

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-08*

©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 check answers	10
1.3.1	Internal counters	3	4.4	The command <code>\anskey</code>	10
1.3.2	Support for <code>multicol</code>	4	4.5	The environment <code>keyans</code>	10
1.3.3	Support for <code>minipage</code>	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	11
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	12
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	12
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	12
3	The command <code>\setenumext</code>	5	5	Full examples	13
3.1	Keys for <code>label</code> and <code>ref</code>	6	6	The way of non-enumerated lists	16
3.2	Keys for spaces	6	7	References	18
3.2.1	Vertical spaces	7	8	Change history	18
3.2.2	Horizontal spaces	7	9	Index of Documentation	19
3.3	Keys for add code	8	10	Implementation	21
3.4	Keys for start and resume	8	11	Index of Implementation	102
3.5	Keys for <code>multicols</code>	8			
3.6	Keys for <code>minipage</code>	8			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

Motivation and acknowledgments

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

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

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

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

*This file describes a documentation for v1.0, last revised 2024-05-08.

†E-mail: pablgonz@educarchile.cl.

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp), version 1.3 or later (<https://www.latex-project.org/lppl.txt>). The software has the status “maintained”.

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

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

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

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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

(A) value

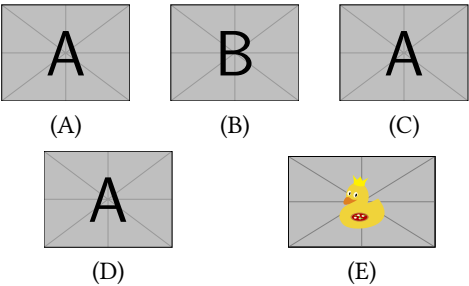
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
-

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

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

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

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

1.1 Description and usage

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

1.3 User interface

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

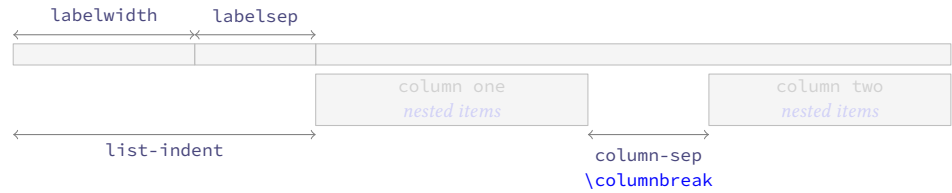


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 `\mini-right` (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 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 `\dimexpr` or `\dimeval` 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 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 `ℒTEX` 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 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`.

`resume ⟨value forbidden⟩` default: *not used*

Sets the *start* to value from the previous of the counter defined by `label` key for the “*first level*”. This `⟨key⟩` does not receive an argument. The `⟨key⟩` can be overwritten using the `start` key. If the `save-ans` key is present and `{⟨store name⟩}` exist, the numbering will continue according to this key. This key is “*only*” available for the “*first level*” of `enumext`.

3.5 Keys for multicol

`columns = {⟨integer⟩}` default: `1`

Set the *number of columns* to be used by the `multicol` environment within the environment. The value must be a positive integer less than or equal to `10`.

`columns-sep = {⟨rigid length⟩}` default: *by level*

Set the *space between columns* used by the `multicol` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

- The `\footnote{⟨text⟩}` command in the nested levels of `multicol` will not work as expected, prefer the use of `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key.

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 the command `\centering`.

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

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

4 The storage system

The entire mechanism for “storing content” it is activated according to `save-ans` key on the “first level” of `enumext` environment. Only when this *key* is “active” the `\anskey` command and the environments `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` environment, `\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.

`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.

`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 = { $\langle true | false \rangle$ }

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 $\langle store\ name \rangle$ 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.

4.2 Keys for internal label and ref

save-ref = { $\langle true | false \rangle$ }

default: *false*

Activates the internal “*label and ref*” mechanism for referencing “*stored content*” in $\langle store\ name \rangle$ set by `save-ans` key. To reference the location of the “*stored content*” within the environment you must use `\ref{ $\langle store\ name \rangle$: $\langle position \rangle$ }`, where $\langle position \rangle$ corresponds to the position occupied by the “*stored content*” in the $\langle store\ name \rangle$ 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 = { $\langle symbol \rangle$ }

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 check answers

check-ans = { $\langle true | false \rangle$ }

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 $\langle value\ forbidden \rangle$

default: *not used*

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

4.4 The command \anskey

`\anskey` `\anskey{ $\langle content \rangle$ }`

The `\anskey` command takes a mandatory argument and is triggered by `save-ans` key. The “*content*” are “*stored*” in $\langle store\ name \rangle$ 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

- | | |
|--|--|
| <p>★ 1. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* first answer</p> <p>2. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">(a) Question.</p> <p style="margin-left: 40px;">* second answer</p> | <p>3. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* third answer</p> <p>4. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* fourth answer</p> |
|--|--|

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

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

The `keyans` 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}[ $\langle key = val \rangle$ ]
    \item  $\langle item\ content \rangle$ 
    \item [ $\langle custom \rangle$ ]  $\langle item\ content \rangle$ 
  \end{keyans}
\end{enumext}
```

```
\item* <item content>
\item* [<content>] <item content>
\end{keyans}
\end{enumext}
```

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

4.5.1 The `\item*` in `keyans`

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

The `\item*` and `\item* [<content>]` command store the current `<label>` set by `label` key next to the `<content>` (if it is present) in `<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.

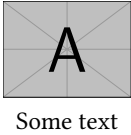
Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans]
  \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



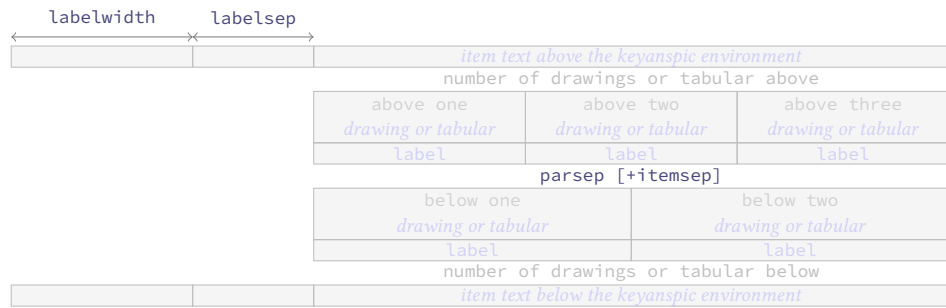
Some text

4.6 The environment `keyanspic`

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

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

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.

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

4.6.1 The command `\anspic`

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

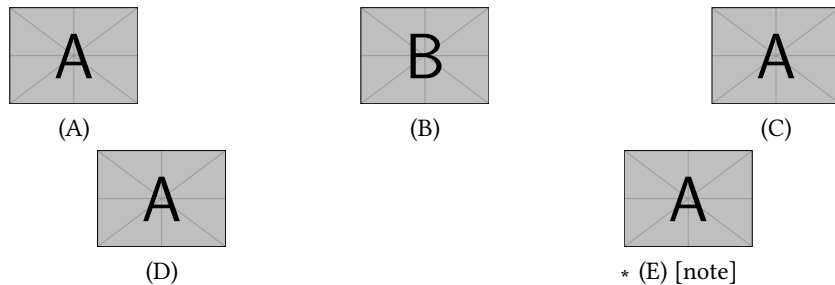
The `\anspic` command take three arguments, the *starred version* ‘`*`’ store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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,nosep,save-ref]
  \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) ~~ETEX~~e 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.
- A $1 \text{ \AA} = 1 \times 10^5$ fm.

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

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

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

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

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

B

C

D

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


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

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

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

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 $1,00 \times 10^2 \text{ m/s}$ espressa in km/h è:
- A

B

C

D

36 km/h.

360 km/h.

$27,8 \text{ km/h.}$

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

B

C

D

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

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

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

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

B

C

D

36 km/h.

360 km/h.

$27,8 \text{ km/h.}$

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

B

C

D

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

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

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

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

1. B
2. A
3. B
4. A
- *

*

*

*

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value
- B

 correct
- C

 value
- D

 value
2. Second type of questions
- I. $2\alpha + 2\delta = 90^\circ$
- II. $\alpha = \delta$
- III. $\angle EDF = 45^\circ$
- A

 I only
- B

 II only
- C

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

 value
- B

 value
- C

 value
4. Question with image and label below:



A



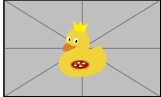
D



B



C



E



5. Question with image on left side:
- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value

- D

 I and III only
- E

 I, II, and III

- D

 value
- E

 value

Test keys

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

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

*

*

*

Example 4

A “simple worksheet” using ducks :) 📄.



Factor $x^2 - 2x + 1$



Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)



True False

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



Related to Linux

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

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

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

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

*

*

*

*

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

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

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

Fake description environment

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

long 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 2024-05-08 – First public release.

9 Index of Documentation

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

C

Document class:

article2

book2

exam3

letter2

report2

\columnbreak5

\columnsep9

Commands provide by enumext:

\anskey4, 10, 11

\anspic*4, 10–13

\anspic10, 12, 13

\getkeyans4, 11, 13

\item*4–7, 10–12

\item6, 7, 9–11

\miniright4, 5, 10

\printkeyans4, 6, 11, 13

\setenumext4, 6, 7, 10, 12, 14

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

enumext*4, 5, 10

enumext4–6, 9–14, 17

keyans*4, 5, 10

keyanspic4, 7, 10–13, 17

keyans4–12, 17

Environments:

enumerate1, 3, 4, 6, 19

list4, 9, 19

minipage3–5, 9, 10, 19

multicols3, 5, 9

I

\item4, 5

\itemsep8

K

Keys for environments provide by enumext:

above*8

above8

after9, 10

align7, 18

before*9

before9

below*8

below8

check-ans11

columns-sep5, 9

columns5, 8, 9

first9

font7

item-pos*6

item-sym*6

itemindent8

itemsep8, 12

labelsep4, 6–10, 18

labelwidth4, 6, 7, 9, 10, 18

label7, 9, 10, 12, 13, 17, 18

list-indent4, 8, 9

list-offset4, 8, 18

listparindent8

mark-ans10

mark-pos10

mark-ref11

mini-env5, 8, 9

mini-sep5, 9

miniright*10

miniright10

no-store11

noitemsep8

nosep8, 17

parsep8, 12

partopsep8

ref5, 7

resume9

rightmargin8

save-ans5, 9–13

save-ref5, 7, 11, 13

save-sep10

show-ans10

show-length7

show-pos10, 11, 13

start9

topsep8

widest7

wrap-ans10

wrap-label*7, 18

wrap-label7

wrap-opt10

L

\label5

Labels provide by enumext:

\Alph*7, 12

\Roman*7

\alph*7

\arabic*7

\roman*7

\labelsep4, 7

\labelwidth4, 7

\linewidth9

\listparindent8

P

Packages:

enumerate18

enumext1–4, 13, 18, 19

enumitem4, 5, 9, 18, 19

footnotehyper5

hyperref5, 11, 19

l3prop1, 19

l3seq1, 19

multicol	1, 2, 5, 19	\ref	5
xsim	3	\rightmargin	8
\parsep	8		
\partopsep	8		
R		T	
\raggedcolumns	5	\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   {2024-05-08}
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 \l__enumext_standar_bool
28 \bool_new:N \g__enumext_standar_bool
29 \bool_new:N \l__enumext_keyans_env_bool

```

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

```

\l__enumext_counter_i_tl
\l__enumext_counter_ii_tl
\l__enumext_counter_iii_tl
\l__enumext_counter_iv_tl
\l__enumext_counter_v_tl
\l__enumext_counter_vi_tl
\l__enumext_counter_vii_tl
\l__enumext_counter_viii_tl

```

Variables to store the “name of the counters” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment. The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments. The initial values of these variables are set by the function `__enumext_define_counters:Nn` and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§10.8).

```

30 \cs_set_protected:Npn \__enumext_tmp:n #1
31 {
32   \tl_new:c { l__enumext_counter_#1_tl }
33 }
34 \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.)

```

\l__enumext_resume_bool
\g__enumext_resume_int
\l__enumext_resume_vii_bool
\g__enumext_resume_vii_int
\g__enumext_item_symbol_tl

```

The boolean variable `\l__enumext_resume_bool` is used by `resume` key, the value from which the environment’s will start is stored in the integer variable `\g__enumext_resume_int` (§10.21). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.26).

```

35 \bool_new:N \l__enumext_resume_bool
36 \int_new:N \g__enumext_resume_int
37 \bool_new:N \l__enumext_resume_vii_bool
38 \int_new:N \g__enumext_resume_vii_int
39 \tl_new:N \g__enumext_item_symbol_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default `<label style>` and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§10.12) and `label` (§10.10) keys.

```

40 \dim_new:N \l__enumext_current_widest_dim
41 \tl_new:N \g__enumext_counter_styles_tl
42 \tl_new:N \g__enumext_widest_label_tl
43 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§10.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§10.30) which determines the internal values for `\leftmargin` and `\itemindent`.

```

44 \cs_set_protected:Npn \__enumext_tmp:n #1
45 {
46   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
47   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
48   \dim_new:c { l__enumext_leftmargin_#1_dim }
49   \dim_new:c { l__enumext_itemindent_#1_dim }
50 }
51 \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).

```

52 \cs_set_protected:Npn \__enumext_tmp:n #1
53 {
54   \skip_new:c { l__enumext_multicols_above_#1_skip }
55   \skip_new:c { l__enumext_multicols_below_#1_skip }
56 }
57 \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).

```

58 \int_new:N \g__enumext_minipage_stat_int
59 \skip_new:N \l__enumext_minipage_left_skip
60 \skip_new:N \l__enumext_minipage_right_skip
61 \skip_new:N \l__enumext_minipage_after_skip
62 \skip_new:N \g__enumext_minipage_right_skip
63 \skip_new:N \g__enumext_minipage_after_skip
64 \cs_set_protected:Npn \__enumext_tmp:n #1
65 {
66   \dim_new:c { \l__enumext_minipage_left_#1_dim }
67   \bool_new:c { \l__enumext_minipage_active_#1_bool }
68 }
69 \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

```

70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \bool_new:c { \l__enumext_wrap_label_#1_bool }
73   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
74   \int_new:c { \l__enumext_start_#1_int }
75   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
76   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
77   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
78   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
79   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
80 }
81 \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
\l__enumext_keyans_tmpb_tl
\l__enumext_keyans_tmpa_dim

```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§10.21) 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 (§10.21).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.28.2) 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.

```

82 \bool_new:N \l__enumext_store_active_bool
83 \tl_new:N \l__enumext_store_name_tl
84 \tl_new:N \g__enumext_store_name_tl
85 \tl_new:N \l__enumext_store_anskey_arg_tl
86 \int_new:N \l__enumext_store_columns_join_int
87 \tl_new:N \l__enumext_store_keyans_label_tl
88 \tl_new:N \l__enumext_store_keyans_item_opt_tl
89 \tl_new:N \l__enumext_keyans_item_opt_tl
90 \tl_new:N \l__enumext_keyans_tmpa_tl
91 \tl_new:N \l__enumext_keyans_tmpb_tl
92 \dim_new:N \l__enumext_keyans_tmpa_dim

```

(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).

```

93 \tl_new:N \l__enumext_setkey_tmpa_tl
94 \tl_new:N \l__enumext_setkey_tmpb_tl
95 \int_new:N \l__enumext_setkey_tmpa_int
96 \seq_new:N \l__enumext_setkey_tmpa_seq
97 \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 [$\langle key = val \rangle$] in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.38) and the keys `columns*` and `columns-sep*`.

```
98 \cs_set_protected:Npn \l__enumext_tmp:n #1
99 {
100   \tl_new:c { \l__enumext_store_opt_#1_tl }
101   \tl_new:c { \l__enumext_print_keyans_#1_tl }
102   \bool_new:c { \l__enumext_store_columns_#1_bool }
103   \int_new:c { \l__enumext_store_columns_#1_int }
104   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
105   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
106   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
107 }
108 \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` (§10.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.23).

```
109 \bool_new:N \l__enumext_show_answer_bool
110 \bool_new:N \l__enumext_show_position_bool
111 \tl_new:N \l__enumext_mark_ref_sym_tl
112 \tl_new:N \l__enumext_mark_answer_sym_tl
113 \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).

```
114 \seq_new:N \l__enumext_keyans_pic_body_seq
115 \dim_new:N \l__enumext_keyans_pic_width_dim
116 \int_new:N \l__enumext_keyans_pic_above_int
117 \int_new:N \l__enumext_keyans_pic_below_int
118 \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_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int
```

Internal variables used by “check answer” mechanism (§10.22) controlled by the `check-ans` and `no-store` keys.

```
119 \bool_new:N \l__enumext_store_ans_bool
120 \bool_new:N \l__enumext_check_ans_bool
121 \bool_new:N \g__enumext_check_ans_show_bool
122 \bool_new:N \g__enumext_check_ans_show_h_bool
123 \tl_new:N \g__enumext_check_ans_item_tl
124 \int_new:N \g__enumext_count_item_anskey_int
125 \int_new:N \g__enumext_count_item_number_int
126 \int_new:N \g__enumext_standar_star_env_int
127 \int_new:N \g__enumext_starred_star_env_int
128 \int_new:N \g__enumext_starred_keyans_star_env_int
129 \int_new:N \g__enumext_standar_keyans_star_env_int
130 \int_new:N \g__enumext_standar_keyans_pic_star_env_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`.

```
131 \bool_new:N \l__enumext_hyperref_bool
132 \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.

```
133 \tl_new:N \l__enumext_newlabel_arg_one_tl
134 \tl_new:N \l__enumext_newlabel_arg_two_tl
```

```

135 \tl_new:N \l__enumext_store_write_aux_file_tl
136 \cs_set_protected:Npn \__enumext_tmp:n #1
137 {
138   \tl_new:c { l__enumext_label_copy_#1_tl }
139 }
140 \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`

Internal variables used for redefinition of `\footnote`.

`\g__enumext_footnote_arg_seq`
`\g__enumext_footnote_int_seq`

```

141 \int_new:N \g__enumext_footnote_int
142 \seq_new:N \g__enumext_footnote_arg_seq
143 \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`.)

`\c__enumext_counter_style_tl`
`\l__enumext_ref_key_arg_tl`
`\l__enumext_ref_aux_tl`
`\l__enumext_the_counter_X_tl`
`\l__enumext_counter_style_for_ref_X_tl`

Internal variables used by `ref` key (§10.17, §10.18).

```

144 \tl_const:Nn \c__enumext_counter_style_tl
145 { { arabic } { roman } { Roman } { alph } { Alph } }
146 \tl_new:N \l__enumext_ref_key_arg_tl
147 \tl_new:N \l__enumext_ref_aux_tl
148 \cs_set_protected:Npn \__enumext_tmp:n #1
149 {
150   \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
151   \tl_new:c { l__enumext_the_counter_#1_tl }
152   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
153 }
154 \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.)

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

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

```

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

```

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

`\c__enumext_all_envs_clist`

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

```

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

```

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 Some utility functions

`__enumext_at_begin_document:n`

A internal “hook” function used for copying plain `list` and `minipage` environments definition and `hyperref` detection.

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

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

`__enumext_after_env:nn`

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

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

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

`__enumext_level:`

Function for check current level in `enumext`.

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

(End of definition for `__enumext_level:.`)

`__enumext_if_is_int:nT`

A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys. This function is taken directly from the answer given by Henri Menke in [How to test if an expl3 function argument is an integer expression?](#).

`__enumext_if_is_int:nF`

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

(End of definition for `__enumext_if_is_int:nT`, `__enumext_if_is_int:nF`, and `__enumext_if_is_int:nTF`.)

`__enumext_show_length:nnn`

Internal function used by `show-length` key to show “all lengths” calculated and use in `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```
198 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
199 {
200   * ~ #2
201   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
202   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
203 }
```

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
204 \cs_set_protected:Nn \__enumext_zero_count_level:
205 {
206   \cs_set_protected:Npn \__enumext_tmp:n ##1
207   {
208     \int_gzero:c { g__enumext_count_level_##1_int }
209   }
210   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
211 }
```

(End of definition for `__enumext_zero_count_level:.`)

`__enumext_current_env:`

The function `__enumext_current_env:` will set the global variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` with which we will distinguish whether the environments `enumext` and `enumext*` are nested in each other.

```
212 \cs_new_protected:Nn \__enumext_current_env:
213 {
214   \str_case:en { \@currenvir }
215   {
216     {enumext}
```

```

217         {
218             \int_compare:nNtT { \__enumext_level_h_int } = { \c_zero_int }
219             {
220                 \bool_gset_true:N \g__enumext_standar_bool
221                 \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
222                 \typeout{working-on-enumext}
223             }
224         }
225     {enumext*}
226     {
227         \int_compare:nNtT { \__enumext_level_int } = { \c_zero_int }
228         {
229             \bool_gset_true:N \g__enumext_starred_bool
230             \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
231             \typeout{working-on-enumext*}
232         }
233     }
234 }
235 }

```

(End of definition for `__enumext_current_env:`.)

10.6 Copying list and minipage environments

The `\list` environment provided by \LaTeX has the following plain form:

```

\list{⟨arg one⟩}{⟨arg two⟩}
  \item[⟨opt⟩]
\endlist

```

As a precaution we copy them using `__enumext_at_begin_document:n` in case any package redefines the `\list` environment or a related command.

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w

```

The functions `__enumext_start_list:nn`, `__enumext_stop_list:` and `__enumext_item_std:w` correspond to copies of `\list`, `\endlist` and `\item` from plain definition of `\list` environment.

```

236 \__enumext_at_begin_document:n
237 {
238     \cs_new_eq:NN \__enumext_start_list:nn \list
239     \cs_new_eq:NN \__enumext_stop_list: \endlist
240     \cs_new_eq:NN \__enumext_item_std:w \item
241 }

```

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

The `\minipage` environment provided by \LaTeX has the following (simplified) plain form:

```

\minipage[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}
  ⟨internal implement⟩
\endminipage

```

As a precaution we copy them using `__enumext_at_begin_document:n` in case any package redefines the `\minipage` environment or a related command.

```

\__enumext_minipage:w
\__enumext_endminipage:

```

The functions `__enumext_minipage:w`, `__enumext_endminipage:` and correspond to copies of `\minipage`, `\endminipage` from plain definition of `\minipage` environment.

```

242 \__enumext_at_begin_document:n
243 {
244     \cs_new_eq:NN \__enumext_minipage:w \minipage
245     \cs_new_eq:NN \__enumext_endminipage: \endminipage
246 }

```

(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.

```

247 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
248 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

The function `__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyperref_bool` to “true” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `__enumext_footnotes_key_bool` to “true”.

```

249 \cs_new_protected:Nn \__enumext_after_hyperref:

```

```

250 {
251   \IfPackageLoadedTF { hyperref }
252   {
253     \msg_info:nnn { enumext } { package-load } { hyperref }
254     \bool_set_true:N \l__enumext_hyperref_bool
255     \IfHyperBoolean{hyperfootnotes}
256     {
257       \typeout{hyperfootnotes=true}
258       \bool_set_true:N \l__enumext_footnotes_key_bool
259     }
260     { \typeout{hyperfootnotes=false} }
261   }
262   { }

```

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`.

```

263   \bool_if:NT \l__enumext_footnotes_key_bool
264   {
265     \IfPackageLoadedTF { footnotehyper }
266     {
267       \msg_info:nnn { enumext } { package-load } { footnotehyper }
268     }
269     {
270       \typeout{No ~ footnotehyper ~ load}
271       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
272       \bool_set_false:N \l__enumext_footnotes_key_bool
273     }
274   }

```

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.

```

275   \bool_if:NTF \l__enumext_hyperref_bool
276   {
277     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
278     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
279   }
280   {
281     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
282     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
283   }
284 }

```

(End of definition for `__enumext_after_hyperref:`, `__enumext_hypertarget:nn`, and `__enumext_phantomsection:`.)

`__enumext_newlabel:nn` The function `__enumext_newlabel:nn` write the information to the `.aux` file when using the `save-ref` key. The arguments taken by the function are:

#1: `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

The trick here is to manage the number of arguments passed to `\newlabel{#1}{#2}` according to the presence of the `hyperref` package.

```

285 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
286 {
287   \protected@write \@auxout { }
288   {
289     \token_to_str:N \newlabel {#1}
290     {
291       {#2}
292       \bool_if:NT \l__enumext_hyperref_bool
293       { { \thepage } {#2} {#1} }
294       { }
295     }
296   }
297   \__enumext_hypertarget:nn {#1} { }
298   \__enumext_phantomsection:
299 }

```

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

10.8 Definition of counters

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

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

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

```
300 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
301 {
302   \cs_if_exist:cTF { c@ #2 }
303   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
304   {
305     \tl_set:Nn #1 { #2 }
306     \newcounter { #2 }
307   }
308 }
```

(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      309 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     310 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    311 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     312 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv      313 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvii    314 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii   315 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
316 \__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.

```
317 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
318 {
319   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
320   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
321 }
322 \__enumext_register_counter_style:Nn \arabic { 0 }
323 \__enumext_register_counter_style:Nn \Alph { M }
324 \__enumext_register_counter_style:Nn \alph { m }
325 \__enumext_register_counter_style:Nn \Roman { VIII }
326 \__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.

```
327 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
328 {
329   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
330   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
331 }
332 \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.

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

```

334 {
335   \tl_clear_new:N #1
336   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
337   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
338   \tl_map_inline:Nn \g__enumext_counter_styles_tl
339   {
340     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
341     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
342     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
343   }
344   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
345   { \tl_use:N \g__enumext_widest_label_tl }
346   \tl_set_eq:cN { the #2 } #1
347 }
348 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

10.10 Setting keys associated with label

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

```

labelsep
labelwidth
wrap-label
wrap-label*
349 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
350 {
351   \keys_define:nn { enumext / #1 }
352   {
353     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
354     font .value_required:n = true,
355     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
356     labelsep .initial:n = {0.3333em},
357     labelsep .value_required:n = true,
358     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
359     labelwidth .value_required:n = true,
360     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
361     wrap-label .initial:n = {##1},
362     wrap-label .value_required:n = true,
363     wrap-label* .code:n = {
364       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
365       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
366     },
367     wrap-label* .value_required:n = true,
368   }
369 }
370 \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.

```

371 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
372 {
373   \keys_define:nn { enumext / #1 }
374   {
375     align .choice:,
376     align / left .code:n =
377     {
378       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
379       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
380     },
381     align / right .code:n =
382     {
383       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
384       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
385     },
386     align / center .code:n =
387     {
388       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
389       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
390     },

```

```

391     align .initial:n = left,
392     align .value_required:n = true,
393   }
394 }
395 \clist_map_inline:nn
396 {
397   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
398 }
399 { \__enumext_tmp:nn #1 }

```

Definition of `align` key for `enumext*` and `keyans*` environments.

```

400 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
401 {
402   \keys_define:nn { enumext / #1 }
403   {
404     align .choice:,
405     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
406     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
407     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
408     align .initial:n = left,
409     align .value_required:n = true,
410   }
411 }
412 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

10.11 Setting label and ref keys

`__enumext_regex_label_ref_key:`

The internal function `__enumext_regex_label_ref_key:` replace the `*` with the actual counter of the running level and is used by the `__enumext_set_label_ref:n` function.

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.

```

413 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
414 {
415   \tl_map_inline:Nn \c__enumext_counter_style_tl
416   {
417     \regex_replace_once:nnN { \c{##1}}{*}
418     { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
419   }
420 }

```

(End of definition for `__enumext_regex_label_ref_key:`.)

`__enumext_set_label_ref:n`

The `__enumext_set_label_ref:n` function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable `\l__enumext_the_counter_X_tl` according to the command created for *each counter*, apply the `regex` function `__enumext_regex_label_ref_key:` and then renew the command and save it in the variable `\l__enumext_counter_style_for_ref_X_tl`.

```

421 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
422 {
423   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
424   \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
425   \__enumext_regex_label_ref_key:
426   \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
427   \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
428   {
429     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
430     { \exp_not:V \l__enumext_ref_key_arg_tl }
431   }
432 }

```

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

`__enumext_use_key_ref:`

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

```

433 \cs_new_protected:Nn \__enumext_use_key_ref:
434 {
435   \tl_if_empty:cF { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
436   {

```



```

437         \tl_use:c { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
438     }
439 }

```

(End of definition for `__enumext_use_key_ref:`.)

For `enumext*` and `keyans*` environments the situation is a bit different since `hyperref` interferes here (I am not clear why), so we will define a new function to execute the task.

To handle that we will look at the nesting level of the starred environments, later I will run the constraint functions to make everything OK.

The `__enumext_set_label_ref_h:n` function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable `\l__enumext_the_counter_X_tl` according to the command created for *each counter*, apply the `regex` function `__enumext_regex_label_ref_key:` and then renew the command and save it in the variable `\l__enumext_counter_style_for_ref_X_tl`.

```

440 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
441 {
442     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
443     \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
444     {
445         \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
446         \__enumext_regex_label_ref_key:
447         \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
448         \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
449         {
450             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
451             { \exp_not:V \l__enumext_ref_key_arg_tl }
452         }
453     }
454     {
455         \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
456         \__enumext_regex_label_ref_key:
457         \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
458         \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
459         {
460             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
461             { \exp_not:V \l__enumext_ref_key_arg_tl }
462         }
463     }
464 }

```

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

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

```

465 \cs_new_protected:Npn \__enumext_use_key_ref_h:
466 {
467     \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
468     {
469         \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
470         {
471             \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
472         }
473     }
474     {
475         \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
476         {
477             \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
478         }
479     }
480 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
481 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
482 {
483     \keys_define:nn { enumext / #1 }

```

```

484     {
485         label .code:n = {
486             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
487             { l__enumext_counter_#2_tl } {##1}
488             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
489             \l__enumext_current_widest_dim
490         },
491         label .initial:n = #3,
492         label .value_required:n = true,
493         ref .code:n = \__enumext_set_label_ref:n {##1},
494         ref .value_required:n = true,
495     }
496 }
497 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
498 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
499 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
500 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

10.11.2 Define and set `label` key for `enumext*` and `keyans*` environments

Here we set the default `<label>` for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key.

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
501 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
502 {
503     \keys_define:nn { enumext / #1 }
504     {
505         label .code:n = {
506             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
507             { l__enumext_counter_#2_tl } {##1}
508             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
509             \l__enumext_current_widest_dim
510         },
511         label .initial:n = #3,
512         label .value_required:n = true,
513         ref .code:n = \__enumext_set_label_ref_h:n {##1},
514         ref .value_required:n = true,
515     }
516 }
517 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
518 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

10.11.3 Define and set `label` key for `keyans` and `keyanspic` environment

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

Define and set `label` key for `keyans` environment.

```

519 \keys_define:nn { enumext / keyans }
520 {
521     label .code:n = {
522         \__enumext_label_style:cvn { l__enumext_label_v_tl }
523         { l__enumext_counter_v_tl } {##1}
524         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
525         \l__enumext_current_widest_dim
526         \__enumext_label_style:cvn { l__enumext_label_vi_tl }
527         { l__enumext_counter_vi_tl } {##1}
528         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
529         \l__enumext_current_widest_dim
530     },
531     label .initial:n = (\Alph*),
532     label .value_required:n = true,
533 }

```

(End of definition for `label`, `\l__enumext_label_v_tl`, and `\l__enumext_label_vi_tl`.)

10.12 Setting start and widest keys

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: $\langle integer \text{ or } string \rangle$

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 $\langle integer \rangle$ or $\langle string \rangle$ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start=A` or `start=1` to be used.

```

534 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
535 {
536   \__enumext_if_is_int:nTF { #3 }
537   {
538     \int_set:Nn #2 {#3}
539   }
540   {
541     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
542     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
543     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
544     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
545   }
546 }
547 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

#1: The counter associated with the environment level

#2: `\l__enumext_label_X_tl`

#3: `\l__enumext_labelwidth_X_dim`

#4: $\langle integer \text{ or } string \rangle$

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

```

548 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
549 {
550   \__enumext_if_is_int:nTF {#4}
551   {
552     \setcounter{enumX#1} { #4 }
553   }
554   {
555     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
556     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
557     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
558     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
559   }
560   \__enumext_label_width_by_box:cv
561   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
562 }
563 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

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

```

start
widest
\l__enumext_start_X_int
564 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
565 {
566   \keys_define:nn { enumext / #1 }
567   {
568     start .code:n = {
569       \__enumext_start_from:ccn
570       { \l__enumext_label_#2_tl }
571       { \l__enumext_start_#2_int } {##1}
572     },
573     start .initial:n = 1,
574     widest .code:n = {
575       \__enumext_widest_from:nccn {#2}
576       { \l__enumext_label_#2_tl }
577       { \l__enumext_labelwidth_#2_dim } {##1}
578     },
579     widest .value_required:n = true,
580     start .value_required:n = true,
581   }
582 }
583 \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.

```

topsep
partopsep
parsep
noitemsep
nosep
584 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
585 {
586   \keys_define:nn { enumext / #1 }
587   {
588     topsep      .skip_set:c = { l__enumext_topsep_#2_skip },
589     topsep      .initial:n   = { #3 },
590     topsep      .value_required:n = true,
591     partopsep   .skip_set:c = { l__enumext_partopsep_#2_skip },
592     partopsep   .initial:n   = { #4 },
593     partopsep   .value_required:n = true,
594     parsep      .skip_set:c = { l__enumext_parsep_#2_skip },
595     parsep      .initial:n   = { #5 },
596     parsep      .value_required:n = true,
597     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
598     itemsep     .initial:n   = { #6 },
599     itemsep     .value_required:n = true,
600     noitemsep   .meta:n      = { itemsep = 0pt, parsep = 0pt },
601     noitemsep   .value_forbidden:n = true,
602     nosep       .meta:n      = {
603                           itemsep = 0pt, parsep = 0pt,
604                           topsep = 0pt, partopsep = 0pt,
605                           },
606     nosep       .value_forbidden:n = true,
607   }
608 }
```

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

```

609 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
610 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
611 { 4.0pt plus 2.0pt minus 1.0pt }
612 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
613 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
614 { 2.0pt plus 1.0pt minus 1.0pt }
615 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
616 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
617 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
618 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
619 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
620 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
621 { 2.0pt plus 1.0pt minus 1.0pt }
622 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
623 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
624 { 4.0pt plus 2.0pt minus 1.0pt }
625 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
626 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
627 { 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.

```

628 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
629 {
630   \keys_define:nn { enumext / #1 }
631   {
632     itemindent   .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
633     itemindent   .value_required:n = true,
634     rightmargin  .dim_set:c = { l__enumext_rightmargin_#2_dim },
635     rightmargin  .value_required:n = true,
636     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
637     listparindent .value_required:n = true,
638     list-offset  .dim_set:c = { l__enumext_listoffset_#2_dim },
639     list-offset  .value_required:n = true,
640     list-indent  .code:n      =
```

```

641         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
642         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
643     list-indent .value_required:n = true,
644 }
645 }
646 \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.

```

647 \cs_set_protected:Npn \__enumext_tmp:n #1
648 {
649     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
650 }
651 \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.

```

652 \cs_set_protected:Nn \__enumext_fake_item:
653 {
654     \dim_compare:nNnT
655     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
656     >
657     { \c_zero_dim }
658     {
659         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
660         {
661             \exp_not:N \mode_leave_vertical:
662             \exp_not:n { \skip_horizontal:n }
663             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
664             \ignorespaces
665         }
666     }
667 }
668 \cs_set_protected:Nn \__enumext_keyans_fake_item:
669 {
670     \dim_compare:nNnT
671     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
672     {
673         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
674         {
675             \exp_not:N \mode_leave_vertical:
676             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
677         }
678     }
679 }
680 \cs_set_protected:Nn \__enumext_fake_item_vii:
681 {
682     \dim_compare:nNnT
683     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
684     {
685         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
686         {
687             \exp_not:N \mode_leave_vertical:
688             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
689         }
690     }
691 }
692 \cs_set_protected:Nn \__enumext_fake_item_viii:
693 {
694     \dim_compare:nNnT
695     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
696     {
697         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
698         {
699             \exp_not:N \mode_leave_vertical:
700             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim

```

```

701     }
702   }
703 }

```

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

10.15 Setting show-length key

show-length

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

```

704 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
705 {
706   \keys_define:nn { enumext / #1 }
707   {
708     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
709     show-length .initial:n = false,
710   }
711 }
712 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

before
before*
after
first

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

```

713 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
714 {
715   \keys_define:nn { enumext / #1 }
716   {
717     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
718     before .value_required:n = true,
719     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
720     before* .value_required:n = true,
721     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
722     after .value_required:n = true,
723     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
724     first .value_required:n = true,
725   }
726 }
727 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16.1 Functions for before, after and first keys in enumext

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

```

728 \cs_new_protected:Nn \__enumext_before_args_exec:
729 {
730   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
731 }

```

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

```

732 \cs_new_protected:Nn \__enumext_before_keys_exec:
733 {
734   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
735 }

```

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

```

736 \cs_new_protected:Nn \__enumext_after_stop_list:
737 {
738   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
739 }

```

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`.

```
740 \cs_new_protected:Nn \__enumext_after_args_exec:
741 {
742   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
743 }
```

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

10.16.2 Functions for before, after and first keys in keyans

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

```
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
744 \cs_new_protected:Nn \__enumext_before_args_exec_v:
745 {
746   \tl_use:N \l__enumext_before_starred_key_v_tl
747 }
```

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

```
748 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
749 {
750   \tl_use:N \l__enumext_before_no_starred_key_v_tl
751 }
```

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

```
752 \cs_new_protected:Nn \__enumext_after_stop_list_v:
753 {
754   \tl_use:N \l__enumext_after_stop_list_v_tl
755 }
```

The function `__enumext_after_args_exec_v`: executes the $\{\langle code \rangle\}$ set by the `first` key after the end of $\{\langle arg two \rangle\}$ of the list defining the `keyans` environment, just before the first occurrence of `\item`.

```
756 \cs_new_protected:Nn \__enumext_after_args_exec_v:
757 {
758   \tl_use:N \l__enumext_after_list_args_v_tl
759 }
```

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

```
\__enumext_before_keys_exec_vii:
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
760 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
761 {
762   \tl_use:N \l__enumext_before_starred_key_vii_tl
763 }
764 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
765 {
766   \tl_use:N \l__enumext_before_starred_key_viii_tl
767 }
```

The functions `__enumext_before_keys_exec_vii`: and `__enumext_before_keys_exec_viii`: executes the $\{\langle code \rangle\}$ set by the `before` key “before” in `enumext*` and `keyans*` environments is started in $\{\langle arg two \rangle\}$ of the list. The $\{\langle code \rangle\}$ is executed “knowing” all definition and values provides by $\langle keys \rangle$.

```
768 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
769 {
770   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
771 }
772 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
773 {
774   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
775 }
```


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

```

776 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
777 {
778   \tl_use:N \l__enumext_after_stop_list_vii_tl
779 }
780 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
781 {
782   \tl_use:N \l__enumext_after_stop_list_viii_tl
783 }

```

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`.

```

784 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
785 {
786   \tl_use:N \l__enumext_after_list_args_vii_tl
787 }
788 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
789 {
790   \tl_use:N \l__enumext_after_list_args_viii_tl
791 }

```

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

10.17 Setting keys for multicols and minipage

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

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

`columns-sep`

`columns`

```

792 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
793 {
794   \keys_define:nn { enumext / #1 }
795   {
796     mini-env   .dim_set:c = { \l__enumext_minipage_right_#2_dim },
797     mini-env   .value_required:n = true,
798     mini-sep   .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
799     mini-sep   .initial:n = 0.3333em,
800     mini-sep   .value_required:n = true,
801     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
802     columns-sep .value_required:n = true,
803     columns    .int_set:c = { \l__enumext_columns_#2_int },
804     columns    .initial:n = 1,
805     columns    .value_required:n = true,
806   }
807 }
808 \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`.

```

809 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
810 {
811   \keys_define:nn { enumext / #1 }
812   {
813     columns    .initial:n = 2,
814     miniright  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
815     miniright  .value_required:n = true,
816     miniright* .code:n = {
817       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
818       \keys_set:nn { enumext / #1 } { miniright = {##1} }
819     },
820     miniright* .value_required:n = true,
821   }
822 }
823 \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 multicol

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



Figure 7: Representation of the vertical space in `multicol` 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 `multicol` environment. The most appropriate option in this case is to use a “*context sensitive*” vertical space with `\addvspace`.

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

10.18.1 Adjustment of vertical spaces for multicol 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 `multicol` 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_multicol_above_X_skip` and `__enumext_multicol_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

824 \cs_new_protected:Nn \__enumext_multi_set_vskip:
825 {
826   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip {
827     {
828       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
829         }
830       \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip {
831         {
832           \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
833             }
834           \__enumext_add_pre_parsep:
835         }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

836 \cs_new_protected:Nn \__enumext_add_pre_parsep:
837 {
838   \int_case:nn { \__enumext_level_int }
839   {
840     { 2 } {
841       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip } {
842         {
843           \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
844         }
845       }
846     { 3 } {
847       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip } {
848         {
849           \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
850         }
851       }
852     { 4 } {
853       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip } {
854         {
855           \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
856         }
857       }

```

```

858     }
859 }

```

(End of definition for `__enumext_add_pre_parse:`)

`__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)*.

```

860 \cs_new_protected:Nn \__enumext_multi_addvspace:
861 {
862   \__enumext_multi_set_vskip:
863   \mode_if_vertical:T
864   {
865     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
866     {
867       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
868     }
869     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
870     {
871       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
872     }
873   }
874   \par\nopagebreak
875   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
876 }

```

(End of definition for `__enumext_multi_addvspace:`)

10.18.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

877 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
878 {
879   \skip_set:Nn \__enumext_multicols_above_v_skip
880   {
881     \__enumext_topsep_v_skip
882   }
883   \skip_set:Nn \__enumext_multicols_below_v_skip
884   {
885     \__enumext_topsep_v_skip
886   }
887 }
888 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
889 {
890   \__enumext_keyans_multi_set_vskip:
891   \mode_if_vertical:T
892   {
893     \skip_add:Nn \__enumext_multicols_above_v_skip
894     {
895       \skip_use:N \__enumext_partopsep_v_skip
896     }
897     \skip_add:Nn \__enumext_multicols_below_v_skip
898     {
899       \skip_use:N \__enumext_partopsep_v_skip
900     }
901   }
902   \par\nopagebreak
903   \addvspace{ \__enumext_multicols_above_v_skip }
904 }

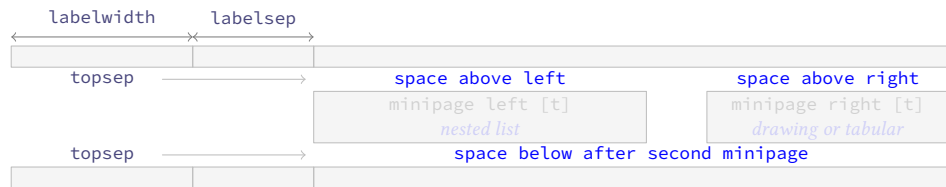
```

(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.

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.

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

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.

```

905 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
906 {
907     \__enumext_minipage:w [ t ] { #1 }
908     \legacy_if_gset_false:n { @minipage }
909     \vspace { 0pt }
910 }
911 { \__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`.

```

912 \cs_new_protected:Nn \__enumext_mini_set_vskip:
913 {
914     \int_compare:nNnTF
915     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
916     {
```

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 `\partopsep` of the previous level, if these are zero we will use `\strutbox` as the basis for the calculations.

```

917     \skip_if_eq:nnTF
918     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
919     {
920         \skip_set:Nn \l__enumext_minipage_left_skip
921         {
922             -0.150\box_dp:N \strutbox
923         }
924         \skip_set:Nn \l__enumext_minipage_right_skip
925         {
926             0.695\box_dp:N \strutbox
927         }
928         \skip_set:Nn \l__enumext_minipage_after_skip
929         {
930             \box_dp:N \strutbox
931         }
932         \__enumext_zero_parsep:
933     }
```

```

934     {
935         \skip_set:Nn \l__enumext_minipage_left_skip
936         {
937             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
938         }
939         \skip_set:Nn \l__enumext_minipage_right_skip
940         {
941             0.695\box_dp:N \strutbox
942         }
943         \skip_set:Nn \l__enumext_minipage_after_skip
944         {
945             1.85\box_dp:N \strutbox
946             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
947         }
948     }
949 }
950 {

```

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.

```

951 \skip_if_eq:nnTF
952 { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
953 {
954     \skip_set:Nn \l__enumext_minipage_left_skip
955     {
956         0.5\box_dp:N \strutbox
957         - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
958     }
959     \skip_set:Nn \l__enumext_minipage_right_skip
960     {
961         \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
962     }
963     \skip_set:Nn \l__enumext_minipage_after_skip
964     {
965         1.6\box_dp:N \strutbox
966     }
967 }
968 {
969     \skip_set:Nn \l__enumext_minipage_left_skip
970     {
971         0.5875\box_dp:N \strutbox
972         - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
973     }
974     \skip_set:Nn \l__enumext_minipage_right_skip
975     {
976         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
977         + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
978     }
979     \skip_set:Nn \l__enumext_minipage_after_skip
980     {
981         0.325\box_dp:N \strutbox
982         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
983     }
984 }
985 }
986 }

```

(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.

```

987 \cs_new_protected:Nn \__enumext_zero_parsep:
988 {
989     \int_case:nn { \l__enumext_level_int }
990     {
991         { 2 }{
992             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
993             {
994                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }

```

```

995         }
996     }
997     { 3 }{
998         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
999         {
1000             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1001         }
1002     }
1003     { 4 }{
1004         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1005         {
1006             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1007         }
1008     }
1009 }
1010 }

```

(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*.

```

1011 \cs_new_protected:Nn \__enumext_mini_addvspace:
1012 {
1013     \__enumext_mini_set_vskip:
1014     \mode_if_vertical:T
1015     {
1016         \skip_add:Nn \l__enumext_minipage_left_skip
1017         {
1018             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1019         }
1020         \skip_add:Nn \l__enumext_minipage_after_skip
1021         {
1022             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1023         }
1024     }
1025     \par\nopagebreak
1026     \addvspace { \l__enumext_minipage_left_skip }
1027 }

```

(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`.

```

1028 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1029 {
1030     \skip_zero_new:N \l__enumext_minipage_after_skip
1031     \skip_zero_new:N \l__enumext_minipage_left_skip
1032     \skip_zero_new:N \l__enumext_minipage_right_skip
1033     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1034     {
1035         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1036         {
1037             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1038             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1039             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1040             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1041             {
1042                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1043             }
1044         }
1045     }
1046     \skip_set:Nn \l__enumext_minipage_left_skip
1047     {
1048         \skip_use:N \l__enumext_topsep_v_skip
1049     }
1050     \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1051         {
1052             0.705\box_dp:N \strutbox
1053         }
1054         \skip_set:Nn \l__enumext_minipage_after_skip
1055         {
1056             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1057         }
1058     }
1059 }
1060 {
1061     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1062     {
1063         \skip_set:Nn \l__enumext_minipage_left_skip
1064         {
1065             0.5\box_dp:N \strutbox
1066             + \l__enumext_partopsep_v_skip
1067         }
1068         \skip_set:Nn \l__enumext_minipage_right_skip
1069         {
1070             \l__enumext_partopsep_v_skip
1071         }
1072         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1073     }
1074     {
1075         \skip_set:Nn \l__enumext_minipage_left_skip
1076         {
1077             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1078         }
1079         \skip_set:Nn \l__enumext_minipage_right_skip
1080         {
1081             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1082         }
1083         \skip_set:Nn \l__enumext_minipage_after_skip
1084         {
1085             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1086         }
1087     }
1088 }
1089 }

```

(End of definition for `\l__enumext_keyans_mini_set_vskip:`)

`\l__enumext_keyans_mini_addvspace:`

The function `\l__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\l__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`.

```

1090 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1091 {
1092     \l__enumext_keyans_mini_set_vskip:
1093     \mode_if_vertical:T
1094     {
1095         \skip_add:Nn \l__enumext_minipage_left_skip
1096         {
1097             \l__enumext_partopsep_v_skip
1098         }
1099         \skip_add:Nn \l__enumext_minipage_after_skip
1100         {
1101             \l__enumext_partopsep_v_skip
1102         }
1103     }
1104     \par\nopagebreak
1105     \addvspace { \l__enumext_minipage_left_skip }
1106 }

```

(End of definition for `\l__enumext_keyans_mini_addvspace:`)

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

`\l__enumext_mini_set_vskip_vii:`

`\l__enumext_mini_set_vskip_viii:`

The functions `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `\l__enumext_mini_env*` environment in `enumext*` and `keyans*`.


```

1107 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1108 {
1109   \skip_zero_new:N \l__enumext_minipage_left_skip
1110   \skip_gzero_new:N \g__enumext_minipage_right_skip
1111   \skip_gzero_new:N \g__enumext_minipage_after_skip
1112   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1113   {
1114     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1115     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1116   }
1117   {
1118     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1119     \skip_gset:Nn \g__enumext_minipage_right_skip
1120       {
1121         \l__enumext_topsep_vii_skip
1122       }
1123     \skip_gset:Nn \g__enumext_minipage_after_skip
1124       {
1125         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1126       }
1127   }
1128 }
1129 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1130 {
1131   \skip_zero_new:N \l__enumext_minipage_after_skip
1132   \skip_zero_new:N \l__enumext_minipage_left_skip
1133   \skip_zero_new:N \l__enumext_minipage_right_skip
1134   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1135   {
1136     \skip_set:Nn \l__enumext_minipage_left_skip
1137       {
1138         0.5\box_dp:N \strutbox
1139       }
1140     \skip_set:Nn \l__enumext_minipage_right_skip
1141       {
1142         \l__enumext_partopsep_viii_skip
1143       }
1144     \skip_set:Nn \l__enumext_minipage_after_skip
1145       {
1146         1.6\box_dp:N \strutbox
1147       }
1148   }
1149   {
1150     \skip_set:Nn \l__enumext_minipage_left_skip
1151       {
1152         0.5875\box_dp:N \strutbox
1153       }
1154     \skip_set:Nn \l__enumext_minipage_right_skip
1155       {
1156         \l__enumext_topsep_viii_skip
1157       }
1158     \skip_set:Nn \l__enumext_minipage_after_skip
1159       {
1160         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1161       }
1162   }
1163 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext_mini_env*` environment on the *left side* when the `miniright` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether T_EX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1164 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1165 {
1166   \__enumext_mini_set_vskip_vii:
1167   \par\nopagebreak
1168   \addvspace { \l__enumext_minipage_left_skip }
1169 }

```

```

1170 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1171 {
1172   \__enumext_mini_set_vskip_viii:
1173   \par\nopagebreak
1174   \addvspace { \__enumext_minipage_left_skip }
1175 }

```

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

10.19.4 The command \miniright

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

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

```

1176 \NewDocumentCommand \miniright { s }
1177 {
1178   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1179   {
1180     \msg_error:nnn { enumext } { wrong-miniright-place }
1181   }
1182   \int_compare:nNt { \__enumext_level_int } = { 0 }
1183   {
1184     \msg_error:nnn { enumext } { wrong-miniright-place }
1185   }
1186   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1187   {
1188     \__enumext_keyans_mini_right_cmd:n {#1}
1189   }
1190   { \__enumext_mini_right_cmd:n {#1} }
1191 }

```

(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.

```

1192 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1193 {
1194   \dim_compare:nNtF
1195   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1196   {
1197     \__enumext_multicols_stop:
1198     \end{__enumext_mini_env*}
1199     \hfill
1200     \begin{__enumext_mini_env*}
1201     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1202     \par\addvspace { \__enumext_minipage_right_skip }
1203     \bool_if:nF {#1}
1204     {
1205       \centering
1206     }
1207     \int_gzero:N \g__enumext_minipage_stat_int
1208   }
1209   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1210 }

```

(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 `\\mini\\right` 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.

```

1211 \\cs_new_protected:Npn \\_enumext_keyans_mini_right_cmd:n #1
1212 {
1213   \\dim_compare:nNnTF { \\_enumext_minipage_right_v_dim } > { \\c_zero_dim }
1214   {
1215     \\_enumext_keyans_multicols_stop:
1216     \\end{\\_enumext_mini_env*}
1217     \\hfill
1218     \\begin{\\_enumext_mini_env*}{ \\_enumext_minipage_right_v_dim }
1219     \\par\\addvspace { \\_enumext_minipage_right_skip }
1220     \\bool_if:nF {#1}
1221     {
1222       \\centering
1223     }
1224     \\int_gzero:N \\g__enumext_minipage_stat_int
1225   }
1226   { \\msg_error:nnn { enumext } { wrong-miniright-use } }
1227 }

```

(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.

```

above* 1228 \\cs_set_protected:Npn \\_enumext_tmp:nn #1 #2
below   1229 {
below*  1230   \\keys_define:nn { enumext / #1 }
        1231   {
        1232     above .skip_set:c = { \\_enumext_vspace_above_#2_skip },
        1233     above .value_required:n = true,
        1234     above* .code:n      = \\bool_set_true:c { \\_enumext_vspace_a_star_#2_bool }
        1235                   \\keys_set:nn { enumext / #1 } { above = {##1} },
        1236     above* .value_required:n = true,
        1237     below .skip_set:c = { \\_enumext_vspace_below_#2_skip },
        1238     below .value_required:n = true,
        1239     below* .code:n      = \\bool_set_true:c { \\_enumext_vspace_b_star_#2_bool }
        1240                   \\keys_set:nn { enumext / #1 } { below = {##1} },
        1241     below* .value_required:n = true,
        1242   }
        1243 }
        1244 \\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.

```

1245 \\cs_new_protected:Nn \\_enumext_vspace_above:
1246 {
1247   \\skip_if_eq:nnF
1248   { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } } { \\c_zero_skip }
1249   {
1250     \\bool_if:cTF { \\_enumext_vspace_a_star_ \\_enumext_level: _bool }
1251     {
1252       \\vspace*{ \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1253     }
1254     {
1255       \\vspace { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1256     }
1257   }
1258 }

```

(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.

```

1259 \cs_new_protected:Nn \__enumext_vspace_below:
1260 {
1261   \skip_if_eq:nnF
1262     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1263   {
1264     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1265     {
1266       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1267     }
1268     {
1269       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1270     }
1271   }
1272 }

```

(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.

```

1273 \cs_new_protected:Nn \__enumext_vspace_above_v:
1274 {
1275   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1276   {
1277     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1278     {
1279       \vspace*{ \l__enumext_vspace_above_v_skip }
1280     }
1281     { \vspace { \l__enumext_vspace_above_v_skip } }
1282   }
1283 }

```

(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.

```

1284 \cs_new_protected:Nn \__enumext_vspace_below_v:
1285 {
1286   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1287   {
1288     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1289     {
1290       \vspace*{ \l__enumext_vspace_below_v_skip }
1291     }
1292     { \vspace { \l__enumext_vspace_below_v_skip } }
1293   }
1294 }

```

(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`:

```

1295 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1296 {
1297   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1298   {
1299     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1300     {
1301       \vspace*{ \l__enumext_vspace_above_vii_skip }
1302     }
1303     { \vspace { \l__enumext_vspace_above_vii_skip } }
1304   }
1305 }
1306 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1307 {
1308   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1309   {

```

```

1310         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1311         {
1312             \vspace*{ \l__enumext_vspace_above_viii_skip }
1313         }
1314         { \vspace { \l__enumext_vspace_above_viii_skip } }
1315     }
1316 }

```

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

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

```

1317 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1318 {
1319     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1320     {
1321         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1322         {
1323             \vspace*{ \l__enumext_vspace_below_vii_skip }
1324         }
1325         { \vspace { \l__enumext_vspace_below_vii_skip } }
1326     }
1327 }
1328 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1329 {
1330     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1331     {
1332         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1333         {
1334             \vspace*{ \l__enumext_vspace_below_viii_skip }
1335         }
1336         { \vspace { \l__enumext_vspace_below_viii_skip } }
1337     }
1338 }

```

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

10.21 Setting save-ans and resume keys

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

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

```

resume
resume*
1339 \keys_define:nn { enumext / level-1 }
1340 {
1341     save-ans .code:n = \__enumext_storing_set:n {#1},
1342     save-ans .value_required:n = true,
1343     resume .code:n = \__enumext_resume_counter:,
1344     resume .value_forbidden:n = true,
1345     resume* .code:n = \__enumext_resume_counter_star:,
1346     resume* .value_forbidden:n = true,
1347 }
1348 \keys_define:nn { enumext / enumext* }
1349 {
1350     save-ans .code:n = \__enumext_storing_set:n {#1},
1351     save-ans .value_required:n = true,
1352     resume .code:n = \__enumext_resume_counter_vii:,
1353     resume .value_forbidden:n = true,
1354 }

```

(End of definition for `save-ans`, `resume`, and `resume*`.)

`__enumext_storing_set:n` The function `__enumext_storing_set:n` executed by the `save-ans` key sets the parameters for the operation of `\anskey`, `keyans` and `keyanspic`. The variable `\l__enumext_store_name_tl` will have the “store name” with which the *⟨sequence⟩* and *⟨prop list⟩* will be created, if it does not exist it will create it globally.

The boolean var `\l__enumext_store_active_bool` will be set to true activating the entire internal *storage mechanism*, then the integer variable for the `resume` key will be created (if not exist), finally the function `__enumext_check_ans_int:n` will be called to activate the internal mechanism for checking the answers if the boolean variable `\l__enumext_check_ans_bool` set by `check-ans` key are active.

```

1355 \cs_new_protected:Npn \__enumext_storing_set:n #1
1356 {
1357   \tl_set:Nx \l__enumext_store_name_tl {#1}
1358   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1359   {
1360     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1361   }
1362   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1363   {
1364     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1365   }
1366   \bool_set_true:N \l__enumext_store_active_bool
1367   \bool_set_true:N \l__enumext_store_ans_bool
1368   \int_if_exist:cF { g__enumext_resume_#1_int }
1369   {
1370     \int_new:c { g__enumext_resume_#1_int }
1371   }
1372 }

```

(End of definition for __enumext_storing_set:n.)

__enumext_resume_counter: The functions __enumext_resume_counter: and __enumext_resume_counter_vii: used by
 __enumext_resume_counter_vii: resume key in enumext and enumext*. If save-ans key present then set the start value from integer created by __enumext_storing_set:n.

```

1373 \cs_new_protected:Nn \__enumext_resume_counter:
1374 {
1375   \bool_if:NT \l__enumext_store_active_bool
1376   {
1377     \int_gset:Nn \g__enumext_resume_int
1378     {
1379       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1380     }
1381   }
1382   \bool_set_true:N \l__enumext_resume_bool
1383 }
1384 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1385 {
1386   \bool_if:NT \l__enumext_store_active_bool
1387   {
1388     \int_gset:Nn \g__enumext_resume_int
1389     {
1390       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1391     }
1392   }
1393   \bool_set_true:N \l__enumext_resume_vii_bool
1394 }

```

(End of definition for __enumext_resume_counter: and __enumext_resume_counter_vii:.)

10.22 The check answer mechanism

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

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

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

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

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

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

- If the list only has one level the number of `\item` + `\item*` = `\anskey`
- 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.

10.22.1 Setting check-ans key

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

```
check-ans 1395 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1396 {
1397   \keys_define:nn { enumext / #1 }
1398   {
1399     check-ans .bool_set:N = \__enumext_check_ans_bool,
1400     check-ans .initial:n = false,
1401     no-store .code:n = {
1402       \bool_set_false:N \__enumext_store_ans_bool
1403       \bool_set_false:N \__enumext_check_ans_bool
1404     },
1405     no-store .value_forbidden:n = true,
1406   }
1407 }
1408 \clist_map_inline:nn
1409 {
1410   level-1, level-2, level-3, level-4, enumext*
1411 }
1412 { \__enumext_tmp:n {#1} }
```

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

10.22.2 Set-up check answer mechanism

`__enumext_check_ans_set:` The function `__enumext_check_ans_set:` will adjust the value of the variable `\g__enumext_count_item_number_int` by decrementing its value by one each time you open a nested level `enumext` environment.

```
1413 \cs_new_protected:Nn \__enumext_check_ans_set:
1414 {
1415   \int_case:nn { \__enumext_level_int }
1416   {
1417     { 1 }{
1418       \bool_lazy_all:nT
1419       {
1420         { \bool_if_p:N \g__enumext_starred_bool }
1421         { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_one_int } }
1422       }
1423       {
1424         \int_gdecr:N \g__enumext_count_item_number_int
1425         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1426       }
1427     }
1428     { 2 }{
1429       \int_gdecr:N \g__enumext_count_item_number_int
1430     }
1431     { 3 }{
1432       \int_gdecr:N \g__enumext_count_item_number_int
1433     }
1434     { 4 }{
1435       \int_gdecr:N \g__enumext_count_item_number_int
1436     }
1437   }
1438   \int_case:nn { \__enumext_level_h_int }
1439   {
1440     { 1 }{
1441       \bool_if:NT \g__enumext_standar_bool
1442       {
1443         \int_gdecr:N \g__enumext_count_item_number_int
1444         \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}

```



```

1445         }
1446     }
1447 }
1448 }

```

(End of definition for `__enumext_check_ans_set:`.)

`__enumext_check_ans_exec:` The function `__enumext_check_ans_exec:` will count the number of times the `\item` and `\item*` commands appears per level within the `enumext` environment. The boolean variable `\l__enumext_store_ans_bool` controlled by the `no-store` key will increment the integer variable of the level counter by 1 to preserve the equality that we will use in the final comparison of the process.

```

1449 \cs_new_protected:Nn \__enumext_check_ans_exec:
1450 {
1451     \bool_if:NT \l__enumext_check_ans_bool
1452     {
1453         \__enumext_check_ans_set:
1454     }
1455 }

```

(End of definition for `__enumext_check_ans_exec:`.)

`__enumext_check_ans_show:` The function `__enumext_check_ans_show:` compares all executions of `\item` and `\item*` with the executions of `\anskey`. After the function is executed, we set the integer variables to zero.

```

1456 \cs_new_protected:Nn \__enumext_check_ans_show:
1457 {
1458     \int_compare:nNnTF
1459     { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1460     {
1461         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1462     }
1463     {
1464         \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1465     }
1466     \int_gzero:N \g__enumext_count_item_number_int
1467     \int_gzero:N \g__enumext_count_item_anskey_int
1468 }

```

(End of definition for `__enumext_check_ans_show:`.)

10.23 Keys and functions associated with storage

We add the keys `wrap-ans`, `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*`.

```

1469 \cs_set_protected:Npn \__enumext_tmp:n #1
1470 {
1471     \keys_define:nn { enumext / #1 }
1472     {
1473         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,
1474         wrap-ans .initial:n = \fbox{##1},
1475         wrap-ans .value_required:n = true,
1476         mark-ans .code:n = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1477         mark-ans .initial:n = \textasteriskcentered,
1478         mark-ans .value_required:n = true,
1479         mark-pos .choice:,
1480         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1481         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1482         mark-pos .initial:n = right,
1483         mark-pos .value_required:n = true,
1484         show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1485                 \bool_set_false:N \l__enumext_show_position_bool,
1486         show-ans .value_forbidden:n = true,
1487         show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1488                 \bool_set_false:N \l__enumext_show_answer_bool,
1489         show-pos .value_forbidden:n = true,
1490         mark-ref .code:n = \tl_set:Nn \l__enumext_mark_ref_sym_tl {##1},
1491         mark-ref .initial:n = \textasteriskcentered,
1492         mark-ref .value_required:n = true,
1493         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1494         save-ref .initial:n = false,
1495     }

```

```

1496   }
1497   \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1498   \cs_set_protected:Npn \__enumext_tmp:n #1
1499   {
1500     \keys_define:nn { enumext / #1 }
1501     {
1502       mark-pos .choice:,
1503       mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1504       mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1505       mark-pos .initial:n = right,
1506       mark-pos .value_required:n = true,
1507       show-ans .code:n = \bool_set_true:N \__enumext_show_answer_bool
1508                       \bool_set_false:N \__enumext_show_position_bool,
1509       show-ans .value_forbidden:n = true,
1510       show-pos .code:n = \bool_set_true:N \__enumext_show_position_bool
1511                       \bool_set_false:N \__enumext_show_answer_bool,
1512       show-pos .value_forbidden:n = true,
1513     }
1514   }
1515   \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1516   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1517   {
1518     \keys_define:nn { enumext / #1 }
1519     {
1520       columns* .code:n = \bool_set_true:c { \__enumext_store_columns_#2_bool }
1521                       \int_set:cn { \__enumext_store_columns_#2_int } {##1}
1522                       \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1523                       {
1524                         columns = \exp_not:v { \__enumext_store_columns_#2_int },
1525                       },,
1526       columns* .value_required:n = true,
1527       columns-sep* .code:n = \bool_set_true:c { \__enumext_store_columns_sep_#2_bool }
1528                       \dim_set:cn { \__enumext_store_columns_sep_#2_dim } {##1}
1529                       \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1530                       {
1531                         columns-sep = \exp_not:v { \__enumext_store_columns_sep_#2_dim },
1532                       },
1533       columns-sep* .value_required:n = true,
1534     }
1535   }
1536   \clist_map_inline:nn
1537   {
1538     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1539   }
1540   { \__enumext_tmp:nn #1 }

```

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

10.23.1 Function for storing content in prop list

`__enumext_store_addto_prop:n` The function `__enumext_store_addto_prop:n` stores the content in $\langle prop list \rangle$ 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 $\langle prop list \rangle$ 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.

```

1541   \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1542   \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1543   {
1544     \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }

```

```

1545     {
1546       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1547     }
1548     { #1 }
1549   }
1550   \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

10.23.2 Function for storing content in sequence

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

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

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

```

1551   \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1552   {
1553     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1554   }
1555   \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

10.23.3 Functions for storing the list structure in the sequence

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

```

1556   \cs_new_protected:Nn \__enumext_store_level_open:
1557   {
1558     \bool_if:NT \l__enumext_store_ans_bool
1559     {
1560       \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1561       {
1562         \__enumext_store_addto_seq:n
1563         {
1564           \item \begin{enumext}
1565         }
1566       }
1567       {
1568         \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1569         {
1570           \item \begin{enumext} [
1571           ]
1572         }
1573         \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1574         {
1575           ]
1576         }
1577         \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1578       }
1579     }
1580   \cs_new_protected:Nn \__enumext_store_level_close:
1581   {
1582     \bool_if:NT \l__enumext_store_ans_bool
1583     {
1584       \__enumext_store_addto_seq:n { \end{enumext} }
1585     }
1586   }

```

(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.

```

1587 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1588 {
1589   \bool_if:NT \l__enumext_store_ans_bool
1590   {
1591     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1592     {
1593       \__enumext_store_addto_seq:n
1594       {
1595         \item \mode_leave_vertical:
1596         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1597         \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1598       }
1599     }
1600     {
1601       \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1602       {
1603         \item \mode_leave_vertical:
1604         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1605         \begin{enumext*}[before={\setlength{\topsep}{0pt}},
1606       }
1607       \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1608       {
1609       ]
1610     }
1611     \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1612   }
1613 }
1614 }
1615 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1616 {
1617   \bool_if:NT \l__enumext_store_ans_bool
1618   {
1619     \__enumext_store_addto_seq:n { \end{enumext*} }
1620   }
1621 }

```

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

10.23.4 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`

```

1622 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1623 {
1624   \mode_leave_vertical:
1625   \skip_horizontal:n { -\dim_use:N #2 }
1626   \makebox[0pt][ r ]
1627   {
1628     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1629     {
1630       \tl_use:N \l__enumext_mark_answer_sym_tl
1631     }
1632   }
1633   \skip_horizontal:n { \dim_use:N #2 }
1634 }
1635 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

10.24 The command \anskey and internal label and ref

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

```

1636 \keys_define:nn { enumext / anskey }
1637 {
1638   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1639   item-sym* .value_required:n = true,
1640   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1641   item-pos* .value_required:n = true,
1642   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1643   store-brk .default:n = true,
1644   store-brk .value_forbidden:n = true,
1645 }

```

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⟩*.

```

1646 \NewDocumentCommand \anskey { d() s o +m }
1647 {
1648   \bool_if:NF \l__enumext_store_active_bool
1649   {
1650     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1651   }
1652   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
1653   {
1654     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1655   }
1656   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1657   {
1658     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1659   }
1660   \group_begin:
1661     \bool_if:NT \l__enumext_store_ans_bool
1662     {
1663       \bool_if:NT \l__enumext_check_ans_bool
1664       {
1665         \int_gincr:N \g__enumext_count_item_anskey_int
1666       }
1667       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1668     }
1669   \group_end:
1670 }

```

(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.

```

1671 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1672 {
1673   \__enumext_store_addto_prop:n {#4}
1674   \bool_if:NT \l__enumext_store_ref_key_bool
1675   {
1676     \__enumext_store_internal_ref:
1677   }
1678   \__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`.

```

1679   \tl_clear:N \l__enumext_store_anskey_arg_tl
1680   \tl_if_novalue:nF {#3}
1681   {
1682     \keys_set:nn { enumext / anskey } {#3}
1683   }
1684   \bool_lazy_and:nnT
1685   { \bool_if_p:N \l__enumext_store_columns_break_bool }
1686   { \bool_not_p:n { \l__enumext_starred_bool } }
1687   {

```

```

1688     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1689   }
1690   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

1691   \tl_if_novalue:nF {#1}
1692   {
1693     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1694     \bool_if:NT \l__enumext_starred_bool
1695     {
1696       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1697       {
1698         ( \exp_not:V \l__enumext_store_columns_join_int )
1699       }
1700     }
1701   }

```

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`.

```

1702   \bool_if:nTF {#2}
1703   {
1704     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1705     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1706     {
1707       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1708       {
1709         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1710       }
1711     }
1712     \dim_compare:nT
1713     {
1714       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1715     }
1716     {
1717       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1718       {
1719         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1720       }
1721     }
1722     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1723   }
1724   {
1725     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1726   }

```

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 $\langle sequence \rangle$.

```

1727   \bool_lazy_and:nnT
1728   { \bool_if_p:N \l__enumext_store_ref_key_bool }
1729   { \bool_if_p:N \l__enumext_hyperref_bool }
1730   {
1731     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1732     {
1733       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1734       { \exp_not:V \l__enumext_mark_ref_sym_tl }
1735     }
1736   }
1737   \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1738 }

```

(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{\langle store name : position \rangle}` and will return `1.(a).i.A`.

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

```

1739 \cs_new_protected:Nn \__enumext_store_internal_ref:
1740 {
1741   \cs_set_protected:Npn \__enumext_tmp:n ##1
1742   {

```

```

1743     \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
1744     \tl_reverse:c { l__enumext_label_copy_##1_tl }
1745     \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
1746     \tl_reverse:c { l__enumext_label_copy_##1_tl }
1747   }
1748   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1749   \cs_set:Npn \__enumext_tmp:n ##1
1750     { . \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.

```

1751   \bool_lazy_all:nT
1752   {
1753     { \bool_if_p:N \g__enumext_starred_bool }
1754     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1755   }
1756   {
1757     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1758       { \tl_use:N \l__enumext_label_copy_vii_tl }
1759   }
1760   \bool_lazy_all:nT
1761   {
1762     { \bool_if_p:N \l__enumext_standar_bool }
1763     { \bool_if_p:N \g__enumext_starred_bool }
1764     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1765   }
1766   {
1767     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1768       {
1769         \tl_use:N \l__enumext_label_copy_vii_tl
1770         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1771       }
1772   }

```

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.

```

1773   \bool_lazy_all:nT
1774   {
1775     { \bool_if_p:N \l__enumext_standar_bool }
1776     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1777     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1778     { \bool_not_p:n { \l__enumext_starred_bool } }
1779   }
1780   {
1781     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1782       {
1783         \tl_use:N \l__enumext_label_copy_i_tl
1784         \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1785       }
1786   }
1787   \cs_set:Npn \__enumext_tmp:n ##1
1788     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1789   \bool_lazy_all:nT
1790   {
1791     { \bool_if_p:N \l__enumext_standar_bool }
1792     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1793     { \bool_not_p:n { \g__enumext_starred_bool } }
1794     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1795   }
1796   {
1797     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1798       {
1799         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1800         . \tl_use:N \l__enumext_label_copy_vii_tl
1801       }
1802   }

```

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

```

1803   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1804   {
1805     \l__enumext_store_name_tl \c_colon_str

```



```

1806         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1807     }

```

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

```

1808     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1809     {
1810         \__enumext_newlabel:nn
1811         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1812         { \l__enumext_newlabel_arg_two_tl }
1813     }
1814     \l__enumext_store_write_aux_file_tl
1815 }

```

(End of definition for `__enumext_store_internal_ref:.`)

`__enumext_store_anskey_show_wrap:n`

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

```

1816 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1817 {
1818     \par
1819     \bool_if:NT \l__enumext_starred_bool
1820     {
1821         \cs_set:Nn \__enumext_level: { vii }
1822     }
1823     \__enumext_print_keyans_box:cc
1824     { \l__enumext_labelwidth_ \__enumext_level: _dim }
1825     { \l__enumext_labelsep_ \__enumext_level: _dim }
1826     \__enumext_anskey_wrapper:n { #1 }
1827 }

```

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

`__enumext_store_anskey_show_left:n`

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

```

1828 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1829 {
1830     \bool_if:NT \l__enumext_show_answer_bool
1831     {
1832         \__enumext_store_anskey_show_wrap:n { #1 }
1833     }
1834     \bool_if:NT \l__enumext_show_position_bool
1835     {
1836         \tl_set:Ne \l__enumext_mark_answer_sym_tl
1837         {
1838             \group_begin:
1839             \exp_not:N \normalfont
1840             \exp_not:N \footnotesize [ \int_eval:n
1841             {
1842                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1843             }
1844             ]
1845             \group_end:
1846         }
1847         \__enumext_store_anskey_show_wrap:n { #1 }
1848     }
1849 }

```

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

10.25 Common functions for keyans, keyans* and keyanspic

10.25.1 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`.

```

1850 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1851 {
1852   \tl_clear:N \l__enumext_store_keyans_label_tl
1853   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1854   {
1855     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1856   }
1857   {
1858     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1859   }
1860   \tl_if_novalue:nF { #1 }
1861   {
1862     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { , \c_space_tl #1 }
1863   }
1864   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1865 }

```

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

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

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

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

```

1866 \cs_new_protected:Nn \__enumext_keyans_store_ref:
1867 {
1868   \bool_if:NT \l__enumext_store_ref_key_bool
1869   {
1870     \cs_set_protected:Npn \__enumext_tmp:n ##1
1871     {
1872       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1873       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1874       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1875       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1876     }
1877     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1878     \__enumext_keyans_store_ref_aux_i:
1879   }
1880 }

```

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

```

1881 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
1882 {
1883   \bool_if:NT \g__enumext_starred_bool
1884   {
1885     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
1886   }
1887   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1888   {
1889     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1890     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
1891   }
1892   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1893   {
1894     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1895     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
1896   }
1897   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
1898   {
1899     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1900     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
1901   }
1902   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1903   {
1904     \l__enumext_store_name_tl \c_colon_str

```

```

1905         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1906     }
1907     \__enumext_keyans_store_ref_aux_ii:
1908 }

```

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

```

1909 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
1910 {
1911     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1912     {
1913         \__enumext_newlabel:nn
1914         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1915         { \l__enumext_newlabel_arg_two_tl }
1916     }
1917     \l__enumext_store_write_aux_file_tl
1918 }

```

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

10.25.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

1919 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1920 {
1921     \tl_clear:N \l__enumext_store_keyans_label_tl
1922     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1923     {
1924         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1925     }
1926     {
1927         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1928     }
1929     \tl_if_novalue:nF { #1 }
1930     {
1931         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { , \c_space_tl #1 }
1932     }
1933     \__enumext_keyans_addto_seq_link:
1934 }

```

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_keyans_check_ans:nn` and increment the value of the integer variable `\g__enumext_count_item_anskey_int` handled by the `check-ans` key.

```

1935 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
1936 {
1937     \bool_lazy_and:nnT
1938     { \bool_if_p:N \l__enumext_store_ref_key_bool }
1939     { \bool_if_p:N \l__enumext_hyperref_bool }
1940     {
1941         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
1942         {
1943             \hfill \exp_not:N \hyperlink
1944             {
1945                 \exp_not:V \l__enumext_newlabel_arg_one_tl
1946             }
1947             { \exp_not:V \l__enumext_mark_ref_sym_tl }
1948         }
1949     }
1950     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
1951     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1952     \bool_if:NT \l__enumext_check_ans_bool
1953     {
1954         \int_gincr:N \g__enumext_count_item_anskey_int
1955     }
1956 }

```

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

10.25.4 Check for starred commands

`__enumext_keyans_check_ans:nn`

The function `__enumext_keyans_check_ans:nn` performs an extra check for the `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.

```

1957 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1958 {
1959   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1960   {
1961     \msg_warning:nnnn { enumext } { missing-starred } { #1 } { #2 }
1962   }
1963   { \tl_gclear:N \g__enumext_check_ans_item_tl }
1964 }

```

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

10.25.5 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`

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_` `__enumext_store_name_tl` `_prop` for `show-pos` key.

```

1965 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1966 {
1967   \bool_if:NT \l__enumext_show_answer_bool
1968   {
1969     \tl_put_left:Nn \l__enumext_label_v_tl
1970     {
1971       \__enumext_print_keyans_box:NN
1972       \l__enumext_labelwidth_i_dim
1973       \l__enumext_labelsep_i_dim
1974     }
1975     \tl_if_novalue:nF { #1 }
1976     {
1977       \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] }
1978     }
1979   }
1980   \bool_if:NT \l__enumext_show_position_bool
1981   {
1982     \tl_set:Nc \l__enumext_mark_answer_sym_tl
1983     {
1984       \group_begin:
1985       \exp_not:N \normalfont
1986       \exp_not:N \footnotesize [ \int_eval:n
1987       {
1988         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
1989       }
1990       ]
1991       \group_end:
1992     }
1993     \tl_put_left:Nn \l__enumext_label_v_tl
1994     {
1995       \__enumext_print_keyans_box:NN
1996       \l__enumext_labelwidth_i_dim
1997       \l__enumext_labelsep_i_dim
1998     }
1999     \tl_if_novalue:nF { #1 }
2000     {
2001       \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] }
2002     }
2003   }
2004 }

```

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

10.26 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*
2005 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2006 {
2007   \keys_define:nn { enumext / #1 }
2008   {
2009     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2010     item-sym* .value_required:n = true,
2011     item-sym* .initial:n = { $\star$ },
2012     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2013     item-pos* .value_required:n = true,
2014   }
2015 }
2016 \clist_map_inline:nn
2017 {
2018   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2019 }
2020 { \__enumext_tmp:nn #1 }
```

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

10.27 Redefining \footnote command

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

```

2021 \cs_new_protected:Nn \__enumext_footnotetext:nn
2022 {
2023   \footnotetext[#1]{#2}
2024 }
2025 \cs_new_protected:Nn \__enumext_renew_footnote:
2026 {
2027   \seq_gclear:N \g__enumext_footnote_arg_seq
2028   \seq_gclear:N \g__enumext_footnote_int_seq
2029   \RenewDocumentCommand \footnote { o +m }
2030   {
2031     \tl_if_novalue:nTF {##1}
2032     {
2033       \stepcounter{footnote}
2034       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2035     }
2036     {
2037       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2038     }
2039     \footnotemark [ \g__enumext_footnote_int ]
2040     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2041     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2042   }
2043 }
2044 \cs_new_protected:Nn \__enumext_print_footnote:
2045 {
2046   \seq_if_empty:NF \g__enumext_footnote_int_seq
2047   {
2048     \seq_map_pairwise_function:NNN
2049     \g__enumext_footnote_int_seq
2050     \g__enumext_footnote_arg_seq
2051     \__enumext_footnotetext:nn
2052   }
2053 }
```

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

10.28 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.

10.28.1 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: (§10.29)`.

```

2054 \cs_new_protected:Npn \__enumext_default_item:n #1
2055 {
2056   \tl_if_novalue:nTF {#1}
2057   {
2058     \bool_if:NT \l__enumext_check_ans_bool
2059     {
2060       \int_gincr:N \g__enumext_count_item_number_int
2061     }
2062     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2063     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2064   }
2065   {
2066     \bool_set_eq:cc
2067     { \l__enumext_wrap_label_ \__enumext_level: _bool }
2068     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2069     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2070   }
2071 }

```

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

`__enumext_starred_item:nn`

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

Then we will see the state of the variable `\l__enumext_check_ans_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: (§10.29)`.

```

2072 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2073 {
2074   \tl_if_novalue:nF {#1}
2075   {
2076     \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2077   }
2078   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2079   \tl_if_novalue:nTF {#2}
2080   {
2081     \dim_set_eq:cc
2082     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2083     { \l__enumext_labelsep_ \__enumext_level: _dim }
2084   }
2085   {
2086     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2087   }
2088   \bool_if:NT \l__enumext_check_ans_bool
2089   {
2090     \int_gincr:N \g__enumext_count_item_number_int

```

```

2091     }
2092     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2093     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2094 }

```

(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).

```

2095 \cs_new_protected:Nn \__enumext_redefine_item:
2096 {
2097   \RenewDocumentCommand \item { s o o }
2098   {
2099     \bool_if:nTF {##1}
2100     {
2101       \__enumext_starred_item:nn {##2} {##3}
2102     }
2103     { \__enumext_default_item:n {##2} }
2104   }
2105 }

```

(End of definition for __enumext_redefine_item:.)

10.28.2 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`.

```

2106 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2107 {
2108   \tl_if_novalue:nTF { #1 }
2109   {
2110     \bool_set_true:N \__enumext_wrap_label_v_bool
2111     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2112   }
2113   {
2114     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2115     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2116   }
2117 }

```

(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 $\langle label \rangle$.

```

2118 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2119 {
2120   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2121   \__enumext_keyans_show_left:n { #1 }
2122   \bool_set_true:N \__enumext_wrap_label_v_bool
2123   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```

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 $\langle sequence \rangle$.

```

2124   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2125   \__enumext_keyans_addto_prop:n { #1 }
2126   \__enumext_keyans_store_ref:
2127   \__enumext_keyans_addto_seq:n { #1 }
2128 }

```

(End of definition for __enumext_keyans_starred_item:n.)

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.31).

```

2129 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2130 {
2131   \RenewDocumentCommand \item { s o }
2132   {
2133     \bool_if:nTF {##1}
2134     {
2135       \peek_remove_spaces:n
2136       {
2137         \__enumext_keyans_starred_item:n {##2}
2138       }
2139     }
2140     {
2141       \__enumext_keyans_default_item:n {##2}
2142     }
2143   }
2144 }

```

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

10.29 Redefining `\makeLabel` command

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

10.29.1 Redefining `\makeLabel` for `enumext`

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

```

2145 \cs_new_protected:Nn \__enumext_item_starred:
2146 {
2147   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2148   {
2149     \mode_leave_vertical:
2150     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2151     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2152     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2153   }
2154 }

```

(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).

```

2155 \cs_new_protected:Nn \__enumext_make_label:
2156 {
2157   \RenewDocumentCommand \makeLabel { m }
2158   {
2159     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2160     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2161     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2162     {
2163       \__enumext_item_starred:
2164       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2165     }
2166     { ##1 }
2167     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2168     \tl_gclear:N \g__enumext_item_symbol_tl
2169   }
2170 }

```

(End of definition for `__enumext_make_label:`.)

10.29.2 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).

```

2171 \cs_new_protected:Nn \__enumext_keyans_make_label:
2172 {
2173   \RenewDocumentCommand \makeLabel { m }
2174   {
2175     \tl_use:N \l__enumext_label_fill_left_v_tl
2176     \tl_use:N \l__enumext_label_font_style_v_tl
2177     \bool_if:NTF \l__enumext_wrap_label_v_bool
2178     {
2179       \__enumext_wrapper_label_v:n { ##1 }
2180     }
2181     { ##1 }
2182     \tl_use:N \l__enumext_label_fill_right_v_tl
2183   }
2184 }

```

(End of definition for `__enumext_keyans_make_label:`)

10.30 Calculation of `\leftmargin` and `\itemindent`

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

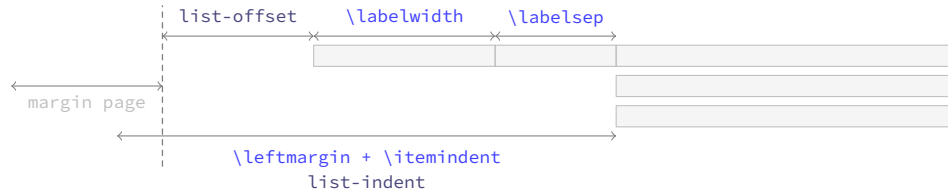


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

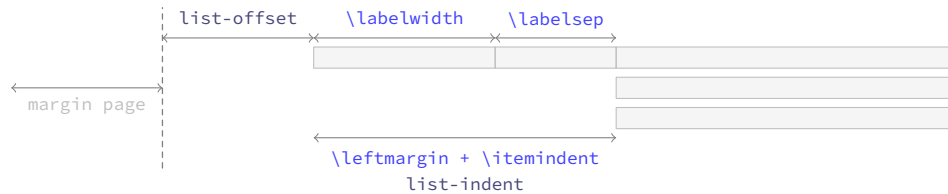


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

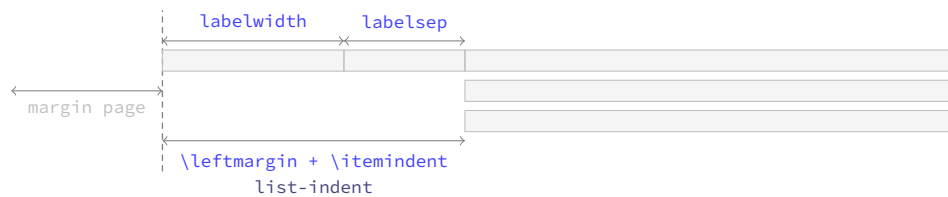


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNN
\__enumext_calc_hspace:cccccc

```

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

#1: <code>\l__enumext_labelwidth_X_dim</code>	#2: <code>\l__enumext_labelsep_X_dim</code>
#3: <code>\l__enumext_listoffset_X_dim</code>	#4: <code>\l__enumext_leftmargin_tmp_X_dim</code>
#5: <code>\l__enumext_leftmargin_X_dim</code>	#6: <code>\l__enumext_itemindent_X_dim</code>
#7: <code>\l__enumext_leftmargin_tmp_X_bool</code>	

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).

```

2185 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
2186 {
2187   \dim_compare:nNnT { #1 } < { \c_zero_dim }

```

```

2188     {
2189       \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2190       \dim_set:Nn #1 { \dim_abs:n { #1 } }
2191     }
2192     \dim_compare:nNnT { #2 } < { \c_zero_dim }
2193     {
2194       \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2195       \dim_set:Nn #2 { \dim_abs:n { #2 } }
2196     }

```

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

```

2197     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2198     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2199     {
2200       \dim_set:Nn #6 { #1 + #2 - #4 }
2201       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2202     }
2203     {
2204       \dim_compare:nNnT { #4 } = { #1 + #2 }
2205       { \dim_set:Nn #6 { \c_zero_dim } }
2206       \dim_compare:nNnT { #4 } < { #1 + #2 }
2207       { \dim_set:Nn #6 { #1 + #2 - #4 } }
2208       \dim_compare:nNnT { #4 } > { #1 + #2 }
2209       {
2210         \dim_set:Nn #6 { -#1 - #2 + #4 }
2211         \dim_set:Nn #6 { #6*-1 }
2212       }
2213       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2214     }
2215   }
2216   \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

10.31 Setting second argument of the lists

At this point of the code we have already programmed 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`.

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments. 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.

```

2217   \cs_set_protected:Npn \__enumext_tmp:n #1
2218   {
2219     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2220     {
2221       \__enumext_calc_hspace:cccccc
2222       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2223       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2224       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2225       { \__enumext_leftmargin_tmp_#1_bool }
2226       \clist_map_inline:nn
2227       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2228       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2229       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2230       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2231       \usecounter { enumX#1 }
2232       \bool_lazy_and:nnTF
2233       { \str_if_eq_p:nn {#1} { i } }
2234       { \bool_if_p:N \__enumext_resume_bool }
2235       { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2236       {
2237         \setcounter { enumX#1 }
2238         { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2239       }
2240       \str_if_eq:nnTF {#1} { v }

```

```

2241     {
2242         \__enumext_keyans_redefine_item:
2243         \__enumext_keyans_make_label:
2244         \__enumext_keyans_fake_item:
2245         \bool_if:cT { \__enumext_show_length_#1_bool }
2246         {
2247             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2248         }
2249     }
2250     {
2251         \__enumext_redefine_item:
2252         \__enumext_make_label:
2253         \__enumext_use_key_ref:
2254         \__enumext_fake_item:
2255         \bool_if:cT { \__enumext_show_length_#1_bool }
2256         {
2257             \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_i
2258         }
2259     }
2260 }
2261 }
2262 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

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

```

2263 \cs_set_protected:Npn \__enumext_tmp:n #1
2264 {
2265     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2266     {
2267         \__enumext_calc_hspace:ccccc
2268         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2269         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2270         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2271         { \__enumext_leftmargin_tmp_#1_bool }
2272         \clist_map_inline:nn
2273         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2274         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2275         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2276         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2277         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2278         \skip_zero:N \partopsep
2279         \usecounter { enumX#1 }
2280         \bool_lazy_and:nnTF
2281         { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2282         { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2283         {
2284             \setcounter { enumX#1 }
2285             { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2286         }
2287         \__enumext_use_key_ref_h:
2288         \str_if_eq:nnTF {#1} { vii }
2289         {
2290             \__enumext_fake_item_vii:
2291             \bool_if:cT { \__enumext_show_length_vii_bool }
2292             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2293         }
2294         {
2295             \__enumext_fake_item_viii:
2296             \bool_if:cT { \__enumext_show_length_#1_bool }
2297             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2298         }
2299     }
2300 }
2301 \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.

```

2302 \NewDocumentEnvironment{enumext}{0}{ }
2303 {
2304   \__enumext_current_env:
2305   \__enumext_safe_exec:
2306   \__enumext_parse_keys:n {#1}
2307   \__enumext_before_list:
2308   \__enumext_start_store_level:
2309   \__enumext_start_list:nn
2310   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2311   {
2312     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2313     \__enumext_before_keys_exec:
2314   }
2315   \__enumext_after_args_exec:
2316 }
2317 {
2318   \__enumext_stop_list:
2319   \__enumext_stop_store_level:
2320   \__enumext_after_list:
2321 }

```

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

`__enumext_safe_exec:` First check the maximum nesting level for the `enumext` environment and set the state of the booleans `\l__enumext_standar_bool` and `\g__enumext_standar_bool` to “true”, the latter only if the environment is NOT nested in the `enumext*` environment.

```

2322 \cs_new_protected:Nn \__enumext_safe_exec:
2323 {
2324   \int_incr:N \l__enumext_level_int
2325   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2326   { \msg_fatal:nn { enumext } { list-too-deep } }
2327   \bool_set_true:N \l__enumext_standar_bool
2328 }

```

(End of definition for `__enumext_safe_exec:`.)

`__enumext_parse_keys:n` Parse [`<key = val>`] by levels in `enumext`. If the variable `\l__enumext_store_active_bool` is true it will call the function `__enumext_parse_store_keys:n` and reprocess the `<keys>` to pass them to the storage sequence.

```

2329 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2330 {
2331   \exp_args:Ne \keys_set:nn
2332   { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2333   \bool_if:NT \l__enumext_store_active_bool
2334   {
2335     \__enumext_parse_store_keys:n {#1}
2336   }
2337 }

```

(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:`.

```

2338 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2339 {
2340   \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2341   {
2342     \regex_match:nnT { \b columns\b } {#1}
2343     {
2344       \int_set_eq:cc
2345       { l__enumext_store_columns_ \__enumext_level: _int }
2346       { l__enumext_columns_ \__enumext_level: _int }
2347       \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2348       {
2349         columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2350       }
2351     }
2352   }

```

```

2351     }
2352   }
2353   \bool_if:cF { \__enumext_store_columns_sep_ \__enumext_level: _bool }
2354   {
2355     \regex_match:nnT { \b columns-sep \b } {#1}
2356     {
2357       \dim_set_eq:cc
2358       { \__enumext_store_columns_sep_ \__enumext_level: _dim }
2359       { \__enumext_columns_sep_ \__enumext_level: _dim }
2360       \tl_put_right:ce { \__enumext_store_opt_ \__enumext_level: _tl }
2361       {
2362         columns-sep = \exp_not:v { \__enumext_store_columns_sep_ \__enumext_level: _dim }
2363       }
2364     }
2365   }
2366 }

```

(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. If enumext are nested in enumext* add __enumext_store_level_open: to preserve the stored structure.

```

2367 \cs_new_protected:Nn \__enumext_start_store_level:
2368 {
2369   \bool_lazy_all:nT
2370   {
2371     { \bool_if_p:N \__enumext_store_active_bool }
2372     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2373     { \bool_not_p:n { \g__enumext_starred_bool } }
2374   }
2375   {
2376     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2377     {
2378       \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
2379       \__enumext_store_level_open:
2380     }
2381   }
2382   \bool_lazy_all:nT
2383   {
2384     { \bool_if_p:N \__enumext_store_active_bool }
2385     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2386     { \bool_if_p:N \g__enumext_starred_bool }
2387   }
2388   {
2389     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2390     {
2391       \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
2392       \__enumext_store_level_open:
2393     }
2394   }
2395 }
2396 \cs_new_protected:Nn \__enumext_stop_store_level:
2397 {
2398   \bool_if:cT { \__enumext_store_upper_level_ \__enumext_level: _bool }
2399   {
2400     \__enumext_store_level_close:
2401   }
2402 }

```

(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.

```

2403 \cs_new_protected:Nn \__enumext_before_list:
2404 {
2405   \__enumext_vspace_above:
2406   \__enumext_before_args_exec:

```

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

```

2407   \__enumext_check_ans_exec:

```

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.

```

2408 \dim_compare:nNtT
2409 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2410 {
2411   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2412   {
2413     \linewidth
2414     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2415     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2416   }

```

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

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

```

2417 \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2418 \int_gincr:N \g__enumext_minipage_stat_int
2419 \__enumext_mini_addvspace:
2420 \nointerlineskip\noindent
2421 \begin{__enumext_mini_env*}
2422 { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2423 }
2424 \__enumext_multicols_start:
2425 }

```

(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.

```

2426 \cs_new_protected:Nn \__enumext_multicols_start:
2427 {
2428   \int_compare:nNtT
2429   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2430   {
2431     \dim_compare:nNtT
2432     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2433     {
2434       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2435       {
2436         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2437         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2438         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2439         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2440       }
2441     }
2442     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2443     \skip_zero:N \multicolsep
2444     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2445     {
2446       \dim_zero:N \columnseprule
2447     }

```

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.

```

2448 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2449 {
2450   \__enumext_multi_addvspace:
2451 }
2452 \raggedcolumns

```



```

2453     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2454 }
2455 }

```

(End of definition for __enumext_multicols_start:.)

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

```

2456 \cs_new_protected:Nn \__enumext_multicols_stop:
2457 {
2458   \int_compare:nNtT
2459     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2460     {
2461       \end{multicols}
2462       \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2463       {
2464         \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2465       }
2466     }

```

If the `check-ans` key is active, we set the boolean variable \g__enumext_check_ans_show_bool to true and copy the stored name to the variable \g__enumext_store_name_tl. These variables will be used by the function __enumext_after_env:n to display the result of the internal check answer mechanism in the terminal.

```

2467   \bool_lazy_and:nnT
2468     { \bool_if_p:N \__enumext_check_ans_bool }
2469     { \bool_not_p:n { \g__enumext_starred_bool } }
2470     {
2471       \bool_gset_true:N \g__enumext_check_ans_show_bool
2472       \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2473     }
2474 }

```

(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.

```

2475 \cs_new_protected:Nn \__enumext_after_list:
2476 {
2477   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2478   {
2479     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2480     {
2481       \msg_warning:nn { enumext } { missing-miniright }
2482       \miniright
2483     }
2484     \int_gzero:N \g__enumext_minipage_stat_int
2485     \end{__enumext_mini_env*}
2486     \par\addvspace { \l__enumext_minipage_after_skip }
2487   }
2488   { \__enumext_multicols_stop: }

```

Now apply the `{⟨code⟩}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

2489   \__enumext_after_stop_list:
2490   \__enumext_vspace_below:

```

Finally save the *current value* of the counter in \g__enumext_resume_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.

```

2491   \bool_set_false:N \l__enumext_standar_bool
2492   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2493   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2494   {
2495     \int_gset_eq:cN
2496       { g__enumext_resume_ \l__enumext_store_name_tl _int }
2497       { \value{enumXi} }

```

```

2498     }
2499 }

```

(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`.

```

2500 \__enumext_after_env:nn {enumext}
2501 {
2502   \int_compare:nNtT { \__enumext_level_int } = { 0 }
2503   {
2504     \bool_if:NT \g__enumext_check_ans_show_bool
2505     {
2506       \__enumext_check_ans_show:
2507     }
2508     \bool_gset_false:N \g__enumext_standar_bool
2509     \bool_gset_false:N \g__enumext_check_ans_show_bool
2510     \tl_gclear:N \g__enumext_store_name_tl
2511   }
2512 }

```

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.

```

2513 \NewDocumentEnvironment{keyans}{ 0{} }
2514 {
2515   \__enumext_keyans_safe_exec:
2516   \__enumext_keyans_parse_keys:n {#1}
2517   \__enumext_before_list_v:
2518   \__enumext_start_list:nn
2519   { \tl_use:N \l__enumext_label_v_tl }
2520   {
2521     \__enumext_list_arg_two_v:
2522     \__enumext_before_keys_exec_v:
2523   }
2524   \__enumext_after_args_exec_v:
2525 }
2526 {
2527   \__enumext_keyans_check_ans:nn { item }{ keyans }
2528   \__enumext_stop_list:
2529   \__enumext_after_list_v:
2530 }

```

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

`__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.

```

2531 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2532 {
2533   \bool_if:NF \l__enumext_store_active_bool
2534   {
2535     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2536   }
2537   \int_incr:N \l__enumext_keyans_level_int
2538   \bool_set_true:N \l__enumext_keyans_env_bool
2539   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2540   \bool_set_false:N \l__enumext_store_active_bool
2541   \int_compare:nNtT { \l__enumext_keyans_level_int } > { 1 }
2542   {
2543     \msg_error:nn { enumext } { keyans-nested }
2544   }
2545   \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2546   {
2547     \msg_error:nn { enumext } { keyans-wrong-level }
2548   }
2549 }

```

(End of definition for `__enumext_keyans_safe_exec:`.)

```
\__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
2550 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2551 {
2552   \keys_set:nn { enumext / keyans } {#1}
2553 }
```

(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.

```
2554 \cs_new_protected:Nn \__enumext_before_list_v:
2555 {
2556   \__enumext_vspace_above_v:
2557   \__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.

```
2558   \dim_compare:nNtT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2559   {
2560     \dim_set:Nn \l__enumext_minipage_left_v_dim
2561     {
2562       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2563     }
2564   }
```

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

```
2564   \bool_set_true:N \l__enumext_minipage_active_v_bool
2565   \int_gincr:N \g__enumext_minipage_stat_int
2566   \__enumext_keyans_mini_addvspace:
2567   \nointerlineskip\noindent
2568   \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2569 }
```

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

```
2570   \__enumext_keyans_multicols_start:
2571 }
```

(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.

```
2572 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2573 {
2574   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
2575   {
```

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

```
2576     \dim_compare:nNtT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2577     {
2578       \dim_set:Nn \l__enumext_columns_sep_v_dim
2579       {
2580         (
2581           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2582         ) / \l__enumext_columns_v_int
2583         - \l__enumext_listoffset_v_dim
2584       }
2585     }
2586     \dim_set_eq:NN \columnsep \l__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).

```
2587     \skip_zero:N \multicolsep
2588     \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.

```

2589     \bool_if:NF \l__enumext_minipage_active_v_bool
2590     {
2591         \__enumext_keyans_multi_addvspace:
2592     }
2593     \raggedcolumns
2594     \begin{multicols}{\l__enumext_columns_v_int }
2595 }
2596 }

```

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

```

2597 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2598 {
2599     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2600     {
2601         \end{multicols}
2602         \bool_if:NF \l__enumext_minipage_active_v_bool
2603         {
2604             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2605         }
2606     }
2607 }

```

(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 `\l__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 `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

2608 \cs_new_protected:Nn \__enumext_after_list_v:
2609 {
2610     \bool_if:NTF \l__enumext_minipage_active_v_bool
2611     {
2612         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2613         {
2614             \msg_warning:nn { enumext } { missing-miniright }
2615             \miniright
2616         }
2617         \int_gzero:N \g__enumext_minipage_stat_int
2618         \end{\__enumext_mini-env*}
2619         \par\addvspace{ \l__enumext_minipage_after_skip }
2620     }
2621     { \__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.

```

2622     \bool_set_false:N \l__enumext_keyans_env_bool
2623     \__enumext_after_stop_list_v:
2624     \__enumext_vspace_below_v:
2625 }

```

(End of definition for `__enumext_after_list_v:`)

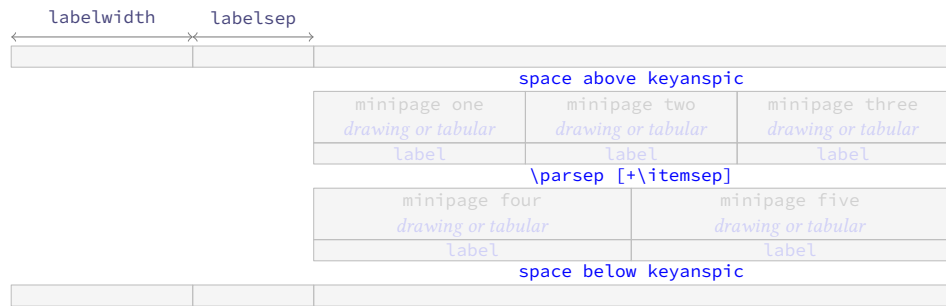
10.34 The environment `keyanspic` and `\anspic`

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.

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?](#).

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

10.34.1 The command `\anspic`

`\anspic` The `\anspic` command take three arguments, the starred (*) versions `\anspic*` and `\anspic*[\langle content \rangle]` store the current `\label` 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. This command is used as a replacement for `\item` in the `keyanspic` environment.

```
2626 \NewDocumentCommand \anspic { s o +m }
2627 {
```

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

```
2628   \bool_if:NF \l__enumext_store_active_bool
2629   {
2630     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2631   }
2632   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2633   {
2634     \msg_error:nn { enumext } { keyanspic-wrong-level }
2635   }
2636   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2637   {
2638     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2639   }
```

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.

```
2640   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2641   {
2642     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2643   }
2644 }
```

(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.

```
2645 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2646 {
2647   \stepcounter { enumXvi }
2648   #3 \\\
2649   \bool_if:nT { #1 }
2650   {
2651     \__enumext_keyans_addto_prop:n { #2 }
2652     \__enumext_keyans_store_ref:
2653     \__enumext_keyans_addto_seq:n { #2 }
2654     \bool_lazy_or:nnT
2655     { \bool_if_p:N \l__enumext_show_answer_bool }
2656     { \bool_if_p:N \l__enumext_show_position_bool }
2657     {
2658       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2659       \__enumext_keyans_show_left:n { #2 }
2660       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2661     }
2662   }
2663   \tl_use:N \l__enumext_label_font_style_v_tl
2664   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
2665 }
```

(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.

```

2666 \NewDocumentEnvironment{keyanspic}{ o }
2667 {
2668   \__enumext_keyans_pic_safe_exec:
2669   \__enumext_start_list:nn
2670   { }
2671   {
2672     \__enumext_keyans_pic_arg_two:
2673   }

```

We apply the “adjusted” vertical spacing above the environment

```

2674   \vspace { \__enumext_keyans_pic_above_skip }
2675 }

```

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.

```

2676 {
2677   \tl_if_novalue:nTF { #1 }
2678   {
2679     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2680   }
2681   { \__enumext_keyans_pic_do:n { #1 } }
2682   \__enumext_stop_list:
2683   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2684   \setcounter { enumXvi } { 0 }
2685   \vspace { \__enumext_topsep_v_skip }
2686   %\bool_set_false:N \l__enumext_store_active_bool
2687 }

```

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

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

```

2688 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2689 {
2690   \int_incr:N \l__enumext_keyans_pic_level_int
2691   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2692   {
2693     \msg_error:nn { enumext } { keyanspic-nested }
2694   }
2695 }

```

(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`.

```

2696 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2697 {
2698   \dim_compare:nNnT { #1 } < { 0pt }
2699   { \skip_set:Nn #1 { -#1 } }
2700 }

```

(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.

```

2701 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2702 {

```

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.

```

2703   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2704   \__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.

```

2705 \skip_add:Nn \parsep { \itemsep }
2706 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2707 \dim_zero:N \labelwidth
2708 \dim_zero:N \listparindent
2709 \dim_zero:N \labelsep
2710 \skip_zero:N \partopsep
2711 \skip_zero:N \itemsep

```

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

```

2712 \__enumxt_keyans_pic_skip_abs:N \parsep
2713 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
2714 {
2715     \box_dp:N \strutbox
2716     + \l__enumxt_topsep_v_skip
2717     - \parsep
2718 }
2719 \__enumxt_item_std:w \scan_stop:
2720 }

```

(End of definition for `__enumxt_keyans_pic_arg_two:`)

```

\__enumxt_keyans_pic_do:n
\__enumxt_keyans_pic_do:e

```

The optional argument is split by comma and is handled directly by the function `__enumxt_keyans_pic_do:n` and passed to the function `__enumxt_keyans_pic_row:n`.

```

2721 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
2722 {
2723     \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
2724 }
2725 \cs_generate_variant:Nn \__enumxt_keyans_pic_do:n { e }

```

(End of definition for `__enumxt_keyans_pic_do:n`)

```
\__enumxt_keyans_pic_row:n
```

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

```

2726 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
2727 {
2728     \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
2729     \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
2730     \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
2731     \int_step_inline:nnn
2732     { \l__enumxt_keyans_pic_above_int + 1 }
2733     { \l__enumxt_keyans_pic_below_int }
2734     {
2735         \__enumxt_minipage:w [ b ]{ \l__enumxt_keyans_pic_width_dim }
2736         \centering
2737         \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
2738         \__enumxt_endminipage:
2739     }
2740     \par
2741 }

```

(End of definition for `__enumxt_keyans_pic_row:n`)

10.35 The enumxt* environment

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

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

10.35.1 Functions for item box width

_enumext_starred_columns_set_vii:

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

```

2742 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2743 {
2744   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2745   {
2746     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2747     {
2748       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2749       / \l__enumext_columns_vii_int
2750     }
2751   }
2752   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2753   \dim_set:Nn \l__enumext_item_width_vii_dim
2754   {
2755     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2756     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2757     - \l__enumext_labelsep_vii_dim
2758   }
2759   \dim_zero_new:N \itemwidth
2760 }

```

(End of definition for _enumext_starred_columns_set_vii:.)

_enumext_starred_joined_item_vii:n

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

```

2761 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2762 {
2763   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2764   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2765   {
2766     \msg_warning:nnee { enumext } { item-joined }
2767     { \int_use:N \l__enumext_joined_item_vii_int }
2768     { \int_use:N \l__enumext_columns_vii_int }
2769     \int_set:Nn \l__enumext_joined_item_vii_int
2770     {
2771       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2772     }
2773   }
2774   \int_compare:nNnT
2775   { \l__enumext_joined_item_vii_int }
2776   >
2777   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2778   {
2779     \msg_warning:nnee { enumext } { item-joined-columns }
2780     { \int_use:N \l__enumext_joined_item_vii_int }
2781     {
2782       \int_eval:n
2783       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2784     }
2785     \int_set:Nn \l__enumext_joined_item_vii_int
2786     {
2787       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2788     }
2789   }

```

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

```

2790   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2791   {
2792     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2793     \int_decr:N \l__enumext_joined_item_aux_vii_int
2794     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2795     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2796     \dim_set:Nn \l__enumext_joined_width_vii_dim
2797     {
2798       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2799       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2800         + \l__enumext_columns_sep_vii_dim
2801         ) * \l__enumext_joined_item_aux_vii_int

```

```

2802     }
2803     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2804   }
2805   {
2806     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2807     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2808   }
2809 }

```

(End of definition for `__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`.

```

2810 \cs_new_protected:Nn \__enumext_start_mini_vii:
2811 {
2812   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2813   {
2814     \dim_set:Nn \l__enumext_minipage_left_vii_dim
2815     {
2816       \linewidth
2817       - \l__enumext_minipage_right_vii_dim
2818       - \l__enumext_minipage_hsep_vii_dim
2819     }
2820     \bool_set_true:N \l__enumext_minipage_active_vii_bool
2821     \dim_gset_eq:NN
2822       \g__enumext_minipage_right_vii_dim
2823       \l__enumext_minipage_right_vii_dim
2824     \__enumext_mini_addvspace_vii:
2825     \nointerlineskip\nindent
2826     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2827   }
2828 }

```

(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:nn` to execute the `__enumext_mini_env*` on the “right side”.

```

2829 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2830 {
2831   \bool_if:NT \l__enumext_minipage_active_vii_bool
2832   {
2833     \end{__enumext_mini_env*}
2834     \hfill
2835     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2836   }
2837 }

```

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”.

```

2838 \__enumext_after_env:nn {enumext*}
2839 {
2840   \bool_if:NT \g__enumext_minipage_active_vii_bool
2841   {
2842     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2843     \par\addvspace { \g__enumext_minipage_right_skip }
2844     \bool_if:NF \g__enumext_minipage_center_vii_bool
2845     {
2846       \centering
2847     }
2848     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2849     \end{__enumext_mini_env*}
2850     \par\addvspace{ \g__enumext_minipage_after_skip }
2851   }
2852   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2853   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2854   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2855   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2856 }

```

(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.

```

2857 \NewDocumentEnvironment{enumext*}{o }
2858 {
2859   \__enumext_current_env:
2860   \__enumext_safe_exec_vii:
2861   \__enumext_parse_keys_vii:n {#1}
2862   \__enumext_before_list_vii:
2863   \__enumext_start_store_level_vii:
2864   \__enumext_start_list:nn { }
2865   {
2866     \__enumext_list_arg_two_vii:
2867     \__enumext_before_keys_exec_vii:
2868   }
2869   \__enumext_starred_columns_set_vii:
2870   \item[] \scan_stop:
2871   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2872   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2873 }
2874 {
2875   \__enumext_stop_item_tmp_vii:
2876   \__enumext_remove_extra_parsep_vii:
2877   \__enumext_stop_list:
2878   \__enumext_stop_store_level_vii:
2879   \__enumext_after_list_vii:
2880 }

```

(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`.

```

2881 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2882 {
2883   \int_incr:N \l__enumext_level_h_int
2884   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2885   {
2886     \msg_error:nn { enumext } { nested }
2887   }
2888   \bool_set_true:N \l__enumext_starred_bool
2889 }

```

(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.

```

2890 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2891 {
2892   \tl_if_novalue:nF {#1}
2893   {
2894     \keys_set:nn { enumext / enumext* } {#1}
2895     \bool_if:NT \l__enumext_store_active_bool
2896     {
2897       \__enumext_parse_store_keys_vii:n {#1}
2898     }
2899   }
2900 }

```

(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:`.

```

2901 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2902 {

```

```

2903 \bool_if:NF \l__enumext_store_columns_vii_bool
2904 {
2905     \regex_match:nnT { \b columns\b } {#1}
2906     {
2907         \int_set_eq:NN
2908             \l__enumext_store_columns_vii_int
2909             \l__enumext_columns_vii_int
2910         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2911         {
2912             columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2913         }
2914     }
2915 }
2916 \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2917 {
2918     \regex_match:nnT { \b columns-sep\b } {#1}
2919     {
2920         \dim_set_eq:NN
2921             \l__enumext_store_columns_sep_vii_dim
2922             \l__enumext_columns_sep_vii_dim
2923         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2924         {
2925             columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2926         }
2927     }
2928 }
2929 }

```

(End of definition for \l__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`.

```

2930 \cs_new_protected:Nn \__enumext_before_list_vii:
2931 {
2932     \__enumext_vspace_above_vii:
2933     \__enumext_check_ans_exec: % need by chek-ans
2934     \__enumext_before_args_exec_vii:
2935     \__enumext_start_mini_vii:
2936 }

```

(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.

```

2937 \cs_new_protected:Nn \__enumext_after_list_vii:
2938 {
2939     \__enumext_stop_mini_vii:
2940     \__enumext_after_stop_list_vii:
2941     \__enumext_vspace_below_vii:
2942     \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2943     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2944     {
2945         \int_gset_eq:cN
2946             { g__enumext_resume_ \l__enumext_store_name_tl _int }
2947         { \value{enumXvii} }
2948     }
2949     \bool_lazy_and:nnT
2950     { \bool_if_p:N \g__enumext_starred_bool }
2951     { \bool_if_p:N \l__enumext_check_ans_bool }
2952     {
2953         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2954         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2955     }
2956     %\bool_gset_false:N \g__enumext_starred_bool
2957     \bool_set_false:N \l__enumext_starred_bool
2958 }

```

(End of definition for `__enumext_after_list_vii:`)

`__enumext_start_store_level_vii:`
`__enumext_stop_store_level_vii:`

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

```

2959 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2960 {
2961   \bool_if:NT \l__enumext_store_active_bool
2962   {
2963     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2964     {
2965       \__enumext_store_level_open_vii:
2966     }
2967   }
2968 }
2969 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2970 {
2971   \bool_if:NT \l__enumext_store_active_bool
2972   {
2973     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2974     {
2975       \__enumext_store_level_close_vii:
2976     }
2977   }
2978 }

```

(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`.

```

2979 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2980 {
2981   \__enumext_stop_item_tmp_vii:
2982   \int_incr:N \l__enumext_item_column_pos_vii_int
2983   \int_gincr:N \g__enumext_item_count_all_vii_int
2984   \__enumext_item_peek_args_vii:
2985 }

```

(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).

```

2986 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2987 {
2988   \peek_meaning:NTF (
2989     { \__enumext_joined_item_vii:w }
2990     { \__enumext_joined_item_vii:w (1) }
2991   }

```

(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`.

```

2992 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2993 {
2994   \__enumext_starred_joined_item_vii:n {#1}
2995   \peek_meaning_remove:NTF *
2996     { \__enumext_starred_item_vii:w }
2997     { \__enumext_standard_item_vii:w }
2998 }

```

(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].

```

2999 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3000 {
3001   \bool_set_false:N \l__enumext_item_starred_vii_bool
3002   \peek_meaning:NTF [
3003     {
3004       \bool_set_eq:NN
3005         \l__enumext_wrap_label_vii_bool
3006         \l__enumext_wrap_label_opt_vii_bool
3007       \__enumext_start_item_vii:w
3008     }
3009     {
3010       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3011       \legacy_if_set_true:n { @noitemarg }
3012       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3013     }
3014   }

```

(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⟩]`.

```

3015 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3016 {
3017   \bool_set_true:N \l__enumext_item_starred_vii_bool
3018   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3019   \peek_meaning:NTF [
3020     { \__enumext_starred_item_vii_aux_i:w }
3021     { \__enumext_starred_item_vii_aux_ii:w }
3022   }
3023   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3024   {
3025     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3026     \__enumext_starred_item_vii_aux_ii:w
3027   }
3028   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3029   {
3030     \peek_meaning:NTF [
3031       { \__enumext_starred_item_vii_aux_iii:w }
3032       {
3033         \dim_set_eq:NN
3034           \l__enumext_item_symbol_sep_vii_dim
3035           \l__enumext_labelsep_vii_dim
3036         \legacy_if_set_true:n { @noitemarg }
3037         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3038       }
3039     }
3040   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3041   {
3042     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3043     \legacy_if_set_true:n { @noitemarg }
3044     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3045   }

```

(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.

```

3046 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3047 {
3048   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3049   \legacy_if:nT { @noitemarg }
3050   {
3051     \legacy_if_set_false:n { @noitemarg }
3052     \legacy_if:nT { @nmbrlist }
3053     {
3054       \bool_if:NT \l__enumext_hyperref_bool
3055       {
3056         \legacy_if_set_true:n { @hyper@item }
3057       }
3058       \refstepcounter{enumXvii}
3059       \bool_if:NT \l__enumext_check_ans_bool
3060       {
3061         \int_gincr:N \g__enumext_count_item_number_int
3062       }
3063     }
3064   }

```

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

```

3065   \group_begin:
3066   \lrbox{ \l__enumext_item_text_vii_box }
3067   \bool_if:NF \l__enumext_footnotes_key_bool
3068   {
3069     \__enumext_renew_footnote:
3070   }
3071   \bool_if:NT \l__enumext_item_starred_vii_bool
3072   {
3073     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3074     {
3075       \tl_gset_eq:NN
3076       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3077     }
3078     \mode_leave_vertical:
3079     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3080     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3081     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3082     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3083   }
3084   \group_begin:
3085   \tl_use:N \l__enumext_label_font_style_vii_tl
3086   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3087   {
3088     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3089     { \__enumext_wrapper_label_vii:n {#1} }
3090   }
3091   {
3092     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3093   }
3094   \group_end:
3095   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3096   \tl_use:N \l__enumext_after_list_args_vii_tl
3097   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3098   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3099   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3100   \tl_use:N \l__enumext_fake_item_indent_vii_tl
3101 }

```

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

`__enumext_stop_item_vii:` The function `__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its $\langle contents \rangle$. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box

and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

3102 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3103 {
3104     \__enumext_endminipage:
3105     \endlrbox
3106     \group_end:
3107     \box_set_wd:Nn \l__enumext_item_text_vii_box
3108     {
3109         \l__enumext_joined_width_vii_dim
3110         + \l__enumext_labelwidth_vii_dim
3111         + \l__enumext_labelsep_vii_dim
3112     }
3113     \int_set:Nn \hbadness { 10000 }
3114     \box_use:N \l__enumext_item_text_vii_box
3115     \bool_if:NF \l__enumext_footnotes_key_bool
3116     {
3117         \__enumext_print_footnote:
3118     }
3119     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3120     {
3121         \par\noindent
3122         \int_zero:N \l__enumext_item_column_pos_vii_int
3123     }
3124     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3125 }

```

(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.

```

3126 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3127 {
3128     \int_compare:nNnT
3129     {
3130         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3131     }
3132     =
3133     { \c_zero_int }
3134     {
3135         \par
3136         \vspace{ -\l__enumext_itemsep_vii_skip }
3137         \int_gzero:N \g__enumext_item_count_all_vii_int
3138     }
3139 }

```

(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`.

```

3140 \__enumext_after_env:nn {enumext*}
3141 {
3142     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3143     {
3144         \bool_if:NT \g__enumext_check_ans_show_h_bool
3145         {
3146             \__enumext_check_ans_show:
3147         }
3148         \bool_gset_false:N \g__enumext_starred_bool
3149         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3150         \tl_gclear:N \g__enumext_store_name_tl
3151     }
3152 }

```

10.36 The keyans* environment

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.

```

3153 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3154 {
3155     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }

```

```

3156     {
3157         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3158         {
3159             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3160             / \l__enumext_columns_viii_int
3161         }
3162     }
3163     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3164     \dim_set:Nn \l__enumext_item_width_viii_dim
3165     {
3166         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3167         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3168         - \l__enumext_labelsep_viii_dim
3169     }
3170     \dim_zero_new:N \itemwidth
3171 }

```

(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`.

```

3172 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3173 {
3174     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3175     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3176     {
3177         \msg_warning:nnee { enumext } { item-joined }
3178         { \int_use:N \l__enumext_joined_item_viii_int }
3179         { \int_use:N \l__enumext_columns_viii_int }
3180         \int_set:Nn \l__enumext_joined_item_viii_int
3181         {
3182             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3183         }
3184     }
3185     \int_compare:nNnT
3186     { \l__enumext_joined_item_viii_int }
3187     >
3188     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3189     {
3190         \msg_warning:nnee { enumext } { item-joined-columns }
3191         { \int_use:N \l__enumext_joined_item_viii_int }
3192         {
3193             \int_eval:n
3194             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3195         }
3196         \int_set:Nn \l__enumext_joined_item_viii_int
3197         {
3198             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3199         }
3200     }
3201     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3202     {
3203         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3204         \int_decr:N \l__enumext_joined_item_aux_viii_int
3205         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3206         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3207         \dim_set:Nn \l__enumext_joined_width_viii_dim
3208         {
3209             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3210             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3211                 + \l__enumext_columns_sep_viii_dim
3212                 ) * \l__enumext_joined_item_aux_viii_int
3213         }
3214         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3215     }
3216     {
3217         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3218         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3219     }
3220 }

```

(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.

```

3221 \cs_new_protected:Nn \__enumext_start_mini_viii:
3222 {
3223   \dim_compare:nNt { \__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3224   {
3225     \dim_set:Nn \__enumext_minipage_left_viii_dim
3226     {
3227       \linewidth
3228       - \__enumext_minipage_right_viii_dim
3229       - \__enumext_minipage_hsep_viii_dim
3230     }
3231     \bool_set_true:N \__enumext_minipage_active_viii_bool
3232     \dim_gset_eq:NN
3233       \__enumext_minipage_right_viii_dim
3234       \__enumext_minipage_right_viii_dim
3235     \__enumext_mini_addvspace_viii:
3236     \nointerlineskip\noindent
3237     \begin{__enumext_mini_env*}{ \__enumext_minipage_left_viii_dim }
3238   }
3239 }
3240 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3241 {
3242   \bool_if:NT \__enumext_minipage_active_viii_bool
3243   {
3244     \end{__enumext_mini_env*}
3245     \hfill
3246     \bool_gset_true:N \__enumext_minipage_active_viii_bool
3247   }
3248 }
3249 \__enumext_after_env:nn {keyans*}
3250 {
3251   \bool_if:NT \__enumext_minipage_active_viii_bool
3252   {
3253     \begin{__enumext_mini_env*}{ \__enumext_minipage_right_viii_dim }
3254     \par\addvspace { \__enumext_minipage_right_skip }
3255     \bool_if:NF \__enumext_minipage_center_viii_bool
3256     {
3257       \centering
3258     }
3259     \tl_use:N \__enumext_miniright_code_viii_tl % the code
3260     \end{__enumext_mini_env*}
3261     \par\addvspace{ \__enumext_minipage_after_skip }
3262   }
3263   \bool_gset_false:N \__enumext_minipage_active_viii_bool
3264   \bool_gset_true:N \__enumext_minipage_center_viii_bool
3265   \tl_gclear:N \__enumext_miniright_code_viii_tl
3266   \dim_gzero:N \__enumext_minipage_right_viii_dim
3267 }

```

(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.

```

3268 \NewDocumentEnvironment{keyans*}{ o }
3269 {
3270   \__enumext_safe_exec_viii:
3271   \__enumext_parse_keys_viii:n {#1}
3272   \__enumext_before_list_viii:
3273   \__enumext_start_list:nn { }
3274   {
3275     \__enumext_list_arg_two_viii:
3276     \__enumext_before_keys_exec_viii:
3277   }
3278   \__enumext_starred_columns_set_viii:
3279   \item[] \scan_stop:
3280   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3281   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3282 }

```

```

3283 {
3284   \__enumext_stop_item_tmp_viii:
3285   \__enumext_remove_extra_parsep_viii:
3286   \__enumext_keyans_check_ans:nn { item }{ keyans* }
3287   \__enumext_stop_list:
3288   \__enumext_after_list_viii:
3289 }

```

(End of definition for `keyans*`. This function is documented on page 10.)

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

```

3290 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3291 {
3292   \int_incr:N \l__enumext_keyans_level_h_int
3293   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3294   {
3295     \msg_error:nn { enumext } { nested }
3296   }
3297   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3298   \bool_set_false:N \l__enumext_store_active_bool
3299   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3300   {
3301     \msg_error:nn { enumext } { keyans-wrong-level }
3302   }
3303 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

3304 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3305 {
3306   \tl_if_novalue:nF {#1}
3307   {
3308     \keys_set:nn { enumext / keyans* } {#1}
3309   }
3310 }

```

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

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

```

3311 \cs_new_protected:Nn \__enumext_before_list_viii:
3312 {
3313   \__enumext_vspace_above_viii:
3314   \__enumext_before_args_exec_viii:
3315   \__enumext_start_mini_viii:
3316 }

```

(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.

```

3317 \cs_new_protected:Nn \__enumext_after_list_viii:
3318 {
3319   \__enumext_stop_mini_viii:
3320   \__enumext_after_stop_list_viii:
3321   \__enumext_vspace_below_viii:
3322 }

```

(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`.

```
3323 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3324 {
3325   \__enumext_stop_item_tmp_viii:
3326   \int_incr:N \l__enumext_item_column_pos_viii_int
3327   \int_gincr:N \g__enumext_item_count_all_viii_int
3328   \__enumext_item_peek_args_viii:
3329 }
```

(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).

```
3330 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3331 {
3332   \peek_meaning:NTF (
3333     { \__enumext_joined_item_viii:w }
3334     { \__enumext_joined_item_viii:w (1) }
3335 }
```

(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`.

```
3336 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3337 {
3338   \__enumext_starred_joined_item_viii:n {#1}
3339   \peek_meaning_remove:NTF *
3340     { \__enumext_starred_item_viii:w }
3341     { \__enumext_standard_item_viii:w }
3342 }
```

(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]`.

```
3343 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3344 {
3345   \bool_set_false:N \l__enumext_item_starred_viii_bool
3346   \peek_meaning:NTF [
3347     {
3348       \bool_set_eq:NN
3349       \l__enumext_wrap_label_viii_bool
3350       \l__enumext_wrap_label_opt_viii_bool
3351       \__enumext_start_item_viii:w
3352     }
3353     {
3354       \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

3355         \legacy_if_set_true:n { @noitemarg }
3356         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3357     }
3358 }

```

(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].

```

3359 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3360 {
3361     \bool_set_true:N \__enumext_item_starred_viii_bool
3362     \bool_set_true:N \__enumext_wrap_label_viii_bool
3363     \peek_meaning:NTF [
3364         { \__enumext_starred_item_viii_aux_i:w }
3365         { \__enumext_starred_item_viii_aux_ii:w }
3366     }

```

The optional argument will be captured in the variables __enumext_keyans_tmpa_tl and __enumext_keyans_tmppb_tl which we will use later for the implementation of the show-ans and show-pos keys together with the stored in \langle sequence \rangle and \langle prop list \rangle.

```

3367 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3368 {
3369     \tl_clear:N \__enumext_store_keyans_label_tl
3370     \tl_if_no_value:nF { #1 }
3371     {
3372         \tl_set:Nx \__enumext_keyans_tmpa_tl { \c_space_tl [#1] }
3373         \tl_set:Nx \__enumext_keyans_tmppb_tl { , \c_space_tl #1 }
3374     }
3375     \__enumext_starred_item_viii_aux_ii:w
3376 }
3377 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3378 {
3379     \legacy_if_set_true:n { @noitemarg }
3380     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3381 }

```

(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.

```

3382 \cs_new_protected:Nn \__enumext_starred_item_exec:
3383 {
3384     \tl_put_left:Nx \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3385     \tl_if_blank:VF \__enumext_keyans_tmppb_tl
3386     {
3387         \tl_put_right:Nx \__enumext_store_keyans_label_tl { \__enumext_keyans_tmppb_tl }
3388     }
3389     \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3390     \__enumext_keyans_store_ref:
3391     \tl_put_left:Nx \__enumext_store_keyans_label_tl { \item }
3392     \__enumext_keyans_addto_seq_link:
3393     \bool_if:NT \__enumext_show_answer_bool
3394     {
3395         \tl_if_blank:VF \__enumext_keyans_tmpa_tl
3396         {
3397             \tl_put_right:Nx \__enumext_label_viii_tl { \__enumext_keyans_tmpa_tl }
3398             \__enumext_label_width_by_box:Nn \__enumext_keyans_tmpa_dim { \__enumext_keyans_tmppb_dim }
3399             \dim_add:Nn \__enumext_fake_item_indent_viii_dim { \__enumext_keyans_tmpa_dim }
3400         }
3401         \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3402     }
3403     \bool_if:NT \__enumext_show_position_bool
3404     {
3405         \tl_set:Nx \__enumext_mark_answer_sym_tl
3406         {
3407             \group_begin:
3408             \exp_not:N \normalfont

```

```

3409         \exp_not:N \footnotesize [ \int_eval:n
3410         {
3411             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3412         }
3413     ]
3414     \group_end:
3415 }
3416 \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3417 {
3418     \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3419     \__enumext_label_width_by_box:Nn \l__enumext_keyans_tmpa_dim { \l__enumext_keyans_tmpa_dim }
3420     \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l__enumext_keyans_tmpa_dim }
3421 }
3422 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3423 }
3424 }

```

(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.

`__enumext_start_item_viii:w`

```

3425 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3426 {
3427     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3428     \legacy_if:nT { @noitemarg }
3429     {
3430         \legacy_if_set_false:n { @noitemarg }
3431         \legacy_if:nT { @nمبرlist }
3432         {
3433             \bool_if:NT \l__enumext_hyperref_bool
3434             {
3435                 \legacy_if_set_true:n { @hyper@item }
3436             }
3437             \refstepcounter{enumXviii}
3438         }
3439     }

```

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

```

3440     \group_begin:
3441     \lrbox{ \l__enumext_item_text_viii_box }
3442     \bool_if:NF \l__enumext_footnotes_key_bool
3443     {
3444         \__enumext_renew_footnote:
3445     }
3446     \bool_if:NT \l__enumext_item_starred_viii_bool
3447     {
3448         \__enumext_starred_item_exec:
3449     }
3450     \group_begin:
3451     \tl_use:N \l__enumext_label_font_style_viii_tl
3452     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3453     {
3454         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3455         { \__enumext_wrapper_label_viii:n {#1} }
3456     }
3457     {
3458         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3459     }
3460     \group_end:
3461     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3462     \tl_use:N \l__enumext_after_list_args_viii_tl
3463     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3464     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3465     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3466     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3467 }

```

(End of definition for `__enumext_start_item_viii:w`)

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

```

3468 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3469 {
3470     \__enumext_endminipage:
3471     \endlrbox
3472     \group_end:
3473     \box_set_wd:Nn \l__enumext_item_text_viii_box
3474     {
3475         \l__enumext_joined_width_viii_dim
3476         + \l__enumext_labelwidth_viii_dim
3477         + \l__enumext_labelsep_viii_dim
3478     }
3479     \int_set:Nn \hbadness { 10000 }
3480     \box_use:N \l__enumext_item_text_viii_box
3481     \bool_if:NF \l__enumext_footnotes_key_bool
3482     {
3483         \__enumext_print_footnote:
3484     }
3485     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3486     {
3487         \par\noindent
3488         \int_zero:N \l__enumext_item_column_pos_viii_int
3489     }
3490     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3491 }

```

(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.

```

3492 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3493 {
3494     \int_compare:nNnT
3495     {
3496         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3497     }
3498     =
3499     { \c_zero_int }
3500     {
3501         \par
3502         \vspace{ -\l__enumext_itemsep_viii_skip }
3503         \int_gzero:N \g__enumext_item_count_all_viii_int
3504     }
3505 }

```

(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 `{⟨store name : position⟩}`. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from *⟨prop list⟩* defined by `save-ans` key.

```

3506 \NewDocumentCommand \getkeyans { m }
3507 {
3508     \exp_args:Ne \__enumext_getkeyans_aux:n
3509     { \tl_to_str:e { \text_expand:n {#1} } }
3510 }

```

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

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

```

3511 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3512 {
3513     \str_if_in:nnTF {#1} { : }
3514     {
3515         \use:e
3516         {
3517             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3518             { {##1} {##2} }

```

```

3519     }
3520     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3521   }
3522   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3523 }

```

(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⟩*.

```

3524 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3525 {
3526   \prop_if_exist:cF { g__enumext_#1_prop }
3527   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3528   \group_begin:
3529     \prop_item:cn { g__enumext_#1_prop }{#2}
3530   \group_end:
3531 }

```

(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.

```

3532 \keys_define:nn { keyanskey / print }
3533 {
3534   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3535                   {
3536                     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3537                   },
3538   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3539   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3540                   {
3541                     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3542                   },
3543   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3544   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3545                   {
3546                     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3547                   },
3548   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3549   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3550                   {
3551                     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3552                   },
3553   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3554   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3555                   {
3556                     \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3557                   },
3558   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3559 }

```

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

```

3560 \NewDocumentCommand \printkeyans { s O{} m }
3561 {
3562   \group_begin:
3563     \tl_use:N \l__enumext_print_keyans_i_tl
3564     \tl_use:N \l__enumext_print_keyans_ii_tl
3565     \tl_use:N \l__enumext_print_keyans_iii_tl
3566     \tl_use:N \l__enumext_print_keyans_iv_tl
3567     \tl_use:N \l__enumext_print_keyans_vii_tl
3568     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3569   \group_end:
3570 }

```

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

`__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

```

3571 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3572 {
3573   \seq_if_exist:cTF { g__enumext_#3_seq }
3574   {
3575     \seq_if_empty:cF { g__enumext_#3_seq }
3576     {
3577       %%\seq_show:c { g__enumext_#3_seq }
3578       \bool_if:nTF {#1}
3579       {
3580         \begin{enumext*}[#2]
3581         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3582         \end{enumext*}
3583       }
3584       {
3585         \begin{enumext}[#2]
3586         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3587         \end{enumext}
3588       }
3589     }
3590   }
3591   {
3592     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3593   }
3594 }

```

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

10.39 The command `\setenumext`

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

```

3595 \keys_define:nn { enumext / meta-families }
3596 {
3597   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3598   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3599   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3600   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3601   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3602   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3603   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3604   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3605   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3606   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3607   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3608   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3609   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3610 }

```

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

```

3611 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3612 {
3613   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3614   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3615 }

```

`\setenumext`

Now we define the user command `\setenumext`.

```

3616 \NewDocumentCommand \setenumext { o +m }
3617 {
3618   \tl_if_novalue:nTF {#1}
3619   {
3620     \seq_map_inline:Nn \c__enumext_all_families_seq
3621     {
3622       \seq_clear:N \l__enumext_setkey_tmpa_seq
3623       \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3624       \int_set:Nn \l__enumext_setkey_tmpa_int

```

```

3626     {
3627         \seq_count:N \l__enumext_setkey_tmpb_seq
3628     }
3629     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3630     {
3631         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3632         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3633         \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3634         {
3635             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3636         }
3637     }
3638     {
3639         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3640     }
3641     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3642     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3643     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3644 }
3645 {
3646     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3647 }
3648 }

```

(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.

```

3649 \cs_new_protected:Npn \__enumext_set_parse:n #1
3650 {
3651     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3652     \int_step_inline:nnn { 0 } { 4 } %<- max level
3653     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3654     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3655     {
3656         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3657         { \tl_trim_spaces:n {#1} }
3658     }
3659     { \__enumext_set_error:nn {#1} { } }
3660 }
3661 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3662 { \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.

```

3663 \msg_new:nnn { enumext } { package-load }
3664 {
3665     The ~ '#1' ~ package ~ is ~ already ~ loaded.
3666 }
3667 \msg_new:nnn { enumext } { package-not-load }
3668 {
3669     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3670 }
3671 \msg_new:nnn { enumext } { package-load-foot }
3672 {
3673     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3674 }

```

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

```

3675 \msg_new:nnn { enumext } { counters }
3676 {
3677     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3678     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3679 }

```

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

```

3680 \msg_new:nnn { enumext } { invalid-key }
3681 {
3682     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.

```

```

3683 }
3684 \msg_new:nnn { enumext } { unknown-key-family }
3685 {
3686   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3687 }

```

Messages used in length calculation.

```

3688 \msg_new:nnn { enumext } { width-negative }
3689 {
3690   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3691   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
3692 }
3693 \msg_new:nnn { enumext } { width-zero }
3694 {
3695   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3696   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
3697 }

```

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

```

3698 \msg_new:nnn { enumext } { list-lengths }
3699 {
3700   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3701   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3702   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3703   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3704   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3705   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3706   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3707   \__enumext_show_length:nnn { skip } { topsep } {#1}
3708   \__enumext_show_length:nnn { skip } { parsep } {#1}
3709   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3710   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3711   ****~
3712 }

```

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

```

3713 \msg_new:nnn { enumext } { list-lengths-not-nested }
3714 {
3715   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3716   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3717   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3718   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3719   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3720   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3721   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3722   \__enumext_show_length:nnn { skip } { topsep } {#1}
3723   \__enumext_show_length:nnn { skip } { parsep } {#1}
3724   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3725   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3726   ****~
3727 }

```

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

```

3728 \msg_new:nnn { enumext } { items-same-answer }
3729 {
3730   ****~Checking~answers~on~'#1'~OK~****~\
3731   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \
3732   ****~
3733   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3734 }
3735 \msg_new:nnn { enumext } { item-different-answer }
3736 {
3737   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3738   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3739 }

```

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

```

3740 \msg_new:nnn { enumext } { missing-starred }
3741 {
3742   Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3743 }

```

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

```
3744 \msg_new:nnn { enumext } { list-too-deep }
3745 {
3746     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
3747     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3748 }
```

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

```
3749 \msg_new:nnn { enumext } { anskey-wrong-place }
3750 {
3751     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3752     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3753 }
3754 \msg_new:nnn { enumext } { anspic-wrong-place }
3755 {
3756     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3757     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3758 }
3759 \msg_new:nnn { enumext } { command-wrong-place }
3760 {
3761     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3762     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3763 }
```

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

```
3764 \msg_new:nnn { enumext } { keyans-nested }
3765 {
3766     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3767 }
3768 \msg_new:nnn { enumext } { keyans-wrong-level }
3769 {
3770     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3771     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3772 }
3773 \msg_new:nnn { enumext } { wrong-place }
3774 {
3775     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3776     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3777 }
3778 \msg_new:nnn { enumext } { keyanspic-nested }
3779 {
3780     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
3781 }
3782 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3783 {
3784     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
3785     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3786 }
```

Messages used by `\getkeyans` command.

```
3787 \msg_new:nnn { enumext } { undefined-storage-anskey }
3788 {
3789     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3790 }
```

Messages used by `\miniright` command.

```
3791 \msg_new:nnn { enumext } { missing-miniright }
3792 {
3793     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
3794     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3795 }
3796 \msg_new:nnn { enumext } { wrong-miniright-place }
3797 {
3798     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3799     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3800 }
3801 \msg_new:nnn { enumext } { wrong-miniright-use }
3802 {
3803     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3804     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3805 }
```

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

```
3806 \msg_new:nnn { enumext } { nested }
3807 {
3808   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3809 }
3810 \msg_new:nnn { enumext } { item-joined }
3811 {
3812   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
3813 }
3814 \msg_new:nnn { enumext } { item-joined-columns }
3815 {
3816   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
3817 }
```

10.41 Finish package

Finish package implementation.

```
3818 \file_input_stop:
3819 </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

*

.....

417

\+

.....

194

\-

.....

194

\\

202, 2648, 3677, 3690, 3695, 3700, 3715, 3730, 3731, 3746, 3751, 3756, 3761, 3770, 3775, 3784, 3793, 3798, 3803

A

above

.....

1228

above*

.....

1228

\addvspace

875, 903, 1026, 1105, 1168, 1174, 1202, 1219, 2464, 2486, 2604, 2619, 2843, 2850, 3254, 3261

after

.....

713

align

.....

371

\Alph

.....

30, 35

\Alph

.....

323, 500, 518, 531, 3553

\alph

.....

30, 35

\alph

.....

324, 498, 3543

\anskey

.....

11, 57, 1636

\anspic

.....

13, 78, 79, 2626

\arabic

.....

30, 32

\arabic

.....

322, 497, 517, 3538, 3558

B

\b

.....

2342, 2355, 2905, 2918

\baselineskip

.....

42

\baselineskip

.....

1596, 1604

before

.....

713

before*

.....

713

below

.....

1228

below*

.....

1228

bool commands:

\bool_gset_false:N

2508, 2509, 2852, 2956, 3148, 3149, 3263

\bool_gset_true:N

220, 229, 817, 2471, 2835, 2853, 2953, 3246, 3264

\bool_if:NTF

263, 275, 292, 1250, 1264, 1277, 1288, 1299, 1310, 1321, 1332, 1375, 1386, 1441, 1451, 1558, 1582, 1589, 1617, 1648, 1661, 1663, 1674, 1694, 1819, 1830, 1834, 1868, 1883, 1952, 1967, 1980, 2058, 2088, 2161, 2177, 2245, 2255, 2291, 2296, 2333, 2340, 2353, 2398, 2448, 2462, 2477, 2504, 2533, 2589, 2602, 2610, 2628, 2831, 2840, 2844, 2895, 2903, 2916, 2961, 2971, 3054, 3059, 3067, 3071, 3086, 3115, 3144, 3242, 3251, 3255, 3393, 3403, 3433, 3442, 3446, 3452, 3481

\bool_if:nTF

1203, 1220, 1702, 2099, 2133, 2197, 2649, 3578

\bool_if_p:N

1420, 1685, 1728, 1729, 1753, 1762, 1763, 1775, 1791, 1938, 1939, 2234, 2281, 2371, 2384, 2386, 2468, 2655, 2656, 2950, 2951

\bool_lazy_all:nTF

1418, 1751, 1760, 1773, 1789, 2369, 2382

\bool_lazy_and:nnTF

1684, 1727, 1937, 2232, 2280, 2467, 2949

\bool_lazy_or:nnTF

.....

2654

\bool_new:N

25, 26, 27, 28, 29, 35, 37, 46, 67, 72, 73, 78, 79, 82, 102, 104, 106, 109, 110, 119, 120, 121, 122, 131, 132, 157, 168, 170

\bool_not_p:n

1686, 1778, 1793, 2372, 2373, 2385, 2469

\bool_set_eq:NN

.....

2066, 2114, 3004, 3348

\bool_set_false:N

272, 1402, 1403, 1485, 1488, 1508, 1511, 2491, 2540, 2622, 2686, 2703, 2957, 3001, 3298, 3345

\bool_set_true:N

254, 258, 364, 641, 1234, 1239, 1366, 1367, 1382, 1393, 1484, 1487, 1507, 1510, 1520, 1527, 2062, 2092, 2110, 2122, 2327, 2378, 2391, 2417, 2538, 2564, 2820, 2888, 3010, 3017, 3018, 3231, 3354, 3361, 3362

box commands:

\box_dp:N

922, 926, 930, 941, 945, 956, 965, 971, 981, 994, 1000, 1006, 1037, 1038, 1039, 1042, 1052, 1056, 1065, 1072, 1077, 1085, 1114, 1115, 1118, 1125, 1138, 1146, 1152, 1160, 2715

\box_new:N

.....

43, 163

\box_set_wd:Nn

.....

3107, 3473

\box_use:N

.....

3114, 3480

\box_wd:N

.....

330

C

\c

.....

417, 418, 541, 543, 555, 557

\cB

.....

418

\cE

.....

418

\centering

.....

1205, 1222, 2736, 2846, 3257

check-ans

.....

1395

Document class:

article

.....

36

clist commands:

\clist_const:Nn

.....

175

\clist_map_function:nN

.....

2723

\clist_map_inline:Nn

370, 583, 646, 712, 727, 808, 1244

\clist_map_inline:nn

34, 51, 57, 69, 81, 108, 140, 154, 174, 210, 395, 412, 651, 823, 1408, 1497, 1515, 1536, 1748, 1877, 2016, 2226, 2229, 2262, 2272, 2275, 2301

\columnbreak

.....

58

\columnbreak

.....

1688

columns

.....

792

columns*

.....

1516

columns-sep

.....

792

columns-sep*

.....

1516

\columnsep

.....

74, 77

\columnsep

.....

2442, 2586

\columnseprule

.....

74, 77

\columnseprule

.....

2446, 2588

Commands provide by enumext:

\anskey

.....

24, 25, 51, 53, 55–62, 64, 73, 86, 96, 97, 101

\anspic*

.....

24, 61–64, 79–81, 96, 97

\anspic

.....

55, 56, 78–81, 101

\getkeyans

.....

55, 96, 101

\item*

.....

24, 55, 56, 61–64, 66, 67, 87, 94, 96, 97

\itemwidth

.....

82, 89, 90

\item

.....

66, 67, 82, 86, 87, 90, 93

\miniright

.....

24, 40, 48, 49, 74, 75, 77, 78, 101

\printkeyans

.....

25, 56, 97

\setenumext

.....

24, 98, 99

Counters defined by enumext:

enumXiii

.....

23, 30

enumXii

.....

23, 30

enumXiv

.....

23, 30

©2024 by Pablo González L

103 / 113

enumXi 23, 30
 enumXviii 23, 30
 enumXvii 23, 30, 87
 enumXvi 23, 30
 enumXv 23, 30
 cs commands:

\cs_generate_variant:Nn 332, 348, 547, 563, 1541,
 1550, 1555, 1635, 2216, 2725
 \cs_if_exist:NTF 302
 \cs_new:Nn 188
 \cs_new:Npn 198
 \cs_new_eq:NN 238, 239, 240, 244, 245, 277, 278, 281,
 282
 \cs_new_protected:Nn . 212, 249, 413, 433, 465, 728,
 732, 736, 740, 744, 748, 752, 756, 760, 764, 768, 772,
 776, 780, 784, 788, 824, 836, 860, 877, 888, 912, 987,
 1011, 1028, 1090, 1107, 1129, 1164, 1170, 1245, 1259,
 1273, 1284, 1295, 1306, 1317, 1328, 1373, 1384, 1413,
 1449, 1456, 1556, 1580, 1587, 1615, 1622, 1739, 1866,
 1881, 1909, 1935, 2021, 2025, 2044, 2095, 2129, 2145,
 2155, 2171, 2322, 2367, 2396, 2403, 2426, 2456, 2475,
 2531, 2554, 2572, 2597, 2608, 2645, 2688, 2701, 2721,
 2726, 2742, 2810, 2829, 2881, 2930, 2937, 2959, 2969,
 2986, 3126, 3153, 3221, 3240, 3290, 3311, 3317, 3330,
 3382, 3492
 \cs_new_protected:Npn 180, 184, 285, 300, 317, 327,
 333, 421, 440, 534, 548, 1192, 1211, 1355, 1542, 1551,
 1671, 1816, 1828, 1850, 1919, 1957, 1965, 2054, 2072,
 2106, 2118, 2185, 2219, 2265, 2329, 2338, 2550, 2696,
 2761, 2890, 2901, 2992, 2999, 3015, 3023, 3028, 3040,
 3172, 3304, 3336, 3343, 3359, 3367, 3377, 3511, 3524,
 3571, 3649, 3661
 \cs_new_protected_nopar:Nn . . . 2979, 3102, 3323,
 3468
 \cs_new_protected_nopar:Npn 3046, 3425
 \cs_set:Nn 1821
 \cs_set:Npn 1749, 1787, 3517
 \cs_set_eq:NN . . 2871, 2872, 3048, 3280, 3281, 3427
 \cs_set_protected:Nn 204, 652, 668, 680, 692
 \cs_set_protected:Npn . 30, 44, 52, 64, 70, 98, 136,
 148, 155, 206, 349, 371, 400, 481, 501, 564, 584, 628,
 647, 704, 713, 792, 809, 1228, 1395, 1469, 1498, 1516,
 1741, 1870, 2005, 2217, 2263
 \cs_to_str:N 319, 342

D

\d 194
 \DeclareDocumentEnvironment 905
 dim commands:

\dim_abs:n 2190, 2195
 \dim_add:Nn 2706, 3399, 3420
 \dim_compare:nNnTF . 654, 670, 682, 694, 1194, 1213,
 2187, 2192, 2198, 2204, 2206, 2208, 2408, 2431, 2558,
 2576, 2698, 2744, 2812, 3155, 3223
 \dim_compare:nTF 1712
 \dim_gset_eq:NN 2821, 3232
 \dim_gzero:N 2855, 3266
 \dim_new:N 40, 47, 48, 49, 66, 92, 105, 115, 164, 165, 171
 \dim_set:Nn . . 330, 642, 1528, 2086, 2190, 2195, 2197,
 2200, 2201, 2205, 2207, 2210, 2211, 2213, 2411, 2434,
 2560, 2578, 2728, 2746, 2753, 2796, 2814, 3042, 3157,
 3164, 3207, 3225
 \dim_set_eq:NN 488, 508, 524, 528, 2081, 2228, 2274,
 2357, 2442, 2586, 2803, 2806, 2807, 2920, 3033, 3214,
 3217, 3218

\dim_use:N 655, 663, 1195, 1201, 1625, 1628, 1633, 2150,
 2152, 2409, 2414, 2415, 2422, 2432, 2436, 2437, 2439
 \dim_zero:N 2446, 2588, 2707, 2708, 2709
 \dim_zero_new:N 2759, 3170
 \c_zero_dim 657, 671, 683, 695, 1195, 1213, 1714, 2187,
 2192, 2198, 2205, 2409, 2432, 2558, 2576, 2744, 2812,
 3155, 3223

E

\end . . 1198, 1216, 1584, 1619, 2461, 2485, 2601, 2618, 2833,
 2849, 3244, 3260, 3582, 3587
 \endlist 28
 \endlist 239
 \endlrbox 3105, 3471
 \endminipage 28
 \endminipage 245
 enumext 5, 2302

enumext internal commands:

\g__enumext_ __enumext_store_name_tl
 _prop 64
 __enumext_add_pre_parsep: . . . 41, 834, 836, 836
 __enumext_after_args_exec: . 39, 728, 740, 2315
 __enumext_after_args_exec_v: . 39, 40, 744, 756,
 2524
 __enumext_after_args_exec_vii: . . . 760, 784
 __enumext_after_args_exec_viii: 788
 __enumext_after_env:n 75
 __enumext_after_env:nn . . 76, 89, 184, 184, 2500,
 2838, 3140, 3249
 __enumext_after_hyperref: . . . 28, 247, 249, 249
 __enumext_after_list: 75, 85, 92, 2320, 2475, 2475
 \l__enumext_after_list_args_v_tl 758
 \l__enumext_after_list_args_vii_tl 786, 3096
 \l__enumext_after_list_args_viii_tl 790, 3462
 __enumext_after_list_v: . . 78, 2529, 2608, 2608
 __enumext_after_list_vii: . . . 2879, 2937, 2937
 __enumext_after_list_viii: . . 3288, 3317, 3317
 __enumext_after_star_env:nn 83
 __enumext_after_stop_list: . . . 38, 40, 728, 736,
 2489
 __enumext_after_stop_list_v: 39, 744, 752, 2623
 \l__enumext_after_stop_list_v_tl 754
 __enumext_after_stop_list_vii: 760, 776, 2940
 \l__enumext_after_stop_list_vii_tl . . . 778
 __enumext_after_stop_list_viii: . 780, 3320
 \l__enumext_after_stop_list_viii_tl . . . 782
 \l__enumext_align_label_vii_str . . 3088, 3092
 \l__enumext_align_label_viii_str . 3454, 3458
 \l__enumext_align_label_X_str 155
 \c__enumext_all_envs_clist . . 175, 370, 583, 646,
 712, 727, 808, 1244
 \c__enumext_all_families_seq . . 98, 3611, 3620,
 3642
 __enumext_anskey_wrapper:n 1473, 1826
 __enumext_at_begin_document:n . . 28, 180, 180,
 236, 242
 __enumext_before_args_exec: 38, 728, 728, 2406
 __enumext_before_args_exec_v: . . 39, 744, 744,
 2557
 __enumext_before_args_exec_vii: . . 760, 760,
 2934
 __enumext_before_args_exec_viii: 764, 3314
 __enumext_before_keys_exec: 38, 728, 732, 2313
 __enumext_before_keys_exec_v: . . 39, 744, 748,
 2522

[__enumext_before_keys_exec_vii](#) [760](#)
[__enumext_before_keys_exec_vii:](#) [39](#), [768](#), [2867](#)
[__enumext_before_keys_exec_viii:](#) . . [39](#), [772](#),
[3276](#)
[__enumext_before_list:](#) . . . [73](#), [2307](#), [2403](#), [2403](#)
[__enumext_before_list_v:](#) . [77](#), [2517](#), [2554](#), [2554](#)
[__enumext_before_list_vii:](#) [85](#), [2862](#), [2930](#), [2930](#)
[__enumext_before_list_viii:](#) . . [92](#), [3272](#), [3311](#),
[3311](#)
[\l__enumext_before_no_starred_key_v_tl](#) [750](#)
[\l__enumext_before_no_starred_key_vii_-](#)
[tl](#) [770](#)
[\l__enumext_before_no_starred_key_viii_-](#)
[tl](#) [774](#)
[\l__enumext_before_starred_key_v_tl](#) . . . [746](#)
[\l__enumext_before_starred_key_vii_tl](#) . [762](#)
[\l__enumext_before_starred_key_viii_tl](#) [766](#)
[__enumext_calc_hspace:NNNNNN](#) [69](#), [2185](#), [2185](#),
[2216](#), [2221](#), [2267](#)
[\l__enumext_check_ans_bool](#) . . . [51](#), [66](#), [119](#), [1399](#),
[1403](#), [1451](#), [1663](#), [1952](#), [2058](#), [2088](#), [2468](#), [2951](#), [3059](#)
[__enumext_check_ans_exec:](#) . . [54](#), [73](#), [1449](#), [1449](#),
[2407](#), [2933](#)
[__enumext_check_ans_int:n](#) [51](#)
[\g__enumext_check_ans_item_tl](#) . . [63](#), [119](#), [1951](#),
[1959](#), [1963](#)
[__enumext_check_ans_set:](#) . [53](#), [1413](#), [1413](#), [1453](#)
[__enumext_check_ans_show:](#) [54](#), [1456](#), [1456](#), [2506](#),
[3146](#)
[\g__enumext_check_ans_show_bool](#) [75](#), [119](#), [2471](#),
[2504](#), [2509](#)
[\g__enumext_check_ans_show_h_bool](#) [119](#), [2953](#),
[3144](#), [3149](#)
[\l__enumext_columns_sep_v_dim](#) [2576](#), [2578](#), [2586](#)
[\l__enumext_columns_sep_vii_dim](#) . . [2744](#), [2746](#),
[2755](#), [2800](#), [2922](#), [3124](#)
[\l__enumext_columns_sep_viii_dim](#) . [3155](#), [3157](#),
[3166](#), [3211](#), [3490](#)
[\l__enumext_columns_v_int](#) [1033](#), [2574](#), [2582](#), [2594](#),
[2599](#)
[\l__enumext_columns_vii_int](#) . . [2749](#), [2752](#), [2756](#),
[2764](#), [2768](#), [2771](#), [2777](#), [2783](#), [2787](#), [2909](#), [3119](#), [3130](#)
[\l__enumext_columns_viii_int](#) . [3160](#), [3163](#), [3167](#),
[3175](#), [3179](#), [3182](#), [3188](#), [3194](#), [3198](#), [3485](#), [3496](#)
[\g__enumext_count_item_anskey_int](#) . . [63](#), [119](#),
[1459](#), [1467](#), [1665](#), [1954](#)
[\g__enumext_count_item_number_int](#) [119](#), [1424](#),
[1429](#), [1432](#), [1435](#), [1443](#), [1459](#), [1466](#), [2060](#), [2090](#), [3061](#)
[\g__enumext_count_item_with_ans_int](#) [58](#)
[\l__enumext_counter_i_tl](#) [30](#), [309](#)
[\l__enumext_counter_ii_tl](#) [30](#), [310](#)
[\l__enumext_counter_iii_tl](#) [30](#), [311](#)
[\l__enumext_counter_iv_tl](#) [30](#), [312](#)
[\l__enumext_counter_style_for_ref_vii_-](#)
[tl](#) [448](#), [458](#), [469](#), [471](#)
[\l__enumext_counter_style_for_ref_viii_-](#)
[tl](#) [475](#), [477](#)
[\l__enumext_counter_style_for_ref_X_tl](#) [144](#)
[\c__enumext_counter_style_tl](#) [32](#), [144](#), [415](#)
[\g__enumext_counter_styles_tl](#) . [23](#), [30](#), [40](#), [320](#),
[338](#)
[\l__enumext_counter_v_tl](#) [30](#), [313](#)
[\l__enumext_counter_vi_tl](#) [30](#), [314](#)
[\l__enumext_counter_vii_tl](#) [30](#), [315](#), [445](#)
[\l__enumext_counter_viii_tl](#) [30](#), [316](#), [455](#)
[__enumext_current_env:](#) [27](#), [212](#), [212](#), [2304](#), [2859](#)
[\l__enumext_current_widest_dim](#) [23](#), [40](#), [344](#), [489](#),
[509](#), [525](#), [529](#)
[__enumext_default_item:n](#) . . . [2054](#), [2054](#), [2103](#)
[__enumext_define_counters:Nn](#) [23](#), [300](#), [300](#), [309](#),
[310](#), [311](#), [312](#), [313](#), [314](#), [315](#), [316](#)
[__enumext_endminipage:](#) . [28](#), [242](#), [245](#), [911](#), [2738](#),
[3104](#), [3470](#)
[__enumext_fake_item:](#) [652](#), [652](#), [2254](#)
[\l__enumext_fake_item_indent_v_dim](#) [671](#), [676](#)
[\l__enumext_fake_item_indent_v_tl](#) [673](#), [2111](#),
[2115](#), [2123](#)
[\l__enumext_fake_item_indent_vii_dim](#) [683](#), [688](#)
[\l__enumext_fake_item_indent_vii_tl](#) [685](#), [3100](#)
[\l__enumext_fake_item_indent_viii_dim](#) . [695](#),
[700](#), [3399](#), [3420](#)
[\l__enumext_fake_item_indent_viii_tl](#) . . [697](#),
[3466](#)
[\l__enumext_fake_item_indent_X_tl](#) [70](#)
[__enumext_fake_item_vii:](#) [652](#), [680](#), [2290](#)
[__enumext_fake_item_viii:](#) [652](#), [692](#), [2295](#)
[\g__enumext_footnote_arg_seq](#) . [141](#), [2027](#), [2040](#),
[2050](#)
[\g__enumext_footnote_int](#) . [141](#), [2034](#), [2037](#), [2039](#),
[2041](#)
[\g__enumext_footnote_int_seq](#) . [141](#), [2028](#), [2041](#),
[2046](#), [2049](#)
[__enumext_footnotes_key_bool](#) [28](#)
[\l__enumext_footnotes_key_bool](#) [25](#), [29](#), [88](#), [131](#),
[258](#), [263](#), [272](#), [3067](#), [3115](#), [3442](#), [3481](#)
[__enumext_footnotetext:nn](#) . . . [2021](#), [2021](#), [2051](#)
[__enumext_getkeyans:nn](#) . . . [97](#), [3520](#), [3524](#), [3524](#)
[__enumext_getkeyans_aux:n](#) . [96](#), [3508](#), [3511](#), [3511](#)
[\l__enumext_hyperref_bool](#) . [25](#), [28](#), [29](#), [131](#), [254](#),
[275](#), [292](#), [1729](#), [1939](#), [3054](#), [3433](#)
[__enumext_hypertarget:nn](#) [29](#), [249](#), [277](#), [281](#), [297](#)
[__enumext_if_is_int:n](#) [192](#)
[__enumext_if_is_int:nTF](#) [192](#), [536](#), [550](#)
[\l__enumext_item_column_pos_vii_int](#) [86](#), [2771](#),
[2777](#), [2783](#), [2787](#), [2794](#), [2982](#), [3119](#), [3122](#)
[\l__enumext_item_column_pos_viii_int](#) . . . [93](#),
[3182](#), [3188](#), [3194](#), [3198](#), [3205](#), [3326](#), [3485](#), [3488](#)
[\l__enumext_item_column_pos_X_int](#) [155](#)
[\g__enumext_item_count_all_vii_int](#) [86](#), [2795](#),
[2983](#), [3130](#), [3137](#)
[\g__enumext_item_count_all_viii_int](#) [93](#), [3206](#),
[3327](#), [3496](#), [3503](#)
[\g__enumext_item_count_all_X_int](#) [155](#)
[__enumext_item_peek_args_vii:](#) [86](#), [2984](#), [2986](#),
[2986](#)
[__enumext_item_peek_args_viii:](#) [93](#), [3328](#), [3330](#),
[3330](#)
[__enumext_item_starred:](#) . . [68](#), [2145](#), [2145](#), [2163](#)
[\l__enumext_item_starred_vii_bool](#) [3001](#), [3017](#),
[3071](#)
[\l__enumext_item_starred_viii_bool](#) [3345](#), [3361](#),
[3446](#)
[\l__enumext_item_starred_X_bool](#) [155](#)
[__enumext_item_std:w](#) [28](#), [66](#), [67](#), [81](#), [236](#), [240](#), [2063](#),
[2069](#), [2093](#), [2111](#), [2115](#), [2123](#), [2719](#)
[\g__enumext_item_symbol_aux_vii_tl](#) [3025](#), [3073](#),
[3076](#), [3080](#), [3082](#)
[\g__enumext_item_symbol_aux_X_tl](#) [155](#)

```

\l__enumext_item_symbol_sep_vii_dim .. 3034,
    3042, 3079, 3081
\g__enumext_item_symbol_tl 23, 66, 35, 2078, 2151,
    2168
\l__enumext_item_symbol_vii_tl ..... 3076
\l__enumext_item_text_vii_box 3066, 3107, 3114
\l__enumext_item_text_viii_box 3441, 3473, 3480
\l__enumext_item_text_X_box ..... 155
\l__enumext_item_width_vii_dim ... 2753, 2798,
    2806, 2807
\l__enumext_item_width_viii_dim .. 3164, 3209,
    3217, 3218
\l__enumext_item_width_X_dim ..... 155
\l__enumext_itemindent_X_dim ..... 44
\l__enumext_itemsep_vii_skip ..... 3136
\l__enumext_itemsep_viii_skip ..... 3502
\l__enumext_joined_item_aux_vii_int .. 2792,
    2793, 2794, 2795, 2801
\l__enumext_joined_item_aux_viii_int . 3203,
    3204, 3205, 3206, 3212
\l__enumext_joined_item_aux_X_int .... 155
\__enumext_joined_item_vii:w .. 86, 2989, 2990,
    2992, 2992
\l__enumext_joined_item_vii_int .. 2763, 2764,
    2767, 2769, 2775, 2780, 2785, 2790, 2792, 2798
\__enumext_joined_item_viii:w . 93, 3333, 3334,
    3336, 3336
\l__enumext_joined_item_viii_int . 3174, 3175,
    3178, 3180, 3186, 3191, 3196, 3201, 3203, 3209
\l__enumext_joined_item_X_int ..... 155
\l__enumext_joined_width_vii_dim . 2796, 2803,
    2806, 3097, 3109
\l__enumext_joined_width_viii_dim 3207, 3214,
    3217, 3463, 3475
\l__enumext_joined_width_X_dim ..... 155
\__enumext_keyans_addto_prop:n 61, 1850, 1850,
    2125, 2651
\__enumext_keyans_addto_seq:n . 63, 1919, 1919,
    2127, 2653
\__enumext_keyans_addto_seq_link: 1919, 1933,
    1935, 3392
\__enumext_keyans_anspic_code:nnn . 79, 2642,
    2645, 2645
\__enumext_keyans_check_ans:nn .. 63, 64, 1957,
    1957, 2527, 2683, 3286
\__enumext_keyans_default_item:n .. 67, 2106,
    2106, 2141
\l__enumext_keyans_env_bool 20, 2372, 2385, 2538,
    2622
\__enumext_keyans_fake_item: .. 652, 668, 2244
\l__enumext_keyans_item_opt_tl ..... 82
\l__enumext_keyans_level_h_int 20, 1897, 3292,
    3293
\l__enumext_keyans_level_int .. 20, 1186, 1652,
    1892, 2537, 2541, 2636
\__enumext_keyans_make_label: 31, 69, 2171, 2171,
    2243
\__enumext_keyans_mini_addvspace: 46, 77, 1090,
    1090, 2566
\__enumext_keyans_mini_right_cmd:n 49, 1188,
    1211, 1211
\__enumext_keyans_mini_set_vskip: . 45, 1028,
    1028, 1092
\__enumext_keyans_multi_addvspace: . 78, 877,
    888, 2591
\__enumext_keyans_multi_set_vskip: . 42, 877,
    877, 890
\__enumext_keyans_multicols_start: 77, 2570,
    2572, 2572
\__enumext_keyans_multicols_stop: . 78, 1215,
    2597, 2597, 2621
\__enumext_keyans_parse_keys:n 2516, 2550, 2550
\l__enumext_keyans_pic_above_int . 114, 2729,
    2730, 2732
\l__enumext_keyans_pic_above_skip .. 81, 114,
    2674, 2713
\__enumext_keyans_pic_arg_two: 80, 2672, 2701,
    2701
\l__enumext_keyans_pic_below_int . 114, 2729,
    2730, 2733
\l__enumext_keyans_pic_body_seq .. 79–81, 114,
    2640, 2679, 2737
\__enumext_keyans_pic_do:n 81, 2679, 2681, 2721,
    2721, 2725
\l__enumext_keyans_pic_level_int .. 20, 1178,
    1656, 1853, 1887, 1922, 2690, 2691
\__enumext_keyans_pic_row:n 81, 2723, 2726, 2726
\__enumext_keyans_pic_safe_exec: .. 80, 2668,
    2688, 2688
\__enumext_keyans_pic_skip_abs:N .. 80, 2696,
    2696, 2712
\l__enumext_keyans_pic_width_dim . 114, 2728,
    2735
\__enumext_keyans_redefine_item: .. 68, 2129,
    2129, 2242
\__enumext_keyans_safe_exec: . 2515, 2531, 2531
\__enumext_keyans_show_left:n . 67, 1965, 1965,
    2121, 2659
\__enumext_keyans_starred_item:n .. 67, 2118,
    2118, 2137
\__enumext_keyans_store_ref: .. 62, 1866, 1866,
    2126, 2652, 3390
\__enumext_keyans_store_ref_aux_i: 62, 1866,
    1878, 1881
\__enumext_keyans_store_ref_aux_ii: 63, 1866,
    1907, 1909
\l__enumext_keyans_tmpa_dim 82, 3398, 3399, 3419,
    3420
\l__enumext_keyans_tmpa_tl 24, 94, 82, 2120, 2124,
    3372, 3395, 3397, 3398, 3416, 3418, 3419
\l__enumext_keyans_tmpb_tl .. 94, 82, 3373, 3385,
    3387
\l__enumext_label_copy_i_tl .. 1783, 1885, 1890,
    1895, 1900
\l__enumext_label_copy_v_tl ..... 1895
\l__enumext_label_copy_vi_tl ..... 1890
\l__enumext_label_copy_vii_tl 1758, 1769, 1800,
    1885
\l__enumext_label_copy_viii_tl ..... 1900
\l__enumext_label_copy_X_tl ..... 133
\l__enumext_label_fill_left_v_tl ..... 2175
\l__enumext_label_fill_left_X_tl ..... 70
\l__enumext_label_fill_right_v_tl .... 2182
\l__enumext_label_fill_right_X_tl ..... 70
\l__enumext_label_font_style_v_tl 2176, 2663
\l__enumext_label_font_style_vii_tl ... 3085
\l__enumext_label_font_style_viii_tl .. 3451
\l__enumext_label_i_tl ..... 481
\l__enumext_label_ii_tl ..... 481

```

`\l__enumext_label_iii_tl` [481](#)
`\l__enumext_label_iv_tl` [481](#)
`__enumext_label_style:Nnn` [23](#), [30](#), [333](#), [333](#), [348](#),
[486](#), [506](#), [522](#), [526](#)
`\l__enumext_label_v_tl` .. [61](#), [63](#), [519](#), [1858](#), [1927](#),
[1969](#), [1977](#), [1993](#), [2001](#), [2120](#), [2124](#), [2519](#), [2658](#), [2660](#)
`\l__enumext_label_vi_tl` . [61](#), [63](#), [519](#), [1855](#), [1924](#),
[2658](#), [2660](#), [2664](#)
`\l__enumext_label_vii_tl` . [501](#), [3012](#), [3037](#), [3044](#)
`\l__enumext_label_viii_tl` [501](#), [3356](#), [3380](#), [3384](#),
[3397](#), [3418](#)
`\l__enumext_label_width_by_box` .. [40](#), [329](#), [330](#)
`__enumext_label_width_by_box:Nn` [30](#), [327](#), [327](#),
[332](#), [344](#), [560](#), [3398](#), [3419](#)
`\l__enumext_labelsep_i_dim` ... [1973](#), [1997](#), [3401](#),
[3422](#)
`\l__enumext_labelsep_v_dim` [2581](#)
`\l__enumext_labelsep_vii_dim` . [2748](#), [2757](#), [2799](#),
[3035](#), [3095](#), [3111](#)
`\l__enumext_labelsep_viii_dim` [3159](#), [3168](#), [3210](#),
[3461](#), [3477](#)
`\l__enumext_labelwidth_i_dim` . [1972](#), [1996](#), [3401](#),
[3422](#)
`\l__enumext_labelwidth_v_dim` [2581](#)
`\l__enumext_labelwidth_vii_dim` ... [2748](#), [2756](#),
[2799](#), [3088](#), [3092](#), [3110](#)
`\l__enumext_labelwidth_viii_dim` .. [3159](#), [3167](#),
[3210](#), [3454](#), [3458](#), [3476](#)
`\l__enumext_leftmargin_tmp_v_bool` . [80](#), [2703](#)
`\l__enumext_leftmargin_tmp_X_bool` [44](#)
`\l__enumext_leftmargin_tmp_X_dim` [44](#)
`\l__enumext_leftmargin_X_dim` [44](#)
`__enumext_level:` [188](#), [188](#), [424](#), [426](#), [427](#), [435](#), [437](#),
[655](#), [659](#), [663](#), [730](#), [734](#), [738](#), [742](#), [826](#), [828](#), [830](#), [832](#),
[865](#), [867](#), [869](#), [871](#), [875](#), [915](#), [918](#), [937](#), [946](#), [952](#), [957](#),
[961](#), [972](#), [976](#), [977](#), [982](#), [1018](#), [1022](#), [1195](#), [1201](#), [1248](#),
[1250](#), [1252](#), [1255](#), [1262](#), [1264](#), [1266](#), [1269](#), [1560](#), [1568](#),
[1572](#), [1576](#), [1821](#), [1824](#), [1825](#), [2062](#), [2063](#), [2067](#), [2068](#),
[2069](#), [2076](#), [2078](#), [2082](#), [2083](#), [2086](#), [2092](#), [2093](#), [2147](#),
[2150](#), [2152](#), [2159](#), [2160](#), [2161](#), [2164](#), [2167](#), [2310](#), [2312](#),
[2340](#), [2345](#), [2346](#), [2347](#), [2349](#), [2353](#), [2358](#), [2359](#), [2360](#),
[2362](#), [2378](#), [2391](#), [2398](#), [2409](#), [2411](#), [2414](#), [2415](#), [2417](#),
[2422](#), [2429](#), [2432](#), [2434](#), [2436](#), [2437](#), [2438](#), [2439](#), [2442](#),
[2448](#), [2453](#), [2459](#), [2462](#), [2464](#), [2477](#)
`\l__enumext_level_h_int` . [20](#), [218](#), [443](#), [467](#), [1421](#),
[1438](#), [1777](#), [1794](#), [2883](#), [2884](#)
`\l__enumext_level_int` [20](#), [190](#), [227](#), [838](#), [989](#), [1182](#),
[1415](#), [1754](#), [1764](#), [1770](#), [1776](#), [1784](#), [1792](#), [1799](#), [2257](#),
[2324](#), [2325](#), [2332](#), [2376](#), [2389](#), [2444](#), [2502](#), [2545](#), [2632](#),
[2963](#), [2973](#), [3142](#), [3299](#)
`__enumext_list_arg_two_i:` [2217](#)
`__enumext_list_arg_two_ii:` [2217](#)
`__enumext_list_arg_two_iii:` [2217](#)
`__enumext_list_arg_two_iv:` [2217](#)
`__enumext_list_arg_two_v:` . [68](#), [2217](#), [2521](#), [2704](#)
`__enumext_list_arg_two_vii:` [2263](#), [2866](#)
`__enumext_list_arg_two_viii:` [2263](#), [3275](#)
`\l__enumext_listoffset_v_dim` [2583](#)
`\l__enumext_listparindent_vii_dim` [3098](#)
`\l__enumext_listparindent_viii_dim` ... [3464](#)
`__enumext_make_label:` [31](#), [66](#), [68](#), [2155](#), [2155](#), [2252](#)
`\l__enumext_mark_answer_sym_tl` . [57](#), [109](#), [1476](#),
[1630](#), [1836](#), [1982](#), [3405](#)
`\l__enumext_mark_position_str` [109](#), [1480](#), [1481](#),
[1503](#), [1504](#), [1628](#)
`\l__enumext_mark_ref_sym_tl` .. [109](#), [1490](#), [1734](#),
[1947](#)
`__enumext_mini_addvspace:` .. [45](#), [74](#), [1011](#), [1011](#),
[2419](#)
`__enumext_mini_addvspace_vii:` [47](#), [1164](#), [1164](#),
[2824](#)
`__enumext_mini_addvspace_viii:` [47](#), [1164](#), [1170](#),
[3235](#)
`__enumext_mini_env*` [905](#)
`__enumext_mini_right_cmd:n` . [48](#), [49](#), [1190](#), [1192](#),
[1192](#)
`__enumext_mini_set_vskip:` .. [43](#), [912](#), [912](#), [1013](#)
`__enumext_mini_set_vskip_vii:` [46](#), [1107](#), [1107](#),
[1166](#)
`__enumext_mini_set_vskip_viii:` [46](#), [1107](#), [1129](#),
[1172](#)
`__enumext_minipage:w` [28](#), [242](#), [244](#), [907](#), [2735](#), [3097](#),
[3463](#)
`\l__enumext_minipage_active_v_bool` .. [77](#), [78](#),
[2564](#), [2589](#), [2602](#), [2610](#)
`\g__enumext_minipage_active_vii_bool` ... [83](#),
[2835](#), [2840](#), [2852](#)
`\l__enumext_minipage_active_vii_bool` . [2820](#),
[2831](#)
`\g__enumext_minipage_active_viii_bool` [3246](#),
[3251](#), [3263](#)
`\l__enumext_minipage_active_viii_bool` [3231](#),
[3242](#)
`\g__enumext_minipage_active_X_bool` ... [155](#)
`\l__enumext_minipage_active_X_bool` [58](#)
`\g__enumext_minipage_after_skip` [58](#), [1111](#), [1123](#),
[2850](#), [3261](#)
`\l__enumext_minipage_after_skip` [43](#), [44](#), [75](#), [78](#),
[58](#), [928](#), [943](#), [963](#), [979](#), [994](#), [1000](#), [1006](#), [1020](#), [1030](#),
[1039](#), [1042](#), [1054](#), [1072](#), [1083](#), [1099](#), [1131](#), [1144](#), [1158](#),
[2486](#), [2619](#)
`\g__enumext_minipage_center_vii_bool` . [2844](#),
[2853](#)
`\g__enumext_minipage_center_viii_bool` [3255](#),
[3264](#)
`\g__enumext_minipage_center_X_bool` ... [155](#)
`\l__enumext_minipage_hsep_v_dim` ... [77](#), [2562](#)
`\l__enumext_minipage_hsep_vii_dim` [2818](#)
`\l__enumext_minipage_hsep_viii_dim` ... [3229](#)
`\l__enumext_minipage_left_skip` [43](#), [77](#), [58](#), [920](#),
[935](#), [954](#), [969](#), [1016](#), [1026](#), [1031](#), [1037](#), [1046](#), [1063](#),
[1075](#), [1095](#), [1105](#), [1109](#), [1114](#), [1118](#), [1132](#), [1136](#), [1150](#),
[1168](#), [1174](#)
`\l__enumext_minipage_left_v_dim` [77](#), [2560](#), [2568](#)
`\l__enumext_minipage_left_vii_dim` [2814](#), [2826](#)
`\l__enumext_minipage_left_viii_dim` [3225](#), [3237](#)
`\l__enumext_minipage_left_X_dim` [58](#)
`\g__enumext_minipage_right_skip` [58](#), [1110](#), [1115](#),
[1119](#), [2843](#), [3254](#)
`\l__enumext_minipage_right_skip` .. [43](#), [58](#), [924](#),
[939](#), [959](#), [974](#), [1032](#), [1038](#), [1050](#), [1068](#), [1079](#), [1133](#),
[1140](#), [1154](#), [1202](#), [1219](#)
`\l__enumext_minipage_right_v_dim` .. [77](#), [1213](#),
[1218](#), [2558](#), [2562](#)
`\g__enumext_minipage_right_vii_dim` [83](#), [2822](#),
[2842](#), [2855](#)
`\l__enumext_minipage_right_vii_dim` [83](#), [2812](#),
[2817](#), [2823](#)

`\g__enumext_minipage_right_viii_dim` .. 3233, 3253, 3266
`\l__enumext_minipage_right_viii_dim` .. 3223, 3228, 3234
`\g__enumext_minipage_right_X_dim` 155
`\g__enumext_minipage_right_X_skip` 155
`\g__enumext_minipage_stat_int` . 74, 77, 58, 1207, 1224, 2418, 2479, 2484, 2565, 2612, 2617
`\g__enumext_miniright_code_vii_tl` . 83, 2848, 2854
`\g__enumext_miniright_code_viii_tl` 3259, 3265
`\g__enumext_miniright_code_X_tl` 155
`__enumext_multi_addvspace`: ... 42, 74, 860, 860, 2450
`__enumext_multi_set_vskip`: .. 41, 824, 824, 862
`\l__enumext_multicols_above_ii_skip` ... 843
`\l__enumext_multicols_above_iii_skip` .. 849
`\l__enumext_multicols_above_iv_skip` ... 855
`\l__enumext_multicols_above_v_skip` 879, 893, 903
`\l__enumext_multicols_above_X_skip` 52
`\l__enumext_multicols_below_v_skip` 883, 897, 2604
`\l__enumext_multicols_below_X_skip` 52
`__enumext_multicols_start`: 74, 2424, 2426, 2426
`__enumext_multicols_stop`: 75, 1197, 2456, 2456, 2488
`__enumext_newlabel:nn` 25, 29, 61, 285, 285, 1810, 1913
`\l__enumext_newlabel_arg_one_tl` 25, 29, 60, 62, 133, 1733, 1803, 1811, 1902, 1914, 1945
`\l__enumext_newlabel_arg_two_tl` 25, 29, 59, 133, 1757, 1767, 1781, 1797, 1812, 1889, 1894, 1899, 1915
`__enumext_parse_keys:n` 2306, 2329, 2329
`__enumext_parse_keys_vii:n` .. 2861, 2890, 2890
`__enumext_parse_keys_viii:n` . 3271, 3304, 3304
`__enumext_parse_store_keys:n` . 72, 2335, 2338, 2338
`__enumext_parse_store_keys_vii:n` . 84, 2897, 2901, 2901
`\l__enumext_parsep_i_skip` . 841, 843, 992, 1040
`\l__enumext_parsep_ii_skip` 847, 849, 998
`\l__enumext_parsep_iii_skip` ... 853, 855, 1004
`\l__enumext_parsep_vii_skip` 3099
`\l__enumext_parsep_viii_skip` 3465
`\l__enumext_partopsep_v_skip` .. 895, 899, 1066, 1070, 1077, 1081, 1097, 1101
`\l__enumext_partopsep_viii_skip` 1142
`__enumext_phantomsection`: 29, 249, 278, 282, 298
`__enumext_print_footnote`: ... 2021, 2044, 3117, 3483
`__enumext_print_keyans_box:NN` 57, 1622, 1622, 1635, 1823, 1971, 1995, 3401, 3422
`\l__enumext_print_keyans_i_tl` 3534, 3563
`\l__enumext_print_keyans_ii_tl` ... 3539, 3564
`\l__enumext_print_keyans_iii_tl` .. 3544, 3565
`\l__enumext_print_keyans_iv_tl` ... 3549, 3566
`\l__enumext_print_keyans_vii_tl` .. 3554, 3567
`\l__enumext_print_keyans_X_tl` 98
`__enumext_printkeyans:nnn` . 98, 3568, 3571, 3571
`__enumext_redefine_item`: . 67, 2095, 2095, 2251
`\l__enumext_ref_aux_tl` 144, 424, 426, 429, 445, 447, 450, 455, 457, 460
`\l__enumext_ref_key_arg_tl` .. 144, 418, 423, 430, 442, 451, 461
`__enumext_regex_label_ref_key`: .. 32, 33, 413, 413, 425, 446, 456
`__enumext_register_counter_style:Nn` .. 317, 317, 322, 323, 324, 325, 326
`__enumext_remove_extra_parsep_vii`: .. 2876, 3126, 3126
`__enumext_remove_extra_parsep_viii`: . 3285, 3492, 3492
`__enumext_renew_footnote`: ... 2021, 2025, 3069, 3444
`\l__enumext_resume_bool` 23, 35, 1382, 2234
`__enumext_resume_counter`: . 52, 1343, 1373, 1373
`__enumext_resume_counter_star`: 1345
`__enumext_resume_counter_vii`: 52, 1352, 1373, 1384
`\g__enumext_resume_int` 23, 75, 35, 1377, 1388, 2235, 2492
`\l__enumext_resume_vii_bool` ... 35, 1393, 2281
`\g__enumext_resume_vii_int` .. 85, 35, 2282, 2942
`__enumext_safe_exec`: 2305, 2322, 2322
`__enumext_safe_exec_vii`: ... 2860, 2881, 2881
`__enumext_safe_exec_viii`: ... 3270, 3290, 3290
`__enumext_set_error:nn` 3649, 3659, 3661
`__enumext_set_label_ref:n` ... 32, 421, 421, 493
`__enumext_set_label_ref_h:n` . 33, 440, 440, 513
`__enumext_set_parse:n` 3632, 3649, 3649
`\l__enumext_setkey_tmpa_int` ... 93, 3625, 3629
`\l__enumext_setkey_tmpa_seq` 93, 3623, 3633, 3639, 3641, 3643, 3656
`\l__enumext_setkey_tmpa_tl` 93, 3631, 3635
`\l__enumext_setkey_tmpb_seq` 93, 3624, 3627, 3631, 3632
`\l__enumext_setkey_tmpb_tl` 93, 3651, 3653, 3654
`\l__enumext_show_answer_bool` . 109, 1484, 1488, 1507, 1511, 1830, 1967, 2655, 3393
`__enumext_show_length:nnn` .. 38, 198, 198, 3701, 3702, 3703, 3704, 3705, 3706, 3707, 3708, 3709, 3710, 3716, 3717, 3718, 3719, 3720, 3721, 3722, 3723, 3724, 3725
`\l__enumext_show_position_bool` 109, 1485, 1487, 1508, 1510, 1834, 1980, 2656, 3403
`\g__enumext_standar_bool` 27, 20, 220, 1441, 2508
`\l__enumext_standar_bool` . 20, 1762, 1775, 1791, 2327, 2491
`\g__enumext_standar_keyans_pic_star_env_int` 130
`\g__enumext_standar_keyans_star_env_int` 129
`\g__enumext_standar_star_env_int` .. 126, 221
`__enumext_standard_item_vii:w` .. 86, 87, 2997, 2999, 2999
`__enumext_standard_item_viii:w` 93, 3341, 3343, 3343
`\g__enumext_starred_bool` 27, 84, 85, 20, 229, 1420, 1753, 1763, 1793, 1883, 2373, 2386, 2469, 2950, 2956, 3148
`\l__enumext_starred_bool` . 84, 85, 20, 1686, 1694, 1778, 1819, 2888, 2957
`__enumext_starred_columns_set_vii`: .. 2742, 2742, 2869
`__enumext_starred_columns_set_viii`: . 3153, 3153, 3278
`__enumext_starred_item:nn` ... 2072, 2072, 2101


```

\__enumext_starred_item_exec: . 94, 3382, 3382,
    3448
\__enumext_starred_item_vii:w 86, 87, 2996, 3015,
    3015
\__enumext_starred_item_vii_aux_i:w .. 3015,
    3020, 3023
\__enumext_starred_item_vii_aux_ii:w . 3015,
    3021, 3026, 3028
\__enumext_starred_item_vii_aux_iii:w 3015,
    3031, 3040
\__enumext_starred_item_viii:w .. 93, 94, 3340,
    3359, 3359
\__enumext_starred_item_viii_aux_i:w . 3359,
    3364, 3367
\__enumext_starred_item_viii_aux_ii:w 3359,
    3365, 3375, 3377
\__enumext_starred_joined_item_vii:n . 82, 86,
    2761, 2761, 2994
\__enumext_starred_joined_item_viii:n 90, 93,
    3172, 3172, 3338
\g__enumext_starred_keyans_star_env_int 128
\g__enumext_starred_star_env_int .. 127, 230
\__enumext_start_from:NNn 34, 534, 534, 547, 569
\__enumext_start_item_tmp_vii: 84, 2872, 2979,
    2979
\__enumext_start_item_tmp_viii: 91, 3281, 3323,
    3323
\__enumext_start_item_vii:w 87, 3007, 3012, 3037,
    3044, 3046, 3046
\__enumext_start_item_viii:w .. 93, 3351, 3356,
    3380, 3425, 3425
\__enumext_start_list:nn 28, 70, 80, 236, 238, 2309,
    2518, 2669, 2864, 3273
\__enumext_start_mini_vii: . 85, 2810, 2810, 2935
\__enumext_start_mini_viii: 92, 3221, 3221, 3315
\__enumext_start_store_level: . 73, 2308, 2367,
    2367
\__enumext_start_store_level_vii: . 86, 2863,
    2959, 2959
\l__enumext_start_X_int ..... 70, 564
\__enumext_stop_item_tmp_vii: . 84, 86, 87, 2871,
    2875, 2981, 3048
\__enumext_stop_item_tmp_viii: .. 91, 93, 3280,
    3284, 3325, 3427
\__enumext_stop_item_vii: 87, 88, 3048, 3102, 3102
\__enumext_stop_item_viii: . 96, 3427, 3468, 3468
\__enumext_stop_list: .. 28, 236, 239, 2318, 2528,
    2682, 2877, 3287
\__enumext_stop_mini_vii: 83, 85, 2829, 2829, 2939
\__enumext_stop_mini_viii: . 92, 3221, 3240, 3319
\__enumext_stop_store_level: .. 73, 2319, 2367,
    2396
\__enumext_stop_store_level_vii: .. 86, 2878,
    2959, 2969
\l__enumext_store_active_bool 24, 51, 72, 84, 82,
    1366, 1375, 1386, 1648, 2333, 2371, 2384, 2533, 2540,
    2628, 2686, 2895, 2961, 2971, 3298
\__enumext_store_addto_prop:n 55, 61, 1541, 1542,
    1550, 1673, 1864, 3389
\__enumext_store_addto_seq:n 56, 63, 1551, 1551,
    1555, 1562, 1576, 1584, 1593, 1611, 1619, 1737, 1950
\l__enumext_store_ans_bool 119, 1367, 1402, 1558,
    1582, 1589, 1617, 1661
\l__enumext_store_anskey_arg_tl 24, 58, 59, 82,
    1679, 1688, 1690, 1696, 1704, 1707, 1717, 1722, 1725,
    1731, 1737
\__enumext_store_anskey_code:nnnn . 58, 1667,
    1671, 1671
\__enumext_store_anskey_show_left:n 61, 1678,
    1828, 1828
\__enumext_store_anskey_show_wrap:n 61, 1816,
    1816, 1832, 1847
\l__enumext_store_columns_break_bool . 1642,
    1685
\l__enumext_store_columns_join_int 82, 1693,
    1698
\l__enumext_store_columns_sep_vii_bool 2916
\l__enumext_store_columns_sep_vii_dim 2921,
    2925
\l__enumext_store_columns_sep_X_bool ... 98
l__enumext_store_columns_sep_X_dim .... 98
\l__enumext_store_columns_vii_bool ... 2903
\l__enumext_store_columns_vii_int 2908, 2912
\l__enumext_store_columns_X_bool ..... 98
\l__enumext_store_columns_X_int ..... 98
\__enumext_store_internal_ref: .. 58, 59, 1676,
    1739, 1739
\l__enumext_store_item_symbol_sep_dim 1640,
    1714, 1719
\l__enumext_store_item_symbol_tl . 1638, 1705,
    1709
\l__enumext_store_keyans_item_opt_tl ... 82
\l__enumext_store_keyans_label_tl 24, 61, 63,
    82, 1852, 1855, 1858, 1862, 1864, 1921, 1924, 1927,
    1931, 1941, 1950, 1951, 3369, 3384, 3387, 3389, 3391
\__enumext_store_level_close: . 56, 1556, 1580,
    2400
\__enumext_store_level_close_vii: 1587, 1615,
    2975
\__enumext_store_level_open: .. 55, 56, 72, 1556,
    1556, 2379, 2392
\__enumext_store_level_open_vii: .. 84, 1587,
    1587, 2965
\g__enumext_store_name_tl 24, 75, 82, 1461, 1464,
    2472, 2510, 2954, 3150
\l__enumext_store_name_tl 24, 51, 82, 1357, 1358,
    1360, 1362, 1364, 1379, 1390, 1544, 1546, 1553, 1805,
    1806, 1842, 1904, 1905, 1988, 2472, 2493, 2496, 2943,
    2946, 2954, 3411
\l__enumext_store_opt_vii_tl . 1591, 1601, 1607,
    1611, 2910, 2923
\l__enumext_store_opt_X_tl ..... 98
\l__enumext_store_ref_key_bool 58, 1493, 1674,
    1728, 1868, 1938
\l__enumext_store_upper_level_X_bool ... 98
\l__enumext_store_write_aux_file_tl 25, 61, 63,
    133, 1808, 1814, 1911, 1917
\__enumext_storing_set:n 51, 52, 1341, 1350, 1355,
    1355
\l__enumext_the_counter_vii_tl ..... 447
\l__enumext_the_counter_viii_tl ..... 457
\l__enumext_the_counter_X_tl ..... 144
\__enumext_tmp:n 30, 34, 44, 51, 52, 57, 64, 69, 70, 81,
    98, 108, 136, 140, 148, 154, 155, 174, 206, 210, 647,
    651, 1395, 1412, 1469, 1497, 1498, 1515, 1741, 1748,
    1749, 1770, 1784, 1787, 1799, 1870, 1877, 2217, 2262,
    2263, 2301
\__enumext_tmp:nn 349, 370, 371, 399, 400, 412, 564,

```

583, 628, 646, 704, 712, 713, 727, 792, 808, 809, 823, 1228, 1244, 1516, 1540, 2005, 2020	_enumext_wrapper_label_vii:n 3089
_enumext_tmp:nnn 481, 497, 498, 499, 500, 501, 517, 518	_enumext_wrapper_label_viii:n 3455
_enumext_tmp:nnnnn 584, 609, 612, 615, 617, 619, 622, 625	_enumext_zero_count_level: 204, 204
_enumext_tmp:w 3517, 3520	_enumext_zero_parsep: 44, 932, 987, 987
\\l_enumext_tmpa_vii_int 2752, 2755	enumext* 5, 2857
\\l_enumext_tmpa_viii_int 3163, 3166	enumXi 309
\\l_enumext_tmpa_X_int 155	enumXii 309
\\l_enumext_topsep_v_skip 881, 885, 1035, 1048, 1056, 1061, 1081, 1085, 2685, 2716	enumXiii 309
\\l_enumext_topsep_vii_skip .. 1112, 1121, 1125	enumXiv 309
\\l_enumext_topsep_viii_skip . 1134, 1156, 1160	enumXv 309
_enumext_use_key_ref: 32, 433, 433, 2253	enumXvi 309
_enumext_use_key_ref_h: .. 33, 465, 465, 2287	enumXvii 309
\\l_enumext_vspace_a_star_v_bool 1277	enumXviii 309
\\l_enumext_vspace_a_star_vii_bool ... 1299	Environments provide by enumext:
\\l_enumext_vspace_a_star_viii_bool ... 1310	enumext* 22, 23, 25–27, 30, 32–34, 37–40, 46, 47, 50–60, 62, 65, 72, 73, 84, 86, 87, 89, 91, 93, 95, 97, 100, 102
\\l_enumext_vspace_a_star_X_bool 70	enumext 22, 23, 25, 27, 30–33, 35, 36, 38–46, 48–58, 60, 62, 65–70, 72, 73, 76, 80, 81, 83, 86, 97, 100, 101
_enumext_vspace_above: .. 49, 1245, 1245, 2405	keyans* 22–24, 26, 27, 30, 32–34, 37–40, 46, 47, 50–53, 55, 62, 65, 92, 100, 102
_enumext_vspace_above_v: . 50, 1273, 1273, 2556	keyanspic .. 22–25, 30, 31, 34, 48, 51–53, 55, 56, 61–64, 78–81, 101
\\l_enumext_vspace_above_v_skip .. 1275, 1279, 1281	keyans 22–25, 27, 30, 31, 34–36, 38–40, 42, 45, 46, 48–53, 55, 56, 61–64, 68–70, 76–80, 83, 93, 100, 101
_enumext_vspace_above_vii: .. 50, 1295, 1295, 2932	Environments:
\\l_enumext_vspace_above_vii_skip 1297, 1301, 1303	enumext* 71
_enumext_vspace_above_viii: . 50, 1295, 1306, 3313	keyans* 71
\\l_enumext_vspace_above_viii_skip 1308, 1312, 1314	list 27, 28, 69, 70, 72
\\l_enumext_vspace_b_star_v_bool 1288	lrbox 81, 88, 95, 96
\\l_enumext_vspace_b_star_vii_bool ... 1321	minipage 27, 28, 40, 42, 43, 78, 80, 81, 88, 96
\\l_enumext_vspace_b_star_viii_bool ... 1332	multicols 41–43, 48, 74, 75, 77, 78
\\l_enumext_vspace_b_star_X_bool 70	exp commands:
_enumext_vspace_below: .. 50, 1259, 1259, 2490	\\exp_after:wN 3520
_enumext_vspace_below_v: . 50, 1284, 1284, 2624	\\exp_args:Ne 2331, 3508
\\l_enumext_vspace_below_v_skip .. 1286, 1290, 1292	\\exp_not:N 152, 340, 429, 450, 460, 661, 675, 676, 687, 688, 699, 700, 1733, 1839, 1840, 1943, 1985, 1986, 3408, 3409, 3517
_enumext_vspace_below_vii: .. 51, 1317, 1317, 2941	\\exp_not:n 429, 430, 450, 451, 460, 461, 662, 1524, 1531, 1698, 1709, 1719, 1733, 1734, 1811, 1914, 1945, 1947, 2349, 2362, 2912, 2925
\\l_enumext_vspace_below_vii_skip 1319, 1323, 1325	F
_enumext_vspace_below_viii: . 51, 1317, 1328, 3321	\\fbbox 1474
\\l_enumext_vspace_below_viii_skip 1330, 1334, 1336	file commands:
_enumext_widest_from:nnNn .. 35, 548, 548, 563, 575	\\file_input_stop: 3818
\\g_enumext_widest_label_tl 23, 30, 40, 337, 341, 345	first 713
\\l_enumext_wrap_label_opt_v_bool 2114	font 349
\\l_enumext_wrap_label_opt_vii_bool 87, 3006	\\footnote 65
\\l_enumext_wrap_label_opt_viii_bool 93, 3350	\\footnote 65, 2029
\\l_enumext_wrap_label_opt_X_bool 70	\\footnotemark 2039
\\l_enumext_wrap_label_v_bool 2110, 2114, 2122, 2177	\\footnotesize 1840, 1986, 3409
\\l_enumext_wrap_label_vii_bool 87, 3005, 3010, 3018, 3086	\\footnotetext 2023
\\l_enumext_wrap_label_viii_bool .. 93, 3349, 3354, 3362, 3452	G
\\l_enumext_wrap_label_X_bool 70	\\getkeyans 13, 96, 3506
_enumext_wrapper_label_v:n 2179, 2664	group commands:
	\\group_begin: .. 1660, 1838, 1984, 3065, 3084, 3407, 3440, 3450, 3528, 3562
	\\group_end: 1669, 1845, 1991, 3094, 3106, 3414, 3460, 3472, 3530, 3569
	H
	\\hbadness 3113, 3479
	hbox commands:
	\\hbox_set:Nn 329

\hfill 379, 383, 388, 389, 1199, 1217, 1733, 1943, 2834, 3245

hook commands:

 \hook_gput_code:nnn 9, 182, 186, 247

 \hook_gset_rule:nnnn 248

\hspace 3124, 3490

\hyperlink 59, 63

\hyperlink 1733, 1943

\hypertarget 29

\hypertarget 277

I

\IfHyperBoolean 255

\IfPackageLoadedTF 11, 251, 265

\ignorespaces 664

\inputlineno 221, 230

int commands:

 \int_add:Nn 2794, 3205

 \int_case:nn 838, 989, 1415, 1438

 \int_compare:nNnTF .. 218, 227, 443, 467, 914, 1033, 1178, 1182, 1186, 1458, 1652, 1656, 1853, 1887, 1892, 1897, 1922, 2325, 2376, 2389, 2428, 2444, 2458, 2479, 2502, 2541, 2545, 2574, 2599, 2612, 2632, 2636, 2691, 2764, 2774, 2790, 2884, 2963, 2973, 3119, 3128, 3142, 3175, 3185, 3201, 3293, 3299, 3485, 3494, 3629

 \int_compare_p:nNn .. 1421, 1754, 1764, 1776, 1777, 1792, 1794

 \int_decr:N 2793, 3204

 \int_eval:n 1546, 1806, 1840, 1905, 1986, 2235, 2238, 2282, 2285, 2782, 3193, 3409

 \int_from_alph:n 542, 556

 \int_from_roman:n 544, 558

 \int_gadd:Nn 2795, 3206

 \int_gdecr:N 1424, 1429, 1432, 1435, 1443

 \int_gincr:N 1665, 1954, 2060, 2090, 2418, 2565, 2983, 3061, 3327

 \int_gset:Nn 221, 230, 1377, 1388, 2037

 \int_gset_eq:NN 2034, 2492, 2495, 2942, 2945

 \int_gzero:N 208, 1207, 1224, 1466, 1467, 2484, 2617, 3137, 3503

 \int_if_exist:NTF 1368, 2493, 2943

 \int_incr:N 2324, 2537, 2690, 2883, 2982, 3292, 3326

 \int_mod:nn 3130, 3496

 \int_new:N 20, 21, 22, 23, 24, 36, 38, 58, 74, 86, 95, 103, 116, 117, 124, 125, 126, 127, 128, 129, 130, 141, 158, 159, 160, 161, 162, 1370

 \int_set:Nn 538, 542, 544, 1521, 1693, 2729, 2730, 2752, 2763, 2769, 2785, 3113, 3163, 3174, 3180, 3196, 3479, 3625

 \int_set_eq:NN 2344, 2792, 2907, 3203

 \int_step_function:nnN 1770, 1784, 1799

 \int_step_inline:nnn 2731, 3652

 \int_to_roman:n 190, 1750, 1788

 \int_use:N .. 915, 1379, 1390, 2238, 2257, 2285, 2332, 2429, 2438, 2453, 2459, 2767, 2768, 2780, 3178, 3179, 3191

 \int_zero:N 3122, 3488

 \c_one_int . 1421, 2752, 2771, 2777, 2783, 2787, 2790, 3163, 3182, 3188, 3194, 3198, 3201

 \c_zero_int .. 218, 227, 1754, 1764, 1776, 1777, 1792, 1794, 2963, 2973, 3133, 3499

\item 28, 39, 40, 56, 65, 78, 79, 81, 84, 91

\item 65–67, 86, 87, 93, 95, 240, 1564, 1570, 1595, 1603, 1690, 1924, 1927, 2097, 2131, 2870, 2872, 3279, 3281, 3391

\item* 6, 12, 2129

item-pos* 2005

item-sym* 2005

\itemindent 23, 70

\itemindent 69

itemindent 628

\itemsep 80, 81

\itemsep 2705, 2711

\itemwidth 2759, 2803, 2807, 3170, 3214, 3218

K

keyans 11, 2513

keyans* 11, 3268

keyanspic 12, 2666

Keys for environments provide by enumext:

 above* 24, 49, 50

 above 24, 49, 50, 73, 77, 85, 92

 after 38–40, 75, 78, 85, 92

 align 24, 31, 32, 68, 88

 before* 38, 39, 73, 85, 92

 before 38, 39, 77

 below* 24, 49–51

 below 24, 49–51, 75, 78, 85, 92

 check-ans 24, 25, 27, 51, 53, 58, 63, 64, 66, 67, 73, 75, 76, 89, 100

 columns-sep* 25, 55, 72, 84

 columns-sep 40, 56, 72, 74, 77, 84

 columns* 25, 55, 72, 84

 columns 23, 40, 43, 49, 56, 72, 74, 77, 84

 first 38–40, 88

 font 31, 68, 88

 item-pos* 57, 59, 65

 item-sym* 23, 57, 59, 65, 66

 item*-sep 66

 itemindent 24, 36, 37, 68, 88

 itemsep 36, 71

 labelsep 31, 66, 70, 88

 labelwidth 30, 31, 33–35, 70

 label 23, 30, 34, 35, 81

 lisparindent 71

 list-indent 23, 36, 37, 80

 list-offset 36, 37

 listparindent 36, 88

 mark-ans 25, 54, 61

 mark-pos 54, 55

 mark-ref 25, 54, 59

 mini-env .. 24, 40, 43, 48, 49, 65, 74, 77, 83, 85, 91, 92

 mini-sep 24, 40, 74, 77

 miniright* 24, 40

 miniright 24, 40, 47, 83

 minirigth* 27

 minirigth 27

 no-store 25, 53, 54

 noitemsep 36, 44

 nosep 36, 44

 parindent 71

 parsep 36, 71, 88

 partopsep 36

 ref 26, 32, 33

 resume 23, 51, 52, 70, 75, 85

 rightmargin 36

 save-ans 24, 51, 52, 55, 56, 58, 61, 63, 67, 75, 76, 79, 85, 93, 94, 96, 97

 save-key 25

 save-ref 25, 29, 54, 58, 59, 62, 63, 67, 94

 show-ans 25, 54, 55, 57, 58, 61, 67, 94

 show-length 27, 38, 70, 100

show-pos	25, 54, 55, 57, 58, 61, 64, 67, 94
start	24, 27, 34, 35, 70
store-brk	57, 58
topsep	36
widest	23, 27, 35
wrap-ans	54, 57, 61
wrap-label*	31, 66, 68, 87, 88, 93
wrap-label	31, 68, 87, 88, 93
keys commands:	
\keys_define:nn	351, 373, 402, 483, 503, 519, 566, 586, 630, 649, 706, 715, 794, 811, 1230, 1339, 1348, 1397, 1471, 1500, 1518, 1636, 2007, 3532, 3595
\l_keys_key_str	3686
\keys_set:nn	365, 818, 1235, 1240, 1682, 2331, 2552, 2894, 3308, 3597, 3598, 3599, 3600, 3601, 3602, 3603, 3604, 3605, 3606, 3607, 3608, 3646
L	
label	481, 501, 519
Labels provide by enumext:	
\Alph*	30
\Roman*	30
\alph*	30
\arabic*	30, 32
\roman*	30
\labelsep	81
\labelsep	2706, 2709
labelsep	349
\labelwidth	30, 81
\labelwidth	2706, 2707
labelwidth	349
\leftmargin	23, 70
\leftmargin	69, 2706
legacy commands:	
\legacy_if:nTF	3049, 3052, 3428, 3431
\legacy_if_gset_false:n	908
\legacy_if_set_false:n	3051, 3430
\legacy_if_set_true:n	3011, 3036, 3043, 3056, 3355, 3379, 3435
\linewidth	74, 77
\linewidth	2413, 2562, 2728, 2755, 2816, 3166, 3227
\list	28
\list	238
list-indent	628
list-offset	628
\listparindent	2708
listparindent	628
\lrbox	3066, 3441
M	
\makebox	81
\makebox	1626, 1628, 2151, 3080, 3088, 3092, 3454, 3458
\makelabel	65, 68, 69, 81
\makelabel	68, 69, 2157, 2173
\makesavenoteenv	271
mark-ans	1469
mark-pos	1469, 1498
mark-ref	1469
mini-env	792
mini-sep	792
\minipage	28
\minipage	244
\miniright	10, 48, 1176, 2482, 2615
\miniright*	10

mode commands:	
\mode_if_vertical:TF	863, 891, 1014, 1093
\mode_leave_vertical:	661, 675, 687, 699, 1595, 1603, 1624, 2149, 3078
msg commands:	
\msg_error:nn	2543, 2547, 2634, 2693, 2886, 3295, 3301, 3609
\msg_error:nnn	1180, 1184, 1209, 1226, 3522, 3527, 3592, 3662
\msg_error:nnnn	1650, 1654, 1658, 2535, 2630, 2638
\msg_fatal:nn	2326
\msg_fatal:nnn	303
\msg_info:nnn	13, 16, 253, 267
\msg_line_context:	3690, 3695, 3700, 3715, 3738, 3742, 3746, 3751, 3756, 3761, 3766, 3770, 3775, 3780, 3784, 3789, 3793, 3798, 3803, 3808, 3812, 3816
\msg_new:nnn	3663, 3667, 3671, 3675, 3680, 3684, 3688, 3693, 3698, 3713, 3728, 3735, 3740, 3744, 3749, 3754, 3759, 3764, 3768, 3773, 3778, 3782, 3787, 3791, 3796, 3801, 3806, 3810, 3814
\msg_term:nnn	1461
\msg_term:nnnn	2247, 2257, 2292, 2297
\msg_warning:nn	2481, 2614
\msg_warning:nnn	1464
\msg_warning:nnnn	1961, 2189, 2194, 2766, 2779, 3177, 3190
\multicolsep	74, 77
\multicolsep	2443, 2587

N	
\NeedsTeXFormat	3
\newcounter	306
\NewDocumentCommand	1176, 1646, 2626, 3506, 3560, 3616
\NewDocumentEnvironment	2302, 2513, 2666, 2857, 3268
\newlabel	29
\newlabel	289
no-store	1395
\noindent	84, 91
\noindent	2420, 2567, 2825, 2871, 3121, 3236, 3280, 3487
\nointerlineskip	2420, 2567, 2825, 3236
noitemsep	584
\nopagebreak	874, 902, 1025, 1104, 1167, 1173
\normalfont	1839, 1985, 3408
nosep	584

P	
Packages:	
enumext	22, 51, 69, 79, 99
enumitem	30
expl3	81
footnotehyper	29
hyperref	25, 27-29, 33, 59, 63, 87, 88, 99
lua-visual-debug	43
multicol	22, 99
shortlst	81
\par	874, 902, 1025, 1104, 1167, 1173, 1202, 1219, 1818, 2464, 2486, 2604, 2619, 2740, 2843, 2850, 3121, 3135, 3254, 3261, 3487, 3501
\parindent	3098, 3464
\parsep	41, 44, 80, 81
\parsep	1596, 1604, 2277, 2705, 2712, 2717
parsep	584
\parskip	3099, 3465
\partopsep	81
\partopsep	2278, 2710
partopsep	584

peek commands:

<code>\peek_meaning:N</code>	NTF 2988, 3002, 3019, 3030, 3332, 3346, 3363
<code>\peek_meaning_remove:N</code>	2995, 3339
<code>\peek_remove_spaces:n</code>	2135
<code>\phantomsection</code>	29
<code>\phantomsection</code>	278
prg commands:	
<code>\prg_do_nothing:</code>	282
<code>\prg_new_protected_conditional:Npnn</code>	192
<code>\prg_replicate:nn</code>	201, 3733
<code>\prg_return_false:</code>	196
<code>\prg_return_true:</code>	195
<code>\printkeyans</code>	13, 97, 3560
prop commands:	
<code>\prop_count:N</code>	1546, 1806, 1842, 1905, 1988, 3411
<code>\prop_gput_if_not_in:Nnn</code>	1541, 1544
<code>\prop_if_exist:N</code>	1358, 3526
<code>\prop_item:Nn</code>	3529
<code>\prop_new:N</code>	1360
<code>\ProvidesExplPackage</code>	4

R

<code>\raggedcolumns</code>	2452, 2593
<code>\ref</code>	59, 62
<code>ref</code>	481, 501
<code>\refstepcounter</code>	3058, 3437
regex commands:	
<code>\regex_match:nnTF</code>	194, 541, 543, 555, 557, 2342, 2355, 2905, 2918
<code>\regex_replace_once:nnN</code>	417
<code>\renewcommand</code>	429, 450, 460
<code>\RenewDocumentCommand</code>	2029, 2097, 2131, 2157, 2173
<code>\RequirePackage</code>	17
<code>resume</code>	1339
<code>resume*</code>	1339
<code>rightmargin</code>	628
<code>\Roman</code>	30, 35
<code>\Roman</code>	325
<code>\roman</code>	30, 35
<code>\roman</code>	326, 499, 3548

S

<code>save-ans</code>	1339
<code>save-ref</code>	1469
scan commands:	
<code>\scan_stop:</code>	81, 2719, 2870, 3279, 3517, 3520
seq commands:	
<code>\seq_clear:N</code>	3623
<code>\seq_const_from_clist:Nn</code>	3611
<code>\seq_count:N</code>	2679, 3627
<code>\seq_gclear:N</code>	2027, 2028
<code>\seq_gput_right:Nn</code>	1553, 2040, 2041
<code>\seq_if_empty:N</code>	2046, 3575, 3641
<code>\seq_if_exist:N</code>	1362, 3573
<code>\seq_item:Nn</code>	2737
<code>\seq_map_function:NN</code>	3632
<code>\seq_map_inline:Nn</code>	3581, 3586, 3620, 3642, 3643
<code>\seq_map_pairwise_function:NNN</code>	2048
<code>\seq_new:N</code>	96, 97, 114, 142, 143, 1364
<code>\seq_pop_left:NN</code>	3631
<code>\seq_put_right:Nn</code>	2640, 3639, 3656
<code>\seq_set_from_clist:Nn</code>	3624
<code>\seq_set_map_e:NNn</code>	3633
<code>\seq_show:N</code>	3577

<code>\setcounter</code>	552, 556, 558, 2235, 2237, 2282, 2284, 2684
<code>\setenumext</code>	6–9, 98, 3536, 3541, 3546, 3551, 3556, 3616
<code>\setlength</code>	1597, 1605
<code>show-ans</code>	1469, 1498
<code>show-length</code>	704
skip commands:	

<code>\skip_add:Nn</code>	843, 849, 855, 865, 869, 893, 897, 994, 1000, 1006, 1016, 1020, 1042, 1095, 1099, 2705
<code>\skip_eval:n</code>	1596, 1604
<code>\skip_gset:Nn</code>	1115, 1119, 1123
<code>\skip_gzero_new:N</code>	1110, 1111
<code>\skip_horizontal:N</code>	676, 688, 700, 3081, 3095, 3461
<code>\skip_horizontal:n</code>	662, 1625, 1633, 2150, 2152, 3079
<code>\skip_if_eq:nnTF</code>	841, 847, 853, 917, 951, 992, 998, 1004, 1035, 1040, 1061, 1112, 1134, 1247, 1261, 1275, 1286, 1297, 1308, 1319, 1330
<code>\skip_new:N</code>	54, 55, 59, 60, 61, 62, 63, 118, 172
<code>\skip_set:Nn</code>	826, 830, 879, 883, 920, 924, 928, 935, 939, 943, 954, 959, 963, 969, 974, 979, 1037, 1038, 1039, 1046, 1050, 1054, 1063, 1068, 1072, 1075, 1079, 1083, 1114, 1118, 1136, 1140, 1144, 1150, 1154, 1158, 2699, 2713
<code>\skip_set_eq:NN</code>	2230, 2276, 2277, 3098, 3099, 3464, 3465
<code>\skip_use:N</code>	828, 832, 867, 871, 875, 895, 899, 918, 937, 946, 952, 957, 961, 972, 976, 977, 982, 1018, 1022, 1048, 1248, 1252, 1255, 1262, 1266, 1269, 2464
<code>\skip_zero:N</code>	2278, 2443, 2587, 2710, 2711
<code>\skip_zero_new:N</code>	1030, 1031, 1032, 1109, 1131, 1132, 1133
<code>\c_zero_skip</code>	841, 847, 853, 918, 952, 992, 998, 1004, 1035, 1040, 1061, 1112, 1134, 1248, 1262, 1275, 1286, 1297, 1308, 1319, 1330
<code>\small</code>	3538, 3543, 3548, 3553, 3558
<code>\star</code>	2011
<code>start</code>	564
<code>\stepcounter</code>	2033, 2647
str commands:	
<code>\c_backslash_str</code>	3742, 3751, 3752, 3756, 3757, 3761, 3762, 3793, 3794, 3798, 3803, 3804
<code>\c_colon_str</code>	1805, 1904, 3517
<code>\str_case:nn</code>	214
<code>\str_count:n</code>	201, 3733
<code>\str_if_eq:nnTF</code>	2240, 2288
<code>\str_if_eq_p:nn</code>	2233, 2281
<code>\str_if_in:nnTF</code>	3513
<code>\str_new:N</code>	113, 167
<code>\str_set:Nn</code>	405, 406, 407, 1480, 1481, 1503, 1504
<code>\string</code>	271
<code>\strutbox</code>	922, 926, 930, 941, 945, 956, 965, 971, 981, 994, 1000, 1006, 1037, 1038, 1039, 1042, 1052, 1056, 1065, 1072, 1077, 1085, 1114, 1115, 1118, 1125, 1138, 1146, 1152, 1160, 2715

T

T_EX and L^AT_EX_{2_ε} commands:

<code>\@auxout</code>	287
<code>\@currenvir</code>	214
<code>\protected@write</code>	287
text commands:	
<code>\text_expand:n</code>	3509
<code>\textasteriskcentered</code>	1477, 1491
<code>\thepage</code>	293
tl commands:	
<code>\c_space_tl</code>	1862, 1931, 1977, 2001, 3372, 3373, 3700,

3715

`\tl_clear:N` 378, 384, 1679, 1852, 1921, 3369

`\tl_clear_new:N` 335

`\tl_const:Nn` 144, 319

`\tl_gclear:N` 1963, 2168, 2510, 2854, 3082, 3150, 3265

`\tl_gput_right:Nn` 320

`\tl_greplace_all:Nnn` 341

`\tl_gset:Nn` 1951, 2472, 2954, 3025

`\tl_gset_eq:NN` 337, 2078, 3075

`\tl_if_blank:nTF` 3073, 3385, 3395, 3416

`\tl_if_empty:nTF` . 435, 469, 475, 1560, 1591, 1705, 1959, 2147, 3654

`\tl_if_novalue:nTF` . . 1680, 1691, 1860, 1929, 1975, 1999, 2031, 2056, 2074, 2079, 2108, 2677, 2892, 3306, 3370, 3618

`\tl_map_inline:Nn` 338, 415

`\tl_new:N` 32, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 90, 91, 93, 94, 100, 101, 111, 112, 123, 133, 134, 135, 138, 146, 147, 150, 151, 166, 169

`\tl_put_left:Nn` 1568, 1601, 1688, 1969, 1993, 3384, 3391

`\tl_put_right:Nn` 336, 427, 448, 458, 1522, 1529, 1572, 1607, 1690, 1696, 1704, 1707, 1717, 1722, 1725, 1731, 1757, 1767, 1781, 1797, 1803, 1808, 1855, 1858, 1862, 1889, 1894, 1899, 1902, 1911, 1924, 1927, 1931, 1941, 1977, 2001, 2347, 2360, 2910, 2923, 3387, 3397, 3418, 3534, 3539, 3544, 3549, 3554

`\tl_remove_all:Nn` 3653

`\tl_remove_once:Nn` 1745, 1874

`\tl_replace_all:Nnn` 340

`\tl_reverse:N` 1744, 1746, 1873, 1875

`\tl_set:Nn` 152, 305, 379, 383, 388, 389, 423, 442, 659, 673, 685, 697, 1357, 1476, 1490, 1836, 1982, 2076, 3372, 3373, 3405, 3651

`\tl_set_eq:NN` 346, 424, 426, 445, 447, 455, 457, 1743,

1872, 1885, 2120, 2124, 2658, 2660

`\tl_to_str:n` 3509

`\tl_trim_spaces:n` 336, 3639, 3651, 3657

`\tl_use:N` . 342, 345, 437, 471, 477, 730, 734, 738, 742, 746, 750, 754, 758, 762, 766, 770, 774, 778, 782, 786, 790, 1630, 1750, 1758, 1769, 1783, 1788, 1800, 2063, 2069, 2093, 2111, 2115, 2123, 2159, 2160, 2167, 2175, 2176, 2182, 2310, 2519, 2663, 2848, 3085, 3096, 3100, 3259, 3451, 3462, 3466, 3563, 3564, 3565, 3566, 3567, 3635

token commands:

`\token_to_str:N` 289

`\topsep` 1597, 1605

`topsep` 584

`\typeout` 222, 231, 257, 260, 270, 271, 1425, 1444

U

`\u` 418

use commands:

`\use:N` 202, 2164, 2312

`\use:n` 3515

`\use_none:nn` 281

`\usecounter` 2231, 2279

V

`\value` 2492, 2497, 2942, 2947

`\vspace` 909, 1252, 1255, 1266, 1269, 1279, 1281, 1290, 1292, 1301, 1303, 1312, 1314, 1323, 1325, 1334, 1336, 1596, 1604, 2674, 2685, 3136, 3502

W

`widest` 564

`wrap-ans` 1469

`wrap-label` 349

`wrap-label*` 349