
4145     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4146     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4147     ****
4148 }

```

Messages used by `ref` key.

```

4149 \msg_new:nnn { enumext } { key-ref-empty }
4150 {
4151     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4152 }

```

Messages used by `save-ans` key.

```

4153 \msg_new:nnn { enumext } { save-ans-empty }
4154 {
4155     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4156 }
4157 \msg_new:nnn { enumext } { save-ans-ok }
4158 {
4159     Set ~ key~ 'save-ans=#1' ~ \msg_line_context:.
4160 }

```

```

4161 % Start environment enumext with key save-ans=algo on line ddd
4162 \msg_new:nnn { enumext } { save-ans-log }
4163 {
4164   * ~ Package ~ enumext: ~ Start ~ #1 ~ with ~ key ~ save-ans=#2 ~ \msg_line_context:.
4165 }
4166 \msg_new:nnn { enumext } { save-ans-log-hook }
4167 {
4168   * ~ Package ~ enumext: ~ Stop ~ #1 ~ with ~ key ~ save-ans=#2 ~ \msg_line_context:.
4169 }
4170 % Stop ~ storing in 'test-1990' on line 15.
4171 \msg_new:nnn { enumext } { save-ans-hook }
4172 {
4173   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4174 }

```

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

```

4175 \msg_new:nnn { enumext } { need-save-ans }
4176 {
4177   Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4178 }
4179 \msg_new:nnn { enumext } { items-same-answer }
4180 {
4181   *****~Checking~answers~on~'#1'~OK~*****\\
4182   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4183   *****
4184   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4185 }
4186 \msg_new:nnn { enumext } { item-different-answer }
4187 {
4188   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4189   answer ~ stored ~ in ~ '#1'~ #2.
4190 }

```

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

```

4191 \msg_new:nnn { enumext } { missing-starred }
4192 {
4193   Missing ~ '\c_backslash_str #1*' ~ #2.
4194 }
4195 \msg_new:nnn { enumext } { many-starred }
4196 {
4197   Many ~ '\c_backslash_str #1*' ~ #2.
4198 }

```

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

```

4199 \msg_new:nnn { enumext } { list-too-deep }
4200 {
4201   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:.. \\
4202   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4203 }

```

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

```

4204 \msg_new:nnn { enumext } { anskey-wrong-place }
4205 {
4206   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.. \\
4207   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4208 }
4209 \msg_new:nnn { enumext } { anspic-wrong-place }
4210 {
4211   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.. \\
4212   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4213 }
4214 \msg_new:nnn { enumext } { command-wrong-place }
4215 {
4216   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.. \\
4217   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4218 }

```

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

```

4219 \msg_new:nnn { enumext } { keyans-nested }
4220 {
4221   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4222 }
4223 \msg_new:nnn { enumext } { keyans-wrong-level }

```

```

4224 {
4225     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4226     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4227 }
4228 \msg_new:nnn { enumext } { wrong-place }
4229 {
4230     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
4231     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4232 }
4233 \msg_new:nnn { enumext } { keyanspic-nested }
4234 {
4235     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
4236 }
4237 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4238 {
4239     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4240     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4241 }

```

Messages used by `\getkeyans` command.

```

4242 \msg_new:nnn { enumext } { undefined-storage-anskey }
4243 {
4244     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4245 }

```

Messages used by `\miniright` command.

```

4246 \msg_new:nnn { enumext } { missing-miniright }
4247 {
4248     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4249     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4250 }
4251 \msg_new:nnn { enumext } { wrong-miniright-place }
4252 {
4253     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4254     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4255 }
4256 \msg_new:nnn { enumext } { wrong-miniright-use }
4257 {
4258     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4259     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4260 }

```

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

```

4261 \msg_new:nnn { enumext } { nested }
4262 {
4263     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4264 }
4265 \msg_new:nnn { enumext } { item-joined }
4266 {
4267     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4268 }
4269 \msg_new:nnn { enumext } { item-joined-columns }
4270 {
4271     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4272 }

```

10.41 Finish package

Finish package implementation.

```

4273 \file_input_stop:
4274 </package>

```

11 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>*</code>	201
<code>\+</code>	193
<code>\-</code>	193
<code>\\</code> 209, 3079, 4086, 4111, 4116, 4121, 4136, 4181, 4182, 4201, 4206, 4211, 4216, 4225, 4230, 4239, 4248, 4253, 4258	
A	
above	<u>1352</u>
above*	<u>1352</u>
<code>\addvspace</code> . 999, 1027, 1150, 1229, 1292, 1298, 1326, 1343, 2918, 2933, 3035, 3050, 3276, 3283, 3666, 3673	
after	<u>837</u>
align	<u>454</u>
<code>\Alph</code>	32, <u>36</u> , 37
<code>\Alph</code>	406, 515, 560, 628, 3962
<code>\alph</code>	32, <u>36</u> , 37
<code>\alph</code>	407, 513, 3952
<code>\anskey</code>	11, 65, <u>2056</u>
<code>\anspic</code>	13, 86, <u>3057</u>
<code>\anspic*</code>	61
<code>\arabic</code>	27, 32
<code>\arabic</code>	405, 512, 559, 3947, 3967
B	
<code>\b</code>	2796, 2809, 3342, 3355
<code>\baselineskip</code>	44
<code>\baselineskip</code>	2016, 2024
before	<u>837</u>
before*	<u>837</u>
below	<u>1352</u>
below*	<u>1352</u>
bool commands:	
<code>\bool_gset_false:N</code> 311, 312, 313, 3285, 3289, 3675	
<code>\bool_gset_true:N</code> 221, 230, 941, 1844, 1850, 3268, 3286, 3658, 3676	
<code>\bool_if:NTF</code> . . 303, 346, 358, 375, 1374, 1388, 1401, 1412, 1423, 1434, 1445, 1456, 1509, 1526, 1531, 1539, 1566, 1604, 1609, 1616, 1620, 1642, 1647, 1655, 1662, 1693, 1701, 1786, 1790, 1794, 1831, 1978, 2002, 2009, 2037, 2068, 2081, 2083, 2094, 2114, 2239, 2250, 2254, 2293, 2308, 2383, 2394, 2398, 2511, 2541, 2615, 2631, 2693, 2703, 2733, 2738, 2786, 2794, 2807, 2852, 2902, 2916, 2924, 2963, 3020, 3033, 3041, 3059, 3264, 3273, 3277, 3332, 3340, 3353, 3385, 3395, 3478, 3483, 3491, 3495, 3510, 3539, 3654, 3663, 3667, 3807, 3811, 3835, 3844, 3848, 3854, 3868, 3890	
<code>\bool_if:nTF</code> 1327, 1344, 2122, 2552, 2587, 2651, 3080, 3987	
<code>\bool_if_p:N</code> . 239, 253, 1673, 1674, 1682, 1683, 1811, 1841, 1842, 1847, 1848, 2105, 2148, 2149, 2173, 2182, 2183, 2195, 2211, 2370, 2371, 2408, 2409, 2825, 2838, 2840, 3087, 3088	
<code>\bool_lazy_all:nTF</code> 237, 251, 1809, 2171, 2180, 2193, 2209, 2823, 2836	
<code>\bool_lazy_and:nnTF</code> . . 217, 226, 1672, 1681, 1840, 1846, 2104, 2147, 2369	
<code>\bool_lazy_or:nnTF</code> 1724, 1732, 2407, 3086	
<code>\bool_new:N</code> 25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88, 93, 94, 97, 115, 117, 119, 122, 123, 132, 133, 134, 141, 142, 156, 167, 169	
<code>\bool_not_p:n</code> 218, 227, 2106, 2198, 2213, 2826, 2827, 2839	
<code>\bool_set_eq:NN</code> 2519, 2567, 3428, 3761	
<code>\bool_set_false:N</code> 355, 1773, 1774, 2939, 2971, 3053, 3118, 3136, 3380, 3425, 3711, 3758	
<code>\bool_set_true:N</code> 244, 258, 337, 341, 447, 765, 1358, 1363, 1629, 1746, 1747, 1941, 1948, 2515, 2545, 2563, 2575, 2769, 2832, 2845, 2871, 2968, 2995, 3253, 3322, 3434, 3441, 3442, 3643, 3767, 3774, 3775	
box commands:	
<code>\box_dp:N</code> . . 1046, 1050, 1054, 1065, 1069, 1080, 1089, 1095, 1105, 1118, 1124, 1130, 1161, 1162, 1163, 1166, 1176, 1180, 1189, 1196, 1201, 1209, 1238, 1239, 1242, 1249, 1262, 1270, 1276, 1284, 3148	
<code>\box_new:N</code>	58, 162
<code>\box_set_wd:Nn</code>	3531, 3882
<code>\box_use:N</code>	3538, 3889
<code>\box_wd:N</code>	413
C	
<code>\c</code>	201, 202, 665, 667, 679, 681
<code>\cB</code>	202
<code>\cE</code>	202
<code>\centering</code>	1329, 1346, 3169, 3279, 3669
check-ans	<u>1765</u>
Document class:	
article	38
clist commands:	
<code>\clist_const:Nn</code>	174
<code>\clist_map_function:nN</code>	3156
<code>\clist_map_inline:Nn</code> . 453, 707, 770, 836, 851, 932, 1368	
<code>\clist_map_inline:nn</code> 36, 47, 66, 72, 84, 96, 121, 150, 173, 478, 495, 775, 947, 1474, 1718, 1779, 1918, 1936, 1957, 2168, 2302, 2469, 2680, 2683, 2710, 2720, 2723, 2743	
<code>\columnbreak</code>	66
<code>\columnbreak</code>	2108
columns	<u>916</u>
columns*	<u>1937</u>
columns-sep	<u>916</u>
columns-sep*	<u>1937</u>
<code>\columnsep</code>	82, 85
<code>\columnsep</code>	2896, 3017
<code>\columnseprule</code>	82, 85
<code>\columnseprule</code>	2900, 3019
Commands provide by enumext:	
<code>\anskey</code> 24, 25, 58, 59, 63, 65, 67–69, 71, 80, 93, 103, 104, 108	
<code>\anspic*</code>	24, 25, 61, 69, 70, 86–88, 103, 104
<code>\anspic</code>	63, 86–88, 108
<code>\getkeyans</code>	63, 103, 109
<code>\item*</code> . . 24, 25, 61, 63, 69, 70, 74, 75, 94, 101, 103, 104	
<code>\itemwidth</code>	89, 97
<code>\item</code>	73, 75, 89, 93–95, 97, 100
<code>\miniright</code>	24, 42, 49, 50, 81, 83–85, 109
<code>\printkeyans</code>	25, 63, 104

`\setenumext` 25, 105–107
 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` 415, 431, 671, 687, 1970, 1975, 2055, 2670, 3158
 `\cs_if_exist:NTF` 385
 `\cs_new:Nn` 187
 `\cs_new:Npn` 205, 1475, 1484, 1493
 `\cs_new_eq:NN` 321, 322, 323, 327, 328, 360, 361, 364, 365
 `\cs_new_protected:Nn` . 197, 211, 235, 266, 293, 332, 536, 599, 651, 852, 856, 860, 864, 868, 872, 876, 880, 884, 888, 892, 896, 900, 904, 908, 912, 948, 960, 984, 1001, 1012, 1036, 1111, 1135, 1152, 1214, 1231, 1253, 1288, 1294, 1369, 1383, 1397, 1408, 1419, 1430, 1441, 1452, 1537, 1640, 1653, 1670, 1691, 1744, 1784, 1804, 1838, 1853, 1976, 2000, 2007, 2035, 2042, 2159, 2291, 2306, 2334, 2367, 2403, 2415, 2423, 2474, 2478, 2497, 2548, 2583, 2599, 2609, 2625, 2763, 2821, 2850, 2857, 2880, 2910, 2922, 2961, 2985, 3003, 3028, 3039, 3076, 3120, 3134, 3154, 3159, 3175, 3243, 3262, 3314, 3367, 3374, 3383, 3393, 3410, 3550, 3565, 3633, 3652, 3702, 3724, 3730, 3743, 3799, 3901
 `\cs_new_protected:Npn` 179, 183, 368, 383, 400, 410, 416, 516, 561, 633, 658, 672, 1316, 1335, 1505, 1524, 1594, 1627, 1719, 1866, 1962, 1971, 2091, 2236, 2248, 2270, 2344, 2388, 2507, 2525, 2559, 2571, 2639, 2673, 2713, 2772, 2792, 2981, 3129, 3194, 3325, 3338, 3416, 3423, 3439, 3447, 3452, 3464, 3584, 3717, 3749, 3756, 3772, 3780, 3794, 3920, 3933, 3980, 4058, 4070
 `\cs_new_protected_nopar:Nn` ... 3403, 3526, 3736, 3877
 `\cs_new_protected_nopar:Npn` 3470, 3827
 `\cs_set:Nn` 2241
 `\cs_set:Npn` 2169, 2207, 3926
 `\cs_set_eq:NN` .. 3304, 3305, 3472, 3692, 3693, 3829
 `\cs_set_protected:Nn` 776, 792, 804, 816
 `\cs_set_protected:Npn` . 32, 41, 59, 67, 79, 85, 111, 146, 154, 432, 454, 483, 496, 543, 688, 708, 752, 771, 828, 837, 916, 933, 1352, 1463, 1710, 1765, 1883, 1919, 1937, 2161, 2295, 2458, 2671, 2711
 `\cs_to_str:N` 402, 425

D

`\d` 193
`\DeclareDocumentEnvironment` 1029
 dim commands:
 `\dim_abs:n` 2644, 2649
 `\dim_add:Nn` 3139
 `\dim_compare:nNnTF` . 778, 794, 806, 818, 1318, 1337, 2641, 2646, 2652, 2658, 2660, 2662, 2862, 2885, 2989, 3007, 3131, 3177, 3245, 3567, 3635
 `\dim_compare:nTF` 2132
 `\dim_gset_eq:NN` 3254, 3644
 `\dim_gzero:N` 3288, 3678
 `\dim_new:N` . 55, 62, 63, 64, 81, 118, 128, 163, 164, 170
 `\dim_set:Nn` .. 413, 766, 1949, 2539, 2644, 2649, 2651, 2654, 2655, 2659, 2661, 2664, 2665, 2667, 2865, 2888,

2991, 3009, 3161, 3179, 3186, 3229, 3247, 3466, 3569, 3576, 3619, 3637
`\dim_set_eq:NN` 503, 550, 621, 625, 2534, 2682, 2722, 2811, 2896, 3017, 3236, 3239, 3240, 3357, 3457, 3626, 3629, 3630
`\dim_use:N` 779, 787, 1319, 1325, 2045, 2048, 2053, 2604, 2606, 2863, 2868, 2869, 2876, 2886, 2890, 2891, 2893
`\dim_zero:N` 2900, 3019, 3140, 3141, 3142
`\dim_zero_new:N` 3192, 3582
`\c_zero_dim` 781, 795, 807, 819, 1319, 1337, 2134, 2641, 2646, 2652, 2659, 2863, 2886, 2989, 3007, 3177, 3245, 3567, 3635

E

`\end ..` 1322, 1340, 2004, 2039, 2915, 2932, 3032, 3049, 3266, 3282, 3656, 3672, 3991, 3996
`\endlist` 29
`\endlist` 322
`\endlrbox` 3529, 3880
`\endminipage` 30
`\endminipage` 328
`enumext` 5, 2744
 enumext internal commands:

`\l__enumext__check_start_line_env_tl` ... 28
`\l__enumext__ref_the_count_tl` 34
`\l__enumext__resume_name_tl` 54, 55
`__enumext_add_pre_parsep:` ... 43, 958, 960, 960
`__enumext_after_args_exec:` . 40, 852, 864, 2756
`__enumext_after_args_exec_v:` . 41, 42, 868, 880, 2954
`__enumext_after_args_exec_vii:` ... 884, 908
`__enumext_after_args_exec_viii:` 912
`__enumext_after_env:nn` . 29, 61, 83, 96, 183, 183, 2942, 3271, 3564, 3661
`__enumext_after_hyperref:` ... 30, 330, 332, 332
`__enumext_after_list:` 83, 93, 99, 2761, 2922, 2922
`\l__enumext_after_list_args_v_tl` 882
`\l__enumext_after_list_args_vii_tl` 910, 3520
`\l__enumext_after_list_args_viii_tl` 914, 3864
`__enumext_after_list_v:` .. 85, 2959, 3039, 3039
`__enumext_after_list_vii:` ... 3312, 3374, 3374
`__enumext_after_list_viii:` .. 3700, 3730, 3730
`__enumext_after_star_env:nn` 90
`__enumext_after_stop_list:` ... 40, 41, 852, 860, 2937
`__enumext_after_stop_list_v:` 41, 868, 876, 3054
`\l__enumext_after_stop_list_v_tl` 878
`__enumext_after_stop_list_vii:` 884, 900, 3377
`\l__enumext_after_stop_list_vii_tl` ... 902
`__enumext_after_stop_list_viii:` . 904, 3733
`\l__enumext_after_stop_list_viii_tl` ... 906
`\l__enumext_align_label_vii_str` .. 3512, 3516
`\l__enumext_align_label_viii_str` . 3856, 3860
`\l__enumext_align_label_X_str` 154
`\c__enumext_all_envs_clist` .. 174, 453, 707, 770, 836, 851, 932, 1368
`\c__enumext_all_families_seq` .. 105, 4020, 4029, 4051
`__enumext_anskey_wrapper:n` 1887, 2246
`__enumext_at_begin_document:n` 29, 30, 179, 179, 319, 325
`__enumext_before_args_exec:` 40, 852, 852, 2860
`__enumext_before_args_exec_v:` .. 41, 868, 868, 2988

`__enumext_before_args_exec_vii:` .. [884](#), [884](#), [3371](#)
`__enumext_before_args_exec_viii:` [888](#), [3727](#)
`__enumext_before_keys_exec:` [40](#), [852](#), [856](#), [2754](#)
`__enumext_before_keys_exec_v:` .. [41](#), [868](#), [872](#), [2952](#)
`__enumext_before_keys_exec_vii` [884](#)
`__enumext_before_keys_exec_vii:` [41](#), [892](#), [3300](#)
`__enumext_before_keys_exec_viii:` .. [41](#), [896](#), [3688](#)
`__enumext_before_list:` ... [81](#), [2748](#), [2857](#), [2857](#)
`__enumext_before_list_v:` . [84](#), [2947](#), [2985](#), [2985](#)
`__enumext_before_list_vii:` [92](#), [3295](#), [3367](#), [3367](#)
`__enumext_before_list_viii:` .. [99](#), [3684](#), [3724](#), [3724](#)
`\l__enumext_before_no_starred_key_v_tl` [874](#)
`\l__enumext_before_no_starred_key_vii_-tl` [894](#)
`\l__enumext_before_no_starred_key_viii_-tl` [898](#)
`\l__enumext_before_starred_key_v_tl` ... [870](#)
`\l__enumext_before_starred_key_vii_tl` . [886](#)
`\l__enumext_before_starred_key_viii_tl` [890](#)
`__enumext_calc_hspace:NNNNNN` [77](#), [2639](#), [2639](#), [2670](#), [2675](#), [2715](#)
`__enumext_check_ans:` .. [60](#), [81](#), [1784](#), [1784](#), [2861](#), [3370](#)
`\g__enumext_check_ans_bool` ... [60](#), [132](#), [303](#), [311](#), [1844](#), [1850](#)
`\l__enumext_check_ans_bool` [60](#), [73](#), [74](#), [132](#), [1769](#), [1774](#), [1786](#), [1841](#), [1847](#), [2083](#), [2383](#), [2511](#), [2541](#), [3483](#)
`\g__enumext_check_ans_item_tl` [71](#)
`__enumext_check_ans_level:` [60](#), [1784](#), [1800](#), [1804](#)
`__enumext_check_ans_show:` . [61](#), [305](#), [1853](#), [1853](#)
`\g__enumext_check_ans_show_bool` [83](#)
`__enumext_check_ans_to_hook:` . [60](#), [1838](#), [1838](#), [2936](#), [3378](#)
`__enumext_check_starred_cmd:n` [28](#), [61](#), [71](#), [1866](#), [1866](#), [2957](#), [3115](#), [3698](#)
`\g__enumext_check_starred_cmd_int` [132](#), [1869](#), [1875](#), [1880](#), [2581](#), [3085](#), [3806](#)
`\l__enumext_check_start_line_env_tl` [132](#), [272](#), [279](#), [286](#), [1872](#), [1878](#), [1881](#)
`\l__enumext_columns_sep_v_dim` [3007](#), [3009](#), [3017](#)
`\l__enumext_columns_sep_vii_dim` .. [3177](#), [3179](#), [3188](#), [3233](#), [3359](#), [3548](#)
`\l__enumext_columns_sep_viii_dim` . [3567](#), [3569](#), [3578](#), [3623](#), [3899](#)
`\l__enumext_columns_v_int` [1157](#), [3005](#), [3013](#), [3025](#), [3030](#)
`\l__enumext_columns_vii_int` .. [3182](#), [3185](#), [3189](#), [3197](#), [3201](#), [3204](#), [3210](#), [3216](#), [3220](#), [3346](#), [3543](#), [3554](#)
`\l__enumext_columns_viii_int` . [3572](#), [3575](#), [3579](#), [3587](#), [3591](#), [3594](#), [3600](#), [3606](#), [3610](#), [3894](#), [3905](#)
`\g__enumext_count_item_with_ans_int` [65](#)
`\l__enumext_counter_i_tl` [32](#), [392](#)
`\l__enumext_counter_ii_tl` [32](#), [393](#)
`\l__enumext_counter_iii_tl` [32](#), [394](#)
`\l__enumext_counter_iv_tl` [32](#), [395](#)
`\c__enumext_counter_style_tl` [27](#), [37](#), [199](#)
`\g__enumext_counter_styles_tl` . [23](#), [32](#), [55](#), [403](#), [421](#)
`\l__enumext_counter_v_tl` [32](#), [396](#), [641](#)
`\l__enumext_counter_vi_tl` [32](#), [397](#)
`\l__enumext_counter_vii_tl` [32](#), [398](#), [571](#)
`\l__enumext_counter_viii_tl` [32](#), [399](#), [588](#)
`\l__enumext_current_widest_dim` [23](#), [55](#), [427](#), [504](#), [551](#), [622](#), [626](#)
`__enumext_default_item:n` ... [2507](#), [2507](#), [2556](#)
`__enumext_define_counters:Nn` [23](#), [383](#), [383](#), [392](#), [393](#), [394](#), [395](#), [396](#), [397](#), [398](#), [399](#)
`__enumext_endminipage:` [30](#), [325](#), [328](#), [1035](#), [3171](#), [3528](#), [3879](#)
`\g__enumext_envir_name_tl` [137](#), [245](#), [259](#), [302](#), [316](#), [1728](#), [1739](#)
`__enumext_execute_after_env:` [29](#), [293](#), [293](#), [2942](#), [3564](#)
`__enumext_fake_item:` [776](#), [776](#), [2702](#)
`\l__enumext_fake_item_indent_v_dim` [795](#), [800](#)
`\l__enumext_fake_item_indent_v_tl` [797](#), [2564](#), [2568](#), [2576](#)
`\l__enumext_fake_item_indent_vii_dim` [807](#), [812](#)
`\l__enumext_fake_item_indent_vii_tl` [809](#), [3524](#)
`\l__enumext_fake_item_indent_viii_dim` . [819](#), [824](#), [3871](#)
`\l__enumext_fake_item_indent_viii_tl` .. [821](#), [3870](#), [3874](#)
`\l__enumext_fake_item_indent_X_tl` [85](#)
`__enumext_fake_item_vii:` [776](#), [804](#), [2732](#)
`__enumext_fake_item_viii:` [776](#), [816](#), [2737](#)
`__enumext_filter_series:n` [53](#), [1475](#), [1475](#), [1517](#), [1529](#), [1534](#)
`__enumext_filter_series_key:n` [53](#), [1475](#), [1480](#), [1484](#)
`__enumext_filter_series_pair:nn` .. [53](#), [1475](#), [1481](#), [1493](#)
`\g__enumext_footnote_arg_seq` . [151](#), [2480](#), [2493](#), [2503](#)
`\g__enumext_footnote_int` . [151](#), [2487](#), [2490](#), [2492](#), [2494](#)
`\g__enumext_footnote_int_seq` . [151](#), [2481](#), [2494](#), [2499](#), [2502](#)
`__enumext_footnotes_key_bool` [30](#)
`\l__enumext_footnotes_key_bool` [25](#), [30](#), [95](#), [141](#), [341](#), [346](#), [355](#), [3491](#), [3539](#), [3844](#), [3890](#)
`__enumext_footnotetext:nn` ... [2474](#), [2474](#), [2504](#)
`__enumext_getkeyans:nn` .. [104](#), [3929](#), [3933](#), [3933](#)
`__enumext_getkeyans_aux:n` [103](#), [3917](#), [3920](#), [3920](#)
`\l__enumext_hyperref_bool` [25](#), [30](#), [141](#), [337](#), [358](#), [375](#), [2149](#), [2371](#), [3478](#), [3835](#)
`__enumext_hypertarget:nn` [30](#), [332](#), [360](#), [364](#), [380](#)
`__enumext_if_is_int:n` [191](#)
`__enumext_if_is_int:nTF` [191](#), [660](#), [674](#)
`__enumext_is_not_nested:` [27](#), [79](#), [211](#), [211](#), [2765](#), [3316](#)
`__enumext_is_on_first_level:` . [28](#), [79](#), [211](#), [235](#), [2770](#), [3323](#)
`\g__enumext_item_anskey_int` . [71](#), [132](#), [310](#), [1856](#), [2085](#), [2385](#)
`\l__enumext_item_column_pos_vii_int` [93](#), [3204](#), [3210](#), [3216](#), [3220](#), [3227](#), [3406](#), [3543](#), [3546](#)
`\l__enumext_item_column_pos_viii_int` .. [100](#), [3594](#), [3600](#), [3606](#), [3610](#), [3617](#), [3739](#), [3894](#), [3897](#)
`\l__enumext_item_column_pos_X_int` [154](#)
`\g__enumext_item_count_all_vii_int` [93](#), [3228](#), [3407](#), [3554](#), [3561](#)
`\g__enumext_item_count_all_viii_int` [100](#), [3618](#), [3740](#), [3905](#), [3912](#)

`\g__enumext_item_count_all_X_int` 154
`\g__enumext_item_number_int` . 60, 132, 309, 1815, 1819, 1822, 1825, 1833, 1856, 2513, 2543, 3485
`__enumext_item_peek_args_vii:` 93, 3408, 3410, 3410
`__enumext_item_peek_args_viii:` . . 100, 3741, 3743, 3743
`__enumext_item_starred:` . . 76, 2599, 2599, 2617
`\l__enumext_item_starred_vii_bool` 3425, 3441, 3495
`\l__enumext_item_starred_viii_bool` 3758, 3774, 3848, 3868
`\l__enumext_item_starred_X_bool` 154
`__enumext_item_std:w` 29, 73–75, 88, 319, 323, 2516, 2522, 2546, 2564, 2568, 2576, 3152
`\g__enumext_item_symbol_aux_vii_tl` 3449, 3497, 3500, 3504, 3506
`\g__enumext_item_symbol_aux_X_tl` 154
`\l__enumext_item_symbol_sep_vii_dim` . . 3458, 3466, 3503, 3505
`\g__enumext_item_symbol_tl` 23, 74, 48, 2531, 2605, 2622
`\l__enumext_item_symbol_vii_tl` 3500
`\l__enumext_item_text_vii_box` 3490, 3531, 3538
`\l__enumext_item_text_viii_box` 3843, 3882, 3889
`\l__enumext_item_text_X_box` 154
`\l__enumext_item_width_vii_dim` . . 3186, 3231, 3239, 3240
`\l__enumext_item_width_viii_dim` . . 3576, 3621, 3629, 3630
`\l__enumext_item_width_X_dim` 154
`\l__enumext_itemindent_X_dim` 59
`\l__enumext_itemsep_vii_skip` 3560
`\l__enumext_itemsep_viii_skip` 3911
`\l__enumext_joined_item_aux_vii_int` . . 3225, 3226, 3227, 3228, 3234
`\l__enumext_joined_item_aux_viii_int` . 3615, 3616, 3617, 3618, 3624
`\l__enumext_joined_item_aux_X_int` . . . 154
`__enumext_joined_item_vii:w` 93, 94, 3413, 3414, 3416, 3416
`\l__enumext_joined_item_vii_int` . . 3196, 3197, 3200, 3202, 3208, 3213, 3218, 3223, 3225, 3231
`__enumext_joined_item_viii:w` . 100, 3746, 3747, 3749, 3749
`\l__enumext_joined_item_viii_int` . 3586, 3587, 3590, 3592, 3598, 3603, 3608, 3613, 3615, 3621
`\l__enumext_joined_item_X_int` 154
`\l__enumext_joined_width_vii_dim` . 3229, 3236, 3239, 3521, 3533
`\l__enumext_joined_width_viii_dim` 3619, 3626, 3629, 3865, 3884
`\l__enumext_joined_width_X_dim` 154
`__enumext_keyans_addto_prop:n` 69, 2270, 2270, 2578, 3082
`__enumext_keyans_addto_seq:n` . 70, 2344, 2344, 2580, 3084
`__enumext_keyans_addto_seq_link:` 2344, 2365, 2367, 3805
`__enumext_keyans_anspic_code:nnn` 86, 87, 3073, 3076, 3076
`__enumext_keyans_default_item:n` . . 75, 2559, 2559, 2595
`\l__enumext_keyans_env_bool` 20, 2826, 2839, 2968, 3053
`__enumext_keyans_fake_item:` . . 776, 792, 2692
`\l__enumext_keyans_item_opt_tl` . 101, 97, 2392, 2405, 2411, 3790
`\l__enumext_keyans_level_h_int` . . 20, 581, 608, 2322, 3704, 3705
`\l__enumext_keyans_level_int` . . 20, 1310, 2072, 2317, 2967, 2972, 3067
`__enumext_keyans_make_label:` 33, 76, 2625, 2625, 2690
`__enumext_keyans_mini_addvspace:` 48, 84, 1214, 1214, 2997
`__enumext_keyans_mini_right_cmd:n` 50, 1312, 1335, 1335
`__enumext_keyans_mini_set_vskip:` . 47, 1152, 1152, 1216
`__enumext_keyans_multi_addvspace:` 85, 1001, 1012, 3022
`__enumext_keyans_multi_set_vskip:` 44, 1001, 1001, 1014
`__enumext_keyans_multicols_start:` . . 84, 85, 3001, 3003, 3003
`__enumext_keyans_multicols_stop:` . 85, 1339, 3028, 3028, 3052
`__enumext_keyans_parse_keys:n` 2946, 2981, 2981
`\l__enumext_keyans_pic_above_int` . 127, 3162, 3163, 3165
`\l__enumext_keyans_pic_above_skip` . . 88, 127, 3106, 3146
`__enumext_keyans_pic_arg_two:` 88, 3104, 3134, 3134
`\l__enumext_keyans_pic_below_int` . 127, 3162, 3163, 3166
`\l__enumext_keyans_pic_body_seq` . . 86–88, 127, 3071, 3111, 3170
`__enumext_keyans_pic_do:n` 88, 3111, 3113, 3154, 3154, 3158
`\l__enumext_keyans_pic_level_int` . . 20, 1302, 2076, 2273, 2312, 2347, 2425, 3122, 3123
`__enumext_keyans_pic_row:n` 88, 3156, 3159, 3159
`__enumext_keyans_pic_safe_exec:` . . 87, 3100, 3120, 3120
`__enumext_keyans_pic_skip_abs:N` . . 88, 3129, 3129, 3145
`\l__enumext_keyans_pic_width_dim` . 127, 3161, 3168
`__enumext_keyans_redefine_item:` . . 75, 2583, 2583, 2689
`__enumext_keyans_ref:` 36, 633, 651, 2691
`__enumext_keyans_ref:n` 36, 630, 633, 633
`__enumext_keyans_safe_exec:` . 2945, 2961, 2961
`__enumext_keyans_save_start_line:` . 28, 266, 266, 2969, 3127, 3709
`__enumext_keyans_show_ans:` . . 2388, 2396, 2415
`__enumext_keyans_show_item_opt:` . 2388, 2403, 2576, 3096, 3871
`__enumext_keyans_show_left:n` . 75, 2388, 2388, 2574, 3091
`__enumext_keyans_show_pos:` . . 2388, 2400, 2423
`__enumext_keyans_starred_item:n` . . 75, 2571, 2571, 2591
`__enumext_keyans_store_ref:` . . 69, 2291, 2291, 2579, 3083, 3803
`__enumext_keyans_store_ref_aux_i:` 70, 2291,

2303, 2306
 __enumext_keyans_store_ref_aux_ii: 70, 2291, 2332, 2334
 \l__enumext_keyans_tmpa_tl .. 24, 97, 2573, 2577
 __enumext_keyans_wrapper_opt:n .. 1890, 2411
 \l__enumext_label_copy_i_tl .. 2203, 2310, 2315, 2320, 2325
 \l__enumext_label_copy_v_tl 2320
 \l__enumext_label_copy_vi_tl 2315
 \l__enumext_label_copy_vii_tl 2178, 2189, 2220, 2310
 \l__enumext_label_copy_viii_tl 2325
 \l__enumext_label_copy_X_tl 143
 \l__enumext_label_fill_left_v_tl 2629
 \l__enumext_label_fill_left_X_tl 85
 \l__enumext_label_fill_right_v_tl 2636
 \l__enumext_label_fill_right_X_tl 85
 \l__enumext_label_font_style_v_tl 2630, 3095
 \l__enumext_label_font_style_vii_tl ... 3509
 \l__enumext_label_font_style_viii_tl .. 3853
 \l__enumext_label_i_tl 496
 \l__enumext_label_ii_tl 496
 \l__enumext_label_iii_tl 496
 \l__enumext_label_iv_tl 496
 __enumext_label_style:Nnn 23, 32, 416, 416, 431, 501, 548, 619, 623
 \l__enumext_label_v_tl .. 69, 70, 616, 2278, 2352, 2417, 2452, 2573, 2577, 2949, 3090, 3092
 \l__enumext_label_vi_tl . 69, 70, 616, 2275, 2349, 3090, 3092, 3096
 \l__enumext_label_vii_tl . 543, 3436, 3461, 3468
 \l__enumext_label_viii_tl 543, 3769, 3797, 3801
 \l__enumext_label_width_by_box .. 55, 412, 413
 __enumext_label_width_by_box:Nn 32, 410, 410, 415, 427, 684
 \l__enumext_labelsep_i_dim ... 2420, 2455, 3809, 3824
 \l__enumext_labelsep_v_dim 3012
 \l__enumext_labelsep_vii_dim . 3181, 3190, 3232, 3459, 3519, 3535
 \l__enumext_labelsep_viii_dim 3571, 3580, 3622, 3863, 3886
 \l__enumext_labelwidth_i_dim . 2420, 2455, 3809, 3824
 \l__enumext_labelwidth_v_dim 3012
 \l__enumext_labelwidth_vii_dim ... 3181, 3189, 3232, 3512, 3516, 3534
 \l__enumext_labelwidth_viii_dim .. 3571, 3579, 3622, 3856, 3860, 3885
 \l__enumext_leftmargin_tmp_v_bool . 88, 3136
 \l__enumext_leftmargin_tmp_X_bool 59
 \l__enumext_leftmargin_tmp_X_dim 59
 \l__enumext_leftmargin_X_dim 59
 __enumext_level: 187, 187, 525, 528, 529, 538, 540, 779, 783, 787, 854, 858, 862, 866, 950, 952, 954, 956, 989, 991, 993, 995, 999, 1039, 1042, 1061, 1070, 1076, 1081, 1085, 1096, 1100, 1101, 1106, 1142, 1146, 1319, 1325, 1372, 1374, 1376, 1379, 1386, 1388, 1390, 1393, 1980, 1988, 1992, 1996, 2241, 2244, 2245, 2515, 2516, 2520, 2521, 2522, 2529, 2531, 2535, 2536, 2539, 2545, 2546, 2601, 2604, 2606, 2613, 2614, 2615, 2618, 2621, 2751, 2753, 2794, 2799, 2800, 2801, 2803, 2807, 2812, 2813, 2814, 2816, 2832, 2845, 2852, 2863, 2865, 2868, 2869, 2871, 2876, 2883, 2886, 2888, 2890, 2891, 2892, 2893, 2896, 2902, 2907, 2913, 2916, 2918, 2924
 \l__enumext_level_h_int .. 20, 219, 241, 254, 564, 601, 1812, 1828, 2197, 2214, 3317, 3318
 \l__enumext_level_int . 79, 20, 189, 228, 240, 255, 295, 962, 1113, 1306, 1806, 2174, 2184, 2190, 2196, 2204, 2212, 2219, 2705, 2766, 2767, 2777, 2784, 2830, 2843, 2898, 2976, 3063, 3387, 3397, 3712
 __enumext_list_arg_two_i: 2671
 __enumext_list_arg_two_ii: 2671
 __enumext_list_arg_two_iii: 2671
 __enumext_list_arg_two_iv: 2671
 __enumext_list_arg_two_v: . 75, 2671, 2951, 3137
 __enumext_list_arg_two_vii: 2711, 3299
 __enumext_list_arg_two_viii: 2711, 3687
 \l__enumext_listoffset_v_dim 3014
 \l__enumext_listparindent_vii_dim 3522
 \l__enumext_listparindent_viii_dim ... 3866
 __enumext_make_label: 33, 73, 74, 76, 2609, 2609, 2700
 \l__enumext_mark_answer_sym_tl . 64, 122, 1896, 2050, 2256, 2427, 2440, 3813
 \l__enumext_mark_position_str 122, 1900, 1901, 1924, 1925, 2048
 \l__enumext_mark_ref_sym_tl .. 122, 1910, 2154, 2379
 __enumext_mini_addvspace: .. 47, 81, 1135, 1135, 2873
 __enumext_mini_addvspace_vii: 49, 1288, 1288, 3257
 __enumext_mini_addvspace_viii: 49, 1288, 1294, 3647
 __enumext_mini_env* 1029
 __enumext_mini_right_cmd:n 50, 1314, 1316, 1316
 __enumext_mini_set_vskip: . 45, 1036, 1036, 1137
 __enumext_mini_set_vskip_vii: 48, 1231, 1231, 1290
 __enumext_mini_set_vskip_viii: 48, 1231, 1253, 1296
 __enumext_minipage:w .. 30, 325, 327, 1031, 3168, 3521, 3865
 \l__enumext_minipage_active_v_bool .. 84, 85, 2995, 3020, 3033, 3041
 \g__enumext_minipage_active_vii_bool ... 90, 3268, 3273, 3285
 \l__enumext_minipage_active_vii_bool . 3253, 3264
 \g__enumext_minipage_active_viii_bool 3658, 3663, 3675
 \l__enumext_minipage_active_viii_bool 3643, 3654
 \g__enumext_minipage_active_X_bool ... 154
 \l__enumext_minipage_active_X_bool 73
 \g__enumext_minipage_after_skip 73, 1235, 1247, 3283, 3673
 \l__enumext_minipage_after_skip 45, 46, 83, 85, 73, 1052, 1067, 1087, 1103, 1118, 1124, 1130, 1144, 1154, 1163, 1166, 1178, 1196, 1207, 1223, 1255, 1268, 1282, 2933, 3050
 \g__enumext_minipage_center_vii_bool . 3277, 3286
 \g__enumext_minipage_center_viii_bool 3667, 3676
 \g__enumext_minipage_center_X_bool ... 154
 \l__enumext_minipage_hsep_v_dim ... 84, 2993

```

\l__enumext_minipage_hsep_vii_dim . . . 3251
\l__enumext_minipage_hsep_viii_dim . . . 3641
\l__enumext_minipage_left_skip 45, 84, 73, 1044,
1059, 1078, 1093, 1140, 1150, 1155, 1161, 1170, 1187,
1199, 1219, 1229, 1233, 1238, 1242, 1256, 1260, 1274,
1292, 1298
\l__enumext_minipage_left_v_dim 84, 2991, 2999
\l__enumext_minipage_left_vii_dim 3247, 3259
\l__enumext_minipage_left_viii_dim 3637, 3649
\l__enumext_minipage_left_X_dim . . . . . 73
\g__enumext_minipage_right_skip 73, 1234, 1239,
1243, 3276, 3666
\l__enumext_minipage_right_skip . 45, 73, 1048,
1063, 1083, 1098, 1156, 1162, 1174, 1192, 1203, 1257,
1264, 1278, 1326, 1343
\l__enumext_minipage_right_v_dim . . 84, 1337,
1342, 2989, 2993
\g__enumext_minipage_right_vii_dim 90, 3255,
3275, 3288
\l__enumext_minipage_right_vii_dim 90, 3245,
3250, 3256
\g__enumext_minipage_right_viii_dim . . 3645,
3665, 3678
\l__enumext_minipage_right_viii_dim . . 3635,
3640, 3646
\g__enumext_minipage_right_X_dim . . . . . 154
\g__enumext_minipage_right_X_skip . . . . . 154
\g__enumext_minipage_stat_int . 81, 84, 73, 1331,
1348, 2872, 2926, 2931, 2996, 3043, 3048
\g__enumext_miniright_code_vii_tl . 91, 3281,
3287
\g__enumext_miniright_code_viii_tl 3671, 3677
\g__enumext_miniright_code_X_tl . . . . . 154
\__enumext_multi_addvspace: . . . 43, 82, 984, 984,
2904
\__enumext_multi_set_vskip: . . 43, 948, 948, 986
\l__enumext_multicols_above_ii_skip . . . 967
\l__enumext_multicols_above_iii_skip . . . 973
\l__enumext_multicols_above_iv_skip . . . 979
\l__enumext_multicols_above_v_skip 1003, 1017,
1027
\l__enumext_multicols_above_X_skip . . . . . 67
\l__enumext_multicols_below_v_skip 1007, 1021,
3035
\l__enumext_multicols_below_X_skip . . . . . 67
\__enumext_multicols_start: 82, 2878, 2880, 2880
\__enumext_multicols_stop: 82, 1321, 2910, 2910,
2935
\__enumext_newlabel:nn 26, 31, 68, 368, 368, 2230,
2338
\l__enumext_newlabel_arg_one_tl 26, 31, 68, 70,
143, 2153, 2223, 2231, 2327, 2339, 2377
\l__enumext_newlabel_arg_two_tl 26, 31, 67, 143,
2177, 2187, 2201, 2217, 2232, 2314, 2319, 2324, 2340
\__enumext_parse_keys:n . . . 54, 2747, 2772, 2772
\__enumext_parse_keys_vii:n 54, 3294, 3325, 3325
\__enumext_parse_keys_viii:n . 3683, 3717, 3717
\__enumext_parse_series:n . . 54, 80, 1505, 1505,
2780, 3331
\__enumext_parse_store_keys:n . 80, 2788, 2792,
2792
\__enumext_parse_store_keys_vii:n . 92, 3334,
3338, 3338
\l__enumext_parsep_i_skip 965, 967, 1116, 1164
\l__enumext_parsep_ii_skip . . . 971, 973, 1122
\l__enumext_parsep_iii_skip . . . 977, 979, 1128
\l__enumext_parsep_vii_skip . . . . . 3523
\l__enumext_parsep_viii_skip . . . . . 3867
\l__enumext_partopsep_v_skip . 1019, 1023, 1190,
1194, 1201, 1205, 1221, 1225
\l__enumext_partopsep_viii_skip . . . . . 1266
\__enumext_phantomsection: 30, 332, 361, 365, 381
\__enumext_print_footnote: . . . 2474, 2497, 3541,
3892
\__enumext_print_keyans_box:NN 64, 2042, 2042,
2055, 2243, 2419, 2454, 3809, 3824
\l__enumext_print_keyans_i_tl . . . . 3943, 3972
\l__enumext_print_keyans_ii_tl . . . 3948, 3973
\l__enumext_print_keyans_iii_tl . . 3953, 3974
\l__enumext_print_keyans_iv_tl . . . 3958, 3975
\l__enumext_print_keyans_vii_tl . . 3963, 3976
\l__enumext_print_keyans_X_tl . . . . . 111
\__enumext_printkeyans:nnn 104, 3977, 3980, 3980
\__enumext_redefine_item: . 74, 2548, 2548, 2699
\l__enumext_ref_key_arg_tl 34, 37, 202, 518, 519,
532, 563, 566, 577, 583, 594, 635, 636, 647
\l__enumext_ref_the_count_tl . 34, 37, 525, 528,
531, 571, 573, 576, 588, 590, 593, 641, 643, 646
\__enumext_regex_counter_style: . . 27, 34, 197,
197, 526, 572, 589, 642
\__enumext_register_counter_style:Nn . . 400,
400, 405, 406, 407, 408, 409
\__enumext_remove_extra_parsep_vii: . . 3309,
3550, 3550
\__enumext_remove_extra_parsep_viii: . 3697,
3901, 3901
\__enumext_renew_footnote: . . . 2474, 2478, 3493,
3846
\l__enumext_renew_the_count_v_tl 644, 653, 655
\l__enumext_renew_the_count_vii_tl 574, 603,
605
\l__enumext_renew_the_count_viii_tl 591, 610,
612
\l__enumext_renew_the_count_X_tl . . . . . 37
\l__enumext_resume_active_bool 54, 56, 48, 1509,
1629
\__enumext_resume_counter: . . 55, 56, 1627, 1633,
1640
\__enumext_resume_counter:n . 54, 56, 1598, 1603,
1627, 1627, 1697, 1705
\__enumext_resume_counter_save_ans: . . 56, 57,
1627, 1638, 1670
\__enumext_resume_counter_series: 56, 57, 1627,
1636, 1653
\g__enumext_resume_int . . . 48, 1550, 1644, 1645
\__enumext_resume_last:n . . 54, 1505, 1511, 1524
\l__enumext_resume_name_tl 48, 1546, 1554, 1557,
1573, 1581, 1584, 1630, 1631, 1659, 1666
\__enumext_resume_save_counter: 54, 1537, 1537,
2940, 3381
\__enumext_resume_series:n . 55, 1469, 1594, 1594
\__enumext_resume_starred: . 57, 1470, 1691, 1691
\g__enumext_resume_vii_int . . 93, 48, 1577, 1649,
1650
\__enumext_safe_exec: . . . . . 79, 2746, 2763, 2763
\__enumext_safe_exec_vii: . . . 3293, 3314, 3314
\__enumext_safe_exec_viii: . . . 3682, 3702, 3702
\l__enumext_series_name_tl . . . . . 56

```

```

\l__enumext_series_str . 54, 80, 1467, 1507, 1515,
    1516, 1518, 1520, 1541, 1544, 1548, 1568, 1571, 1575,
    2776, 3329
\__enumext_set_error:nn . . . . . 4058, 4068, 4070
\__enumext_set_parse:n . . . . . 4041, 4058, 4058
\l__enumext_setkey_tmpa_int . . . 106, 4034, 4038
\l__enumext_setkey_tmpa_seq . . 106, 4032, 4042,
    4048, 4050, 4052, 4065
\l__enumext_setkey_tmpa_tl . . . 106, 4040, 4044
\l__enumext_setkey_tmpb_seq . . 106, 4033, 4036,
    4040, 4041
\l__enumext_setkey_tmpb_tl 106, 4060, 4062, 4063
\l__enumext_show_answer_bool . 122, 1904, 1928,
    2250, 2394, 2408, 3087, 3807
\__enumext_show_length:nnn . . 39, 205, 205, 4122,
    4123, 4124, 4125, 4126, 4127, 4128, 4129, 4130, 4131,
    4137, 4138, 4139, 4140, 4141, 4142, 4143, 4144, 4145,
    4146
\l__enumext_show_position_bool 122, 1907, 1931,
    2254, 2398, 2409, 3088, 3811
\g__enumext_standar_bool . 27, 20, 218, 221, 239,
    312, 1539, 1604, 1616, 1642, 1655, 1693, 1831, 1842
\l__enumext_standar_bool 83, 20, 2182, 2195, 2211,
    2769, 2939
\l__enumext_standar_first_level_bool . 79, 20,
    244, 1526, 1673, 1725, 1733, 1790
\__enumext_standar_ref: . . . . 34, 516, 536, 2701
\__enumext_standar_ref:n . . . . 34, 508, 516, 516
\g__enumext_standar_series_tl . 48, 1528, 1529,
    1695, 1698
\g__enumext_standard_bool . . . . . 79
\l__enumext_standard_bool . . . . . 79
\l__enumext_standard_first_level_bool . . 28
\__enumext_standard_item_vii:w 94, 3421, 3423,
    3423
\__enumext_standard_item_viii:w . . 100, 3754,
    3756, 3756
\g__enumext_starred_bool 27, 91, 93, 20, 227, 230,
    253, 313, 1566, 1609, 1620, 1647, 1662, 1701, 1811,
    1848, 2173, 2183, 2213, 2308, 2827, 2840, 3289
\l__enumext_starred_bool . 91, 93, 20, 2106, 2114,
    2198, 2239, 3322, 3380
\__enumext_starred_columns_set_vii: . . 3175,
    3175, 3302
\__enumext_starred_columns_set_viii: . 3565,
    3565, 3690
\l__enumext_starred_first_level_bool 20, 258,
    1531, 1682, 1726, 1734, 1794
\__enumext_starred_item:nn . . . 2525, 2525, 2554
\__enumext_starred_item_exec: . 101, 3799, 3799,
    3850
\__enumext_starred_item_vii:w . 94, 3420, 3439,
    3439
\__enumext_starred_item_vii_aux_i:w . . 3439,
    3444, 3447
\__enumext_starred_item_vii_aux_ii:w . 3439,
    3445, 3450, 3452
\__enumext_starred_item_vii_aux_iii:w 3439,
    3455, 3464
\__enumext_starred_item_viii:w 100, 101, 3753,
    3772, 3772
\__enumext_starred_item_viii_aux_i:w . . 101,
    3772, 3777, 3780
\__enumext_starred_item_viii_aux_ii:w . 101,
    3772, 3778, 3792, 3794
\__enumext_starred_joined_item_vii:n . 89, 94,
    3194, 3194, 3418
\__enumext_starred_joined_item_viii:n . . 97,
    100, 3584, 3584, 3751
\__enumext_starred_ref: . . . . 35, 561, 599, 2729
\__enumext_starred_ref:n . . . . 35, 555, 561, 561
\g__enumext_starred_series_tl . 48, 1533, 1534,
    1703, 1706
\__enumext_start_from:NNn 36, 658, 658, 671, 693
\l__enumext_start_i_int . . . . . 1645, 1657, 1676
\__enumext_start_item_tmp_vii: 91, 3305, 3403,
    3403
\__enumext_start_item_tmp_viii: 98, 3693, 3736,
    3736
\__enumext_start_item_vii:w . 94, 95, 3431, 3436,
    3461, 3468, 3470, 3470
\__enumext_start_item_viii:w . . 100, 3764, 3769,
    3797, 3827, 3827
\g__enumext_start_line_tl 28, 132, 246, 260, 315,
    1863
\__enumext_start_list:nn 29, 76, 88, 319, 321, 2750,
    2948, 3101, 3297, 3685
\__enumext_start_mini_vii: . 92, 3243, 3243, 3372
\__enumext_start_mini_viii: 99, 3633, 3633, 3728
\__enumext_start_store_level: . 80, 2749, 2821,
    2821
\__enumext_start_store_level_vii: . 93, 3296,
    3383, 3383
\l__enumext_start_vii_int . . . 1650, 1664, 1685
\l__enumext_start_X_int . . . . . 85, 688
\__enumext_stop_item_tmp_vii: . 91, 93, 95, 3304,
    3308, 3405, 3472
\__enumext_stop_item_tmp_viii: . 98, 100, 3692,
    3696, 3738, 3829
\__enumext_stop_item_vii: 95, 96, 3472, 3526, 3526
\__enumext_stop_item_viii: 102, 3829, 3877, 3877
\__enumext_stop_list: . . 29, 319, 322, 2759, 2958,
    3114, 3310, 3699
\__enumext_stop_mini_vii: 90, 93, 3262, 3262, 3376
\__enumext_stop_mini_viii: . 99, 3633, 3652, 3732
\__enumext_stop_store_level: . . 80, 2760, 2821,
    2850
\__enumext_stop_store_level_vii: . . 93, 3311,
    3383, 3393
\l__enumext_store_active_bool 24, 58, 80, 92, 97,
    1674, 1683, 1746, 2068, 2786, 2825, 2838, 2963, 2971,
    3059, 3118, 3332, 3385, 3395, 3711
\__enumext_store_addto_prop:n 63, 69, 1962, 1962,
    1970, 2093, 2289, 3802
\__enumext_store_addto_seq:n 63, 71, 1971, 1971,
    1975, 1982, 1996, 2004, 2013, 2031, 2039, 2157, 2382
\l__enumext_store_ans_bool . 58, 132, 1747, 1773,
    1978, 2002, 2009, 2037, 2081
\l__enumext_store_anskey_arg_tl . . 24, 66, 97,
    2099, 2108, 2110, 2116, 2124, 2127, 2137, 2142, 2145,
    2151, 2157
\__enumext_store_anskey_code:nnnn 65, 66, 2087,
    2091, 2091
\__enumext_store_anskey_show_left:n 68, 2098,
    2248, 2248
\__enumext_store_anskey_show_wrap:n 68, 2236,
    2236, 2252, 2267
\l__enumext_store_columns_break_bool . 2062,
    2105

```

```

\l__enumext_store_columns_join_int 97, 2113,
    2118
\l__enumext_store_columns_sep_vii_bool 3353
\l__enumext_store_columns_sep_vii_dim 3358,
    3362
\l__enumext_store_columns_sep_X_bool .. 111
\l__enumext_store_columns_sep_X_dim ... 111
\l__enumext_store_columns_vii_bool ... 3340
\l__enumext_store_columns_vii_int 3345, 3349
\l__enumext_store_columns_X_bool ..... 111
\l__enumext_store_columns_X_int ..... 111
\__enumext_store_internal_ref: .. 66, 67, 2096,
    2159, 2159
\l__enumext_store_item_symbol_sep_dim 2060,
    2134, 2139
\l__enumext_store_item_symbol_tl . 2058, 2125,
    2129
\l__enumext_store_keyans_item_opt_sep_-
    tl .... 1893, 2283, 2285, 2356, 2360, 3785, 3787
\l__enumext_store_keyans_item_opt_tl ... 97
\l__enumext_store_keyans_label_tl 24, 69–71,
    101, 97, 2272, 2275, 2278, 2285, 2287, 2289, 2346,
    2349, 2352, 2358, 2363, 2373, 2382, 3782, 3787, 3788,
    3801, 3802, 3804
\__enumext_store_level_close: . 63, 1976, 2000,
    2854
\__enumext_store_level_close_vii: 2007, 2035,
    3399
\__enumext_store_level_open: .. 62, 63, 80, 1976,
    1976, 2833, 2846
\__enumext_store_level_open_vii: .. 92, 2007,
    2007, 3389
\g__enumext_store_name_tl . 24, 83, 97, 297, 300,
    302, 314, 1748, 1858, 1863
\l__enumext_store_name_tl . 24, 58, 60, 97, 1560,
    1563, 1587, 1590, 1678, 1687, 1721, 1722, 1737, 1739,
    1748, 1749, 1751, 1752, 1754, 1756, 1757, 1759, 1761,
    1762, 1788, 1964, 1966, 1973, 2225, 2226, 2262, 2329,
    2330, 2433, 2446, 3819
\l__enumext_store_opt_vii_tl . 2011, 2021, 2027,
    2031, 3347, 3360
\l__enumext_store_opt_X_tl ..... 111
\l__enumext_store_ref_key_bool 66, 1913, 2094,
    2148, 2293, 2370
\l__enumext_store_upper_level_X_bool .. 111
\l__enumext_store_write_aux_file_tl 26, 68, 70,
    143, 2228, 2234, 2336, 2342
\__enumext_storing_exec: .. 58, 1719, 1740, 1744
\__enumext_storing_set:n .. 58, 1714, 1719, 1719
\l__enumext_the_counter_v_tl ..... 643
\l__enumext_the_counter_vii_tl ..... 573
\l__enumext_the_counter_viii_tl ..... 590
\l__enumext_the_counter_X_tl ..... 37
\__enumext_tmp:n 32, 36, 41, 47, 59, 66, 67, 72, 79, 84,
    85, 96, 111, 121, 146, 150, 154, 173, 771, 775, 1463,
    1474, 1710, 1718, 1765, 1783, 1883, 1918, 1919, 1936,
    2161, 2168, 2169, 2190, 2204, 2207, 2219, 2295, 2302,
    2671, 2710, 2711, 2743
\__enumext_tmp:nn 432, 453, 454, 482, 483, 495, 688,
    707, 752, 770, 828, 836, 837, 851, 916, 932, 933, 947,
    1352, 1368, 1937, 1961, 2458, 2473
\__enumext_tmp:nnnn 496, 512, 513, 514, 515, 543, 559,
    560
\__enumext_tmp:nnnnnn 708, 733, 736, 739, 741, 743,
    746, 749
\__enumext_tmp:w ..... 3926, 3929
\l__enumext_tmpa_vii_int ..... 3185, 3188
\l__enumext_tmpa_viii_int ..... 3575, 3578
\l__enumext_tmpa_X_int ..... 154
\l__enumext_topsep_v_skip 1005, 1009, 1159, 1172,
    1180, 1185, 1205, 1209, 3117, 3149
\l__enumext_topsep_vii_skip .. 1236, 1245, 1249
\l__enumext_topsep_viii_skip . 1258, 1280, 1284
\l__enumext_vspace_a_star_v_bool ..... 1401
\l__enumext_vspace_a_star_vii_bool ... 1423
\l__enumext_vspace_a_star_viii_bool ... 1434
\l__enumext_vspace_a_star_X_bool ..... 85
\__enumext_vspace_above: .. 51, 1369, 1369, 2859
\__enumext_vspace_above_v: . 52, 1397, 1397, 2987
\l__enumext_vspace_above_v_skip .. 1399, 1403,
    1405
\__enumext_vspace_above_vii: .. 52, 1419, 1419,
    3369
\l__enumext_vspace_above_vii_skip 1421, 1425,
    1427
\__enumext_vspace_above_viii: . 52, 1419, 1430,
    3726
\l__enumext_vspace_above_viii_skip 1432, 1436,
    1438
\l__enumext_vspace_b_star_v_bool ..... 1412
\l__enumext_vspace_b_star_vii_bool ... 1445
\l__enumext_vspace_b_star_viii_bool ... 1456
\l__enumext_vspace_b_star_X_bool ..... 85
\__enumext_vspace_below: .. 51, 1383, 1383, 2938
\__enumext_vspace_below_v: . 52, 1408, 1408, 3055
\l__enumext_vspace_below_v_skip .. 1410, 1414,
    1416
\__enumext_vspace_below_vii: .. 52, 1441, 1441,
    3379
\l__enumext_vspace_below_vii_skip 1443, 1447,
    1449
\__enumext_vspace_below_viii: . 52, 1441, 1452,
    3734
\l__enumext_vspace_below_viii_skip 1454, 1458,
    1460
\__enumext_widest_from:nnnn .. 37, 672, 672, 687,
    699
\g__enumext_widest_label_tl 23, 32, 55, 420, 424,
    428
\l__enumext_wrap_label_opt_v_bool .... 2567
\l__enumext_wrap_label_opt_vii_bool 94, 3430
\l__enumext_wrap_label_opt_viii_bool .. 100,
    3763
\l__enumext_wrap_label_opt_X_bool ..... 85
\l__enumext_wrap_label_v_bool 2563, 2567, 2575,
    2631
\l__enumext_wrap_label_vii_bool 94, 3429, 3434,
    3442, 3510
\l__enumext_wrap_label_viii_bool . 100, 3762,
    3767, 3775, 3854
\l__enumext_wrap_label_X_bool ..... 85
\__enumext_wrapper_label_v:n ..... 2633, 3096
\__enumext_wrapper_label_vii:n ..... 3513
\__enumext_wrapper_label_viii:n ..... 3857
\__enumext_zero_parsep: ... 46, 1056, 1111, 1111
enumext* ..... 5, 3291
enumXi ..... 392
enumXii ..... 392
enumXiii ..... 392

```


enumXiv 392

enumXv 392

enumXvi 392

enumXvii 392

enumXviii 392

Environments provide by enumext:

enumext* 22, 23, 25-27, 29, 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, 29, 31-40, 42-48, 50, 51, 53-55, 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, 33, 35, 50, 58, 59, 61, 63, 69-71, 86-88, 108

keyans 22-25, 27, 28, 31-33, 35, 37-42, 44, 47, 48, 50-52, 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, 30, 42, 44, 45, 86-89, 95, 96, 102

multicols 42-45, 50, 82-85

exp commands:

\exp_after:wN 3929

\exp_args:Ne 2783, 3917

\exp_not:N . 45, 423, 531, 576, 593, 646, 785, 799, 800, 811, 812, 823, 824, 2153, 2259, 2260, 2375, 2430, 2431, 2443, 2444, 3816, 3817, 3926

\exp_not:n 248, 262, 274, 281, 288, 531, 532, 576, 577, 593, 594, 646, 647, 786, 1491, 1503, 1945, 1952, 2118, 2129, 2139, 2153, 2154, 2231, 2339, 2377, 2379, 2803, 2816, 3349, 3362

F

\fbox 1888

file commands:

\file_input_stop: 4273

first 837

font 432

\footnote 73

\footnote 73, 2482

\footnotemark 2492

\footnotesize 2260, 2431, 2444, 3817

\footnotetext 2476

G

\getkeyans 14, 103, 3915

group commands:

\group_begin: . . 2080, 2258, 2429, 2442, 3489, 3508, 3815, 3842, 3852, 3937, 3971

\group_end: 2089, 2265, 2436, 2449, 3518, 3530, 3822, 3862, 3881, 3939, 3978

H

\hbadness 3537, 3888

hbox commands:

\hbox_set:Nn 412

\hfill 462, 466, 471, 472, 1323, 1341, 2153, 2375, 3267, 3657

hook commands:

\hook_gput_code:nnn 9, 181, 185, 330

\hook_gset_rule:nnnn 331

\hspace 3548, 3899

\hyperlink 67, 71

\hyperlink 2153, 2375

\hypertarget 30

\hypertarget 360

I

\IfHyperBoolean 338

\IfPackageLoadedTF 11, 334, 348

\ignorespaces 788

\inputlineno 248, 262, 274, 281, 288

int commands:

\int_add:Nn 3227, 3617

\int_case:nn 962, 1113, 1806, 1828

\int_compare:nNnTF . . 295, 564, 581, 601, 608, 1038, 1157, 1302, 1306, 1310, 1855, 1868, 1874, 2072, 2076, 2273, 2312, 2317, 2322, 2347, 2425, 2767, 2777, 2830, 2843, 2882, 2898, 2912, 2926, 2972, 2976, 3005, 3030, 3043, 3063, 3067, 3123, 3197, 3207, 3223, 3318, 3387, 3397, 3543, 3552, 3587, 3597, 3613, 3705, 3712, 3894, 3903, 4038

\int_compare_p:nNn . . . 219, 228, 240, 241, 254, 255, 1812, 2174, 2184, 2196, 2197, 2212, 2214

\int_decr:N 3226, 3616

\int_eval:n 1966, 2226, 2260, 2330, 2431, 2444, 2686, 2728, 3215, 3605, 3817

\int_from_alph:n 666, 680

\int_from_roman:n 668, 682

\int_gadd:Nn 3228, 3618

\int_gdecr:N 1815, 1819, 1822, 1825, 1833

\int_gincr:N 1644, 1649, 2085, 2385, 2513, 2543, 2581, 2872, 2996, 3085, 3407, 3485, 3740, 3806

\int_gset:Nn 2490

\int_gset_eq:NN 1543, 1550, 1556, 1562, 1570, 1577, 1583, 1589, 2487

\int_gzero:N . 309, 310, 1331, 1348, 1880, 2931, 3048, 3561, 3912

\int_if_exist:NTF 1518, 1554, 1560, 1581, 1587, 1759

\int_incr:N 2766, 2967, 3122, 3317, 3406, 3704, 3739

\int_mod:nn 3554, 3905

\int_new:N . 20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 108, 116, 129, 130, 138, 139, 140, 151, 157, 158, 159, 160, 161, 1520, 1762

\int_set:Nn 662, 666, 668, 1657, 1664, 1676, 1685, 1942, 2113, 3162, 3163, 3185, 3196, 3202, 3218, 3537, 3575, 3586, 3592, 3608, 3888, 4034

\int_set_eq:NN . 1645, 1650, 2798, 3225, 3344, 3615

\int_step_function:nnN 2190, 2204, 2219

\int_step_inline:nnn 3164, 4061

\int_to_roman:n 189, 2170, 2208

\int_use:N . 1039, 1659, 1666, 1678, 1687, 2686, 2705, 2728, 2784, 2883, 2892, 2907, 2913, 3200, 3201, 3213, 3590, 3591, 3603

\int_zero:N 3546, 3897

\c_one_int . 3185, 3204, 3210, 3216, 3220, 3223, 3575, 3594, 3600, 3606, 3610, 3613

\c_zero_int 2174, 2184, 2196, 2197, 2212, 2214, 3387, 3397, 3557, 3908

\item 29, 40-42, 64, 73, 86, 88, 89, 91, 98

\item 73, 75, 93, 95, 100, 102, 323, 1984, 1990, 2015, 2023, 2110, 2349, 2352, 2550, 2585, 3303, 3305, 3691, 3693, 3804

\item* 6, 12, 61, 2583

item-pos* 2458

item-sym* 2458

\itemindent 23, 77

\itemindent 77

itemindent 752

\itemsep	87, 88
\itemsep	3138, 3144
\itemwidth	3192, 3236, 3240, 3582, 3626, 3630
K	
keyans	12, 2943
keyans*	12, 3680
keyanspic	13, 3098
Keys for environments provide by enumext:	
above*	24, 51, 52
above	24, 51, 52, 81, 84, 92, 99
after	40, 41, 83, 86, 93, 99
align	24, 33, 76, 95
before*	40, 41, 81, 92, 99
before	40, 41, 84
below*	24, 51, 52
below	24, 51, 52, 83, 86, 93, 99
check-ans	24, 25, 27, 28, 58-61, 65, 71, 73, 74, 81, 83, 96, 108
columns-sep*	25, 62, 80, 92
columns-sep	42, 63, 80, 82, 85, 92
columns*	25, 62, 80, 92
columns	24, 42, 45, 51, 63, 80, 82, 85, 92
first	40-42, 95
font	32, 76, 95
item-pos*	65, 66, 72
item-sym*	23, 65, 66, 72, 74
item*-sep	74
itemindent	24, 38, 39, 75, 95
itemsep	37, 78
labelsep	32, 74, 77, 95
labelwidth	31-35, 37, 77
label	23, 31-33, 36, 37, 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, 42, 45, 50, 51, 73, 81, 84, 90, 92, 98, 99
mini-sep	24, 42, 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, 53, 54, 57, 58, 83
resume	23, 53-58, 83, 93
rightmargin	38
save-ans	24, 53-58, 60, 63, 65, 69, 70, 75, 80, 84, 86, 93, 100, 101, 103, 104, 107
save-key	25, 53, 80
save-ref	26, 31, 61, 66, 67, 69, 71, 75, 101
save-sep	61, 101
series	23, 53-57, 80, 83
show-ans	25, 61, 62, 64, 66, 68, 75, 101
show-length	27, 39, 107
show-pos	25, 61, 62, 64, 66, 68, 71, 75, 101

start	24, 27, 36, 37, 53
store-brk	65, 66
topsep	37
widest	23, 27, 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	434, 456, 485, 498, 545, 616, 690, 710, 754, 773, 830, 839, 918, 935, 1354, 1465, 1712, 1767, 1885, 1921, 1939, 2056, 2460, 3941, 4004
\l_keys_key_str	4107
\keys_set:nn	448, 942, 1359, 1364, 1606, 1611, 1698, 1706, 2102, 2779, 2783, 2983, 3330, 3721, 4006, 4007, 4008, 4009, 4010, 4011, 4012, 4013, 4014, 4015, 4016, 4017, 4055
keyval commands:	
\keyval_parse:NNn	1479

L	
label	496, 543, 616
Labels provide by enumext:	
\Alph*	31, 32
\Roman*	31, 32
\alph*	31, 32
\arabic*	27, 31, 32
\roman*	31, 32
\labelsep	88
\labelsep	3139, 3142
labelsep	432
\labelwidth	32, 88
\labelwidth	3139, 3140
labelwidth	432
\leftmargin	23, 77
\leftmargin	77, 3139
legacy commands:	
\legacy_if:nTF	3473, 3476, 3830, 3833
\legacy_if_gset_false:n	1032
\legacy_if_set_false:n	3475, 3832
\legacy_if_set_true:n	3435, 3460, 3467, 3480, 3768, 3796, 3837
\linewidth	81, 84
\linewidth	2867, 2993, 3161, 3188, 3249, 3578, 3639
\list	29
\list	321
list-indent	752
list-offset	752
\listparindent	3141
listparindent	752
\lrbox	3490, 3843

M	
\makebox	89
\makebox ..	2046, 2048, 2605, 3504, 3512, 3516, 3856, 3860
\makelabel	73, 76, 89
\makelabel	76, 2611, 2627
\makesavenoteenv	354
mark-ans	1883
mark-pos	1883, 1919
mark-ref	1883
mini-env	916
mini-sep	916
\minipage	30
\minipage	327

`\miniright` 10, 49, 1300, 2929, 3046
`\miniright*` 10
mode commands:
 `\mode_if_vertical:TF` 987, 1015, 1138, 1217
 `\mode_leave_vertical:` .. 785, 799, 811, 823, 2015,
 2023, 2044, 2603, 3502
msg commands:
 `\msg_error:nn` .. 2974, 2978, 3065, 3125, 3320, 3707,
 3714, 4018
 `\msg_error:nnn` 521, 568, 585, 638, 1304, 1308, 1333,
 1350, 1618, 1622, 1728, 3931, 3936, 4001, 4071
 `\msg_error:nnnn` 1792, 1796, 2070, 2074, 2078, 2965,
 3061, 3069
 `\msg_fatal:nn` 2768
 `\msg_fatal:nnn` 386
 `\msg_info:nnn` 13, 16, 336, 350
 `\msg_line_context:` .. 4091, 4095, 4099, 4111, 4116,
 4121, 4136, 4151, 4155, 4159, 4164, 4168, 4173, 4177,
 4201, 4206, 4211, 4216, 4221, 4225, 4230, 4235, 4239,
 4244, 4248, 4253, 4258, 4263, 4267, 4271
 `\msg_log:nnn` 1751, 1756, 1761
 `\msg_log:nnnn` 301, 1738
 `\msg_new:nnn` 4072, 4076, 4080, 4084, 4089, 4093, 4097,
 4101, 4105, 4109, 4114, 4119, 4134, 4149, 4153, 4157,
 4162, 4166, 4171, 4175, 4179, 4186, 4191, 4195, 4199,
 4204, 4209, 4214, 4219, 4223, 4228, 4233, 4237, 4242,
 4246, 4251, 4256, 4261, 4265, 4269
 `\msg_note:nnn` 299, 1736
 `\msg_term:nnn` 1858
 `\msg_term:nnnn` 2695, 2705, 2734, 2739
 `\msg_warning:nn` 2928, 3045
 `\msg_warning:nnnn` 1861, 1871, 1877, 2643, 2648, 3199,
 3212, 3589, 3602
`\multicolsep` 82, 85
`\multicolsep` 2897, 3018

N

`\NeedsTeXFormat` 3
`\newcounter` 389
`\NewDocumentCommand` 1300, 2066, 3057, 3915, 3969, 4025
`\NewDocumentEnvironment` . 2744, 2943, 3098, 3291, 3680
`\newlabel` 31
`\newlabel` 372
`no-store` 1765
`\noindent` 91, 98
`\noindent` . 2874, 2998, 3258, 3304, 3545, 3648, 3692, 3896
`\nointerlineskip` 2874, 2998, 3258, 3648
`noitemsep` 708
`\nopagebreak` 998, 1026, 1149, 1228, 1291, 1297
`\normalfont` 2259, 2430, 2443, 3816
`nosep` 708

P

Packages:
 enumext 22, 33, 58, 77, 86, 106
 enumitem 31
 expl3 89
 footnotehyper 30
 hyperref 25, 26, 30, 31, 67, 71, 95, 106
 lua-visual-debug 45
 multicol 22, 106
 shortlst 89
`\par` 998, 1026, 1149, 1228, 1291, 1297, 1326, 1343, 2238, 2918,
 2933, 3035, 3050, 3173, 3276, 3283, 3545, 3559, 3666,
 3673, 3896, 3910
`\parindent` 3522, 3866

`\parsep` 43, 46, 87, 88
`\parsep` 2016, 2024, 2725, 3138, 3145, 3150
`parsep` 708
`\parskip` 3523, 3867
`\partopsep` 88
`\partopsep` 2726, 3143
`partopsep` 708
peek commands:
 `\peek_meaning:NTF` 3412, 3426, 3443, 3454, 3745, 3759,
 3776
 `\peek_meaning_remove:NTF` 3419, 3752
 `\peek_remove_spaces:n` 2589
`\phantomsection` 30
`\phantomsection` 361
prg commands:
 `\prg_do_nothing:` 365
 `\prg_new_protected_conditional:Npnn` ... 191
 `\prg_replicate:nn` 208, 4184
 `\prg_return_false:` 195
 `\prg_return_true:` 194
`\printkeyans` 14, 104, 3969
prop commands:
 `\prop_count:N` 1966, 2226, 2262, 2330, 2433, 2446, 3819
 `\prop_gput_if_not_in:Nnn` 1964
 `\prop_if_exist:NTF` 1749, 3935
 `\prop_item:Nn` 3938
 `\prop_new:N` 1752
`\ProvidesExplPackage` 4

R

`\raggedcolumns` 2906, 3024
`\ref` 67, 69
`ref` 496, 543, 616
`\refstepcounter` 3482, 3839
regex commands:
 `\regex_match:nnTF` 193, 665, 667, 679, 681, 2796, 2809,
 3342, 3355
 `\regex_replace_once:nnN` 201
`\renewcommand` 531, 576, 593, 646
`\RenewDocumentCommand` ... 2482, 2550, 2585, 2611, 2627
`\RequirePackage` 17
`resume` 1463
`resume*` 1463
`rightmargin` 752
`\Roman` 32, 36, 37
`\Roman` 408
`\roman` 32, 36, 37
`\roman` 409, 514, 3957

S

`save-ans` 1710
`save-ref` 1883
`save-sep` 1883
scan commands:
 `\scan_stop:` 88, 3152, 3303, 3691, 3926, 3929
seq commands:
 `\seq_clear:N` 4032
 `\seq_const_from_clist:Nn` 4020
 `\seq_count:N` 3111, 4036
 `\seq_gclear:N` 2480, 2481
 `\seq_gput_right:Nn` 1973, 2493, 2494
 `\seq_if_empty:NTF` 2499, 3984, 4050
 `\seq_if_exist:NTF` 1754, 3982
 `\seq_item:Nn` 3170
 `\seq_map_function:NN` 4041

<code>\seq_map_inline:Nn</code> ..	3990, 3995, 4029, 4051, 4052
<code>\seq_map_pairwise_function:NNN</code>	2501
<code>\seq_new:N</code>	109, 110, 127, 152, 153, 1757
<code>\seq_pop_left:NN</code>	4040
<code>\seq_put_right:Nn</code>	3071, 4048, 4065
<code>\seq_set_from_clist:Nn</code>	4033
<code>\seq_set_map_e:NNn</code>	4042
<code>\seq_show:N</code>	3986
<code>series</code>	1463
<code>\setcounter</code>	676, 680, 682, 2686, 2728, 3116
<code>\setenumext</code> .	6–9, 105, 3945, 3950, 3955, 3960, 3965, 4025
<code>\setlength</code>	2017, 2025
<code>show-ans</code>	1883, 1919
<code>show-length</code>	828
<code>show-pos</code>	1919
skip commands:	
<code>\skip_add:Nn</code>	967, 973, 979, 989, 993, 1017, 1021, 1118, 1124, 1130, 1140, 1144, 1166, 1219, 1223, 3138
<code>\skip_eval:n</code>	2016, 2024
<code>\skip_gset:Nn</code>	1239, 1243, 1247
<code>\skip_gzero_new:N</code>	1234, 1235
<code>\skip_horizontal:N</code>	800, 812, 824, 3505, 3519, 3863
<code>\skip_horizontal:n</code> ...	786, 2045, 2053, 2604, 2606, 3503, 3871
<code>\skip_if_eq:nnTF</code> .	965, 971, 977, 1041, 1075, 1116, 1122, 1128, 1159, 1164, 1185, 1236, 1258, 1371, 1385, 1399, 1410, 1421, 1432, 1443, 1454
<code>\skip_new:N</code>	69, 70, 74, 75, 76, 77, 78, 131, 171
<code>\skip_set:Nn</code> .	950, 954, 1003, 1007, 1044, 1048, 1052, 1059, 1063, 1067, 1078, 1083, 1087, 1093, 1098, 1103, 1161, 1162, 1163, 1170, 1174, 1178, 1187, 1192, 1196, 1199, 1203, 1207, 1238, 1242, 1260, 1264, 1268, 1274, 1278, 1282, 3132, 3146
<code>\skip_set_eq:NN</code>	2684, 2724, 2725, 3522, 3523, 3866, 3867
<code>\skip_use:N</code>	952, 956, 991, 995, 999, 1019, 1023, 1042, 1061, 1070, 1076, 1081, 1085, 1096, 1100, 1101, 1106, 1142, 1146, 1172, 1372, 1376, 1379, 1386, 1390, 1393, 2918
<code>\skip_zero:N</code>	2726, 2897, 3018, 3143, 3144
<code>\skip_zero_new:N</code>	1154, 1155, 1156, 1233, 1255, 1256, 1257
<code>\c_zero_skip</code> .	965, 971, 977, 1042, 1076, 1116, 1122, 1128, 1159, 1164, 1185, 1236, 1258, 1372, 1386, 1399, 1410, 1421, 1432, 1443, 1454
<code>\small</code>	3947, 3952, 3957, 3962, 3967
<code>\star</code>	2464
<code>start</code>	688
<code>\stepcounter</code>	2486, 3078
str commands:	
<code>\c_backslash_str</code>	4091, 4095, 4099, 4193, 4197, 4206, 4207, 4211, 4212, 4216, 4217, 4248, 4249, 4253, 4258, 4259
<code>\c_colon_str</code>	2225, 2329, 3926
<code>\str_case:nn</code>	213, 268
<code>\str_case:nnTF</code>	1486, 1495
<code>\str_clear:N</code>	2776, 3329
<code>\str_count:n</code>	208, 4184
<code>\str_if_empty:NTF</code>	1507, 1548, 1575
<code>\str_if_eq:nnTF</code>	2687, 2730
<code>\str_if_in:nnTF</code>	3922
<code>\str_new:N</code>	126, 166
<code>\str_set:Nn</code> ...	488, 489, 490, 1900, 1901, 1924, 1925
<code>\string</code>	354
<code>\strutbox</code> .	1046, 1050, 1054, 1065, 1069, 1080, 1089, 1095,

1105, 1118, 1124, 1130, 1161, 1162, 1163, 1166, 1176, 1180, 1189, 1196, 1201, 1209, 1238, 1239, 1242, 1249, 1262, 1270, 1276, 1284, 3148
--

T

TeX and LaTeX 2_ε commands:

<code>\@auxout</code>	370
<code>\@currentenv</code>	213, 268
<code>\protected@write</code>	370

text commands:

<code>\text_expand:n</code>	3918
<code>\textasteriskcentered</code>	1897, 1911
<code>\thepage</code>	376

tl commands:

<code>\c_space_tl</code>	2411, 4121, 4136
<code>\tl_clear:N</code> ..	461, 467, 1881, 2099, 2272, 2346, 3782
<code>\tl_clear_new:N</code>	418
<code>\tl_const:Nn</code>	37, 402
<code>\tl_gclear:N</code> .	314, 315, 316, 1528, 1533, 2622, 3287, 3506, 3677
<code>\tl_gclear_new:N</code>	1515
<code>\tl_gput_right:Nn</code>	403
<code>\tl_greplace_all:Nnn</code>	424
<code>\tl_gset:Nn</code>	245, 246, 259, 260, 1516, 1529, 1534, 1748, 3449
<code>\tl_gset_eq:NN</code>	420, 2531, 3499
<code>\tl_if_blank:NTF</code>	3497
<code>\tl_if_empty:NTF</code> .	297, 519, 538, 566, 583, 603, 610, 636, 653, 1541, 1546, 1568, 1573, 1631, 1695, 1703, 1722, 1788, 1980, 2011, 2125, 2283, 2356, 2405, 2601, 3785, 4063
<code>\tl_if_empty:NTF</code>	1596
<code>\tl_if_exist:NTF</code>	1601
<code>\tl_if_novalue:NTF</code> ..	2100, 2111, 2280, 2354, 2390, 2484, 2509, 2527, 2532, 2561, 2774, 3109, 3327, 3719, 3783, 4027
<code>\tl_map_inline:Nn</code>	199, 421
<code>\tl_new:N</code>	34, 39, 40, 43, 44, 50, 52, 53, 54, 56, 57, 90, 91, 92, 98, 99, 100, 102, 103, 104, 105, 106, 107, 113, 114, 124, 125, 135, 136, 137, 143, 144, 145, 148, 165, 168
<code>\tl_put_left:Nn</code>	1988, 2021, 2108, 2417, 2452, 3801, 3804
<code>\tl_put_right:Nn</code>	419, 529, 574, 591, 644, 1943, 1950, 1992, 2027, 2110, 2116, 2124, 2127, 2137, 2142, 2145, 2151, 2177, 2187, 2201, 2217, 2223, 2228, 2275, 2278, 2285, 2287, 2314, 2319, 2324, 2327, 2336, 2349, 2352, 2358, 2363, 2373, 2801, 2814, 3347, 3360, 3787, 3788, 3943, 3948, 3953, 3958, 3963
<code>\tl_remove_all:Nn</code>	4062
<code>\tl_remove_once:Nn</code>	2165, 2299
<code>\tl_replace_all:Nnn</code>	423
<code>\tl_reverse:N</code>	2164, 2166, 2298, 2300
<code>\tl_set:Nn</code> .	45, 272, 279, 286, 388, 462, 466, 471, 472, 518, 563, 635, 783, 797, 809, 821, 1630, 1721, 2256, 2392, 2427, 2440, 2529, 3790, 3813, 4060
<code>\tl_set_eq:NN</code>	429, 524, 527, 571, 573, 588, 590, 641, 643, 2163, 2297, 2310, 2573, 2577, 3090, 3092
<code>\tl_to_str:n</code>	1601, 1607, 1612, 3918
<code>\tl_trim_spaces:n</code>	419, 4048, 4060, 4066
<code>\tl_use:N</code> .	425, 428, 540, 605, 612, 655, 854, 858, 862, 866, 870, 874, 878, 882, 886, 890, 894, 898, 902, 906, 910, 914, 2050, 2170, 2178, 2189, 2203, 2208, 2220, 2516, 2522, 2546, 2564, 2568, 2576, 2613, 2614, 2621, 2629, 2630, 2636, 2751, 2949, 3095, 3281, 3509, 3520, 3524, 3671, 3853, 3864, 3870, 3874, 3972, 3973, 3974, 3975, 3976, 4044

token commands:

 \token_to_str:N 372

\topsep 2017, 2025

topsep 708

\typeout 340, 343, 353, 354

U

\u 202

use commands:

 \use:N 209, 2618, 2753

 \use:n 1477, 3924

 \use_none:nn 364

\usecounter 2685, 2727

V

\value 1544, 1550, 1557, 1563, 1571, 1577, 1584, 1590

\vspace 1033, 1376, 1379, 1390, 1393, 1403, 1405, 1414, 1416, 1425, 1427, 1436, 1438, 1447, 1449, 1458, 1460, 2016, 2024, 3106, 3117, 3560, 3911

W

widest 688

wrap-ans 1883

wrap-label 432

wrap-label* 432

wrap-opt 1883