

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-19^{*}

©2024 by Pablo González[†]

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

Abstract

This package provides “*enumerated list*” environments for creating “*simple exercise sheets*” along with “*multiple choice questions*”, storing the `(answers)` to these in memory using the `multicol` package and the `l3seq` and `l3prop` modules.

Contents

1	Introduction	2	4	The storage system	9
1.1	Description and usage	3	4.1	Keys for storage	9
1.2	The concept of left margin	3	4.2	Keys for internal label and ref	10
1.3	User interface	3	4.3	Keys for debugging and checking	10
1.3.1	Internal counters	3	4.4	The command <code>\anskey</code>	10
1.3.2	Support for <code>multicol</code>	4	4.5	The environment <code>keyans</code>	11
1.3.3	Support for <code>minipage</code>	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	12
1.3.5	Support for <code>\footnote</code>	4	4.6.1	The command <code>\anspic</code>	12
2	The environment <code>enumext</code>	4	4.7	Printing stored content	13
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	13
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	13
3	The command <code>\setenumext</code>	5	5	Full examples	14
3.1	Keys for <code>label</code> and <code>ref</code>	6	6	The way of non-enumerated lists	16
3.2	Keys for spaces	6	7	References	18
3.2.1	Vertical spaces	7	8	Change history	18
3.2.2	Horizontal spaces	7	9	Index of Documentation	19
3.3	Keys for <code>add code</code>	8	10	Implementation	21
3.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	8	11	Index of Implementation	108
3.5	Keys for <code>multicols</code>	9			
3.6	Keys for <code>minipage</code>	9			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-05-19.

[†]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:




(A)




(B)



(C)



(D)



(E)

5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value


- Where what we are interested in the `\label` and a “short note” that we leave as an explanation, and then print them:
1. (B), $x = 5$

2. (D)

3. (C), some note

4. (B)

5. (D), “other note”
- These “simple worksheets” or “multiple choice questions” appear to be easy to obtain using a combination of the `enumerate`, `minipage` and `multicols` environments, but like many things, what “looks simple” is not so simple.
- The `enumext` package was created and designed to meet these small requirements in the creation of “simple worksheets” and “multiple choice questions”.
- ©2024 by Pablo González L
- 3 / 120

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.

```
enumext.sty  » TDS:tex/latex/enumext/
enumext.pdf  » TDS:doc/latex/enumext/
README.md   » TDS:doc/latex/enumext/
enumext.dtx  » TDS:source/latex/enumext/
```

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Support for multicol

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

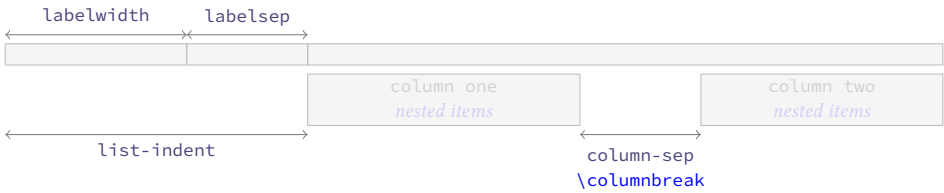


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

1.3.3 Support for minipage

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



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

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

1.3.4 The \label and \ref system

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§3.1), the standard `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§4.2) when the key `save-ans` (§4.1) is active.

- The implementation of `\label` and `\ref` together with the `save-ref` key are compatible with the `hyperref`[7] package.

1.3.5 Support for \footnote

This package provides an internal implementation for the `\footnote` command which is compatible with the `hyperref` package, but, it will not produce the expected links, and when using the `mini-env` key or the starred environments `enumext*` and `keyans*` the output will look like the classic way they are displayed in the `minipage` environment.

The best way to solve this is to use Jean-François Burnol `footnotehyper`[8] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

```
\usepackage{footnotehyper}
\makesavenoteenv{enumext}
\makesavenoteenv{enumext*}
```

2 The environment enumext

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

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

The environment can be nested with at most “*four levels*” and the options can be configured globally using `\setenumext` command and locally using `[⟨key = val⟩]` in the environment.

Example

1. This text is in the first level.
 - (a) This text is in the second level.
 - i. This text is in the third level.
 - A. This text is in the fourth level.
- X This text is in the first level.
- ★ 2. This text is in the first level.

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

2.1 The `\item*` in `enumext`

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

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

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

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

2.1.1 Keys for `\item*` in `enumext`

`item-sym*` = {`⟨symbol⟩`} default: `\star`
 Sets the `symbol` to be displayed in the “*left*” of the box containing the current `⟨label⟩` set by `labelwidth` key for `\item*` in `enumext`. The `symbol` can be in text or math mode, for example `item-sym*={\ast}`.

`item-pos*` = {`⟨rigid length | dim expression⟩`} default: *by levels*
 Sets the `offset` between the box containing the current `⟨label⟩` defined by `labelwidth` key and the `⟨symbol⟩` set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

3 The command `\setenumext`

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

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

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

- It should be kept in mind that using any *key* that sets a *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and *keyans* and *keyanspic* environments. All *keys* related to vertical or horizontal spacing accept a “*skip*” or “*dim*” expression if passed between braces, i.e. you do not need to use `\dimeval` or `\dimexpr` to perform calculations.

3.1 Keys for label and ref

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

Sets the *label* that will be printed at the *current level*. The default value for first level are `\arabic*`, for second level are `(\alph*)`, for third level are `\roman*`, and for fourth level are `\Alph*`.

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

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

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

- Internally, it renews the command associated with each counter when it is executed, i.e., `\theenumXi` is modified when the key is executed at the first level, `\theenumXii` when it is executed at the second level and `\theenumXiii` together with `\theenumXiv` when it is executed at the third and fourth levels.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

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

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

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

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

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

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

- The value of this parameter also affects the *inner levels* and the `keyans` environment. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. \TeX will enter *vertical mode* and apply this value to the “top” and “bottom” the environment or nested level.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

`itemindent` = {*rigid length*} default: 0pt

Extra *horizontal indentation*, beyond `labelsep`, of the “first line” off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.

`rightmargin` = {*rigid length*} default: 0pt

Set the *horizontal space* between the right margin of the environment and the right margin of the enclosing environment, the value it takes must be greater than or equal to 0pt. Internally sets the value of `\rightmargin` for the current level.

`listparindent` = {*rigid length*} default: 0pt

Sets the *horizontal space* indentation, beyond `list-indent`, for second and subsequent paragraphs within a list item. Internally sets the value of `\listparindent` for the current level.

`list-offset` = {*rigid length*} default: 0pt

Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.

`list-indent = {⟨rigid length⟩}` default: `labelwidth + labelsep`

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.

- If `list-indent=0pt` the `⟨label⟩` will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “common paragraph”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

3.3 Keys for add code

- The following `⟨keys⟩` should be used with “caution”, they are intended to inject `{⟨code⟩}` into different parts of the defined environments. We must keep in mind that the defined environments are based on the `list` base environment provided by L^AT_EX which is defined (simplified) as plain form `\list{⟨arg one⟩}{⟨arg two⟩}`. Using the `before*` key does not allow access to the `list` parameters defined by `[⟨key = val⟩]`.

`before = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “before” the environment starts. The `{⟨code⟩}` must be passed between braces, is executed “after” performing all calculations related to the *list parameters* in the environment and the parameters sets by `[⟨key = val⟩]` that is, in the second argument of the list after setting all the parameters `\list{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}`.

`before* = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “before” the environment starts. The `{⟨code⟩}` must be passed between braces, is executed “before” performing all calculations related to the *list parameters* and `[⟨key = val⟩]` sets in the environment that is, before the arguments defining the environment are executed: `{⟨code⟩}\list{⟨arg one⟩}{⟨arg two⟩}`.

`first = {⟨code⟩}` default: *not used*

Executes `{⟨code⟩}` when “starting” the environment. The `{⟨code⟩}` must be passed between braces, is executed right “after” all *list parameters* are done, after the second argument of list, just before the first occurrence of `\item: \list{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}\item`.

- Keep in mind that the code set in this key will affect the entire “body” of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.

`after = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “after” finishing the environment. The `{⟨code⟩}` must be passed between braces.

3.4 Keys for start, series and resume

`start = {⟨integer | string⟩}` default: `1`

Sets the *start value* of the numbering on the current level. Internally `⟨string⟩` is passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

- The following `⟨keys⟩` are “only” available for the “first level” of `enumext` and `enumext*` and are ignored if set when nested inside each other.

`series = {⟨series name⟩}` default: *not used*

Stores the *keys* of the optional argument of the “first level” of the environment in which it is executed in `{⟨series name⟩}` which is used as an argument in the key `resume`. The `⟨keys⟩` stored in `{⟨series name⟩}` are not cumulative and are overwritten if the same `{⟨series name⟩}` is used again.

`resume = {⟨series name⟩}` default: *not used*

Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` key was executed. If passed *without value* this will only set *start value* continue the numbering from the last environment in which `series={⟨series name⟩}` or `resume={⟨series name⟩}` is not present and if the `save-ans` key is active it will continue the numbering from the last environment in which it was executed. The *start value* can be overwritten using the `start` key.

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

Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` or `resume={⟨series name⟩}` keys are NOT present, if the `save-ans` key is active it will continue the numbering from the last environment in which it was executed. The *start value* can be overwritten using the `start` key.

- For security reasons the `series` key will never save in `{⟨series name⟩}` the keys `series`, `resume`, `resume*`, `save-ans`, `save-key` and `start`. When using the key `resume={⟨series name⟩}` it will have hierarchy in the `⟨keys⟩` that are saved in `{⟨series name⟩}`, in order to establish the value of a `⟨key⟩` already saved in `{⟨series name⟩}` it must be placed to the “right” of `resume={⟨series name⟩}`, the same thing happens with the `resume*` key, the exception is the `save-ans` key that must be placed on the “left” if you want to start the numbering with its value. The `resume` key passed “without value” must be exactly “without value”, i.e. `resume=` cannot be used and if executed before `resume*` it will affect the *start value*.

3.5 Keys for multicols

`columns = {⟨integer⟩}` default: 1

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

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

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

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

3.6 Keys for minipage

`mini-env = {⟨rigid length⟩}` default: not used

Sets the *width* of the `minipage` environment on the “right side”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “left side”, taking `\linewidth` as the maximum reference value.

`mini-sep = {⟨rigid length⟩}` default: 0.3333em

Sets the *space between* the `minipage` environment on the “left side” and the `minipage` environment on the “right side”. This separation is applied together with `\hfill`.

3.6.1 The command `\miniright`

`\miniright` The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”. The *starred version* ‘*’ inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `minipage` on the “right side”.

- The `\footnote{⟨text⟩}` command in `minipage` environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key.

3.6.2 The key `miniright`

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

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

Set the *code* for the drawing or tabular to be placed in the `minipage` environment on the “right side” by starting it with `\centering`.

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

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

4 The storage system

The entire mechanism for “storing content” it is activated according to `save-ans` key on the “first level” of `enumext` or `enumext*` environments and it is ignored if they are established when they are nested inside each other. Only when this *⟨key⟩* is “active” the `\anskey` command and the environments `keyans`, `keyans*` and `keyanspic` are available.

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

4.1 Keys for storage

`save-ans = {⟨store name⟩}` default: not set

Sets the *name* of the *⟨sequence⟩* and *⟨prop list⟩* in which the contents will be “stored” by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the *⟨sequence⟩* or *⟨prop list⟩* does not exist, it will be created globally and will not be overwritten if the key is used again..

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

Wraps the *current argument* passed `\anskey` command to referenced by `{#1}`. The *⟨code⟩* must be passed between braces and only affects the *⟨current argument⟩* passed to `\anskey` and NOT the “stored content” in the *⟨store name⟩* set by `save-ans` key. If this key is passed using the `\setenumext` command it is necessary to use double ‘`{##1}`’.

`wrap-opt = {⟨code {#1} more code⟩}` default: [#1]

Wraps the *optional argument* passed to the `\item*` and `\anspic*` commands referenced by `{#1}` in the `keyans`, `keyans*` and `keyanspic` environments. The `{code}` must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in *store name* set by `save-ans` key. If this key is passed using the `\setenumext` command, it is necessary to use the double `{##1}`.

`save-sep = {text symbol}` default: { }
Sets the *text symbol* that will separate the current *label* defined by the `label` key from the *optional argument* (if present), when storing them in the *store name* defined by the `save-ans` key for the `\item*` command in the `keyans` and `keyans*` environment and for the `\anspic` command in the `keyanspic` environment. The `{text symbol}` must always be passed between braces, whitespace ‘ ’ is preserved within the braces and only affects the “stored content” and not what is displayed when using the `show-ans` or `show-pos` keys.

`mark-ans = {symbol}` default: \textasteriskcentered
Sets the *symbol* to be displayed in the left margin of the “stored content” in *store name* set by `save-ans` key when using `show-ans` key.

`mark-pos = {left | right}` default: left
Sets the aligned of the *symbol* defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

4.2 Keys for internal label and ref

`save-ref = {true | false}` default: false
Activates the internal “label and ref” mechanism for referencing “stored content” in *store name* set by `save-ans` key. To reference the location of the “stored content” within the environment you must use `\ref{store name: position}`, where *position* corresponds to the position occupied by the “stored content” in the *store name* returned by the `show-pos` key. For example `\ref{test:4}` will return 3. (b) which corresponds to the location of the “stored content” at position 4 within the environment in which the key `save-ans=test` was set.

`mark-ref = {symbol}` default: \textasteriskcentered
Sets the *symbol* that will be displayed by the `\printkeyans` command only if the `hyperref` package is detected and the `save-ref` key are active. This “symbol” is used as a “link” between the environment in which the `save-ans` key was used and the place where the command is executed.

4.3 Keys for debugging and checking

`show-ans = {true | false}` default: false
Displays the *current argument* passed to `\anskey` in `enumext` environment, the current *label* for `\item*` in `keyans` environment and the current *label* for `\anspic*` in `keyanspic` environment at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown in square brackets.

`show-pos = {true | false}` default: false
Displays the *position* occupied by the “stored content” by `\anskey` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment in *store name* set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans = {true | false}` default: false
Enables the *checking answer* mechanism. This key works under the logic that each question will contain “only one answer”, it is intended to be used in conjunction with `no-store` key.

`no-store value forbidden` default: not used
This is a *meta-key* that does not receive an argument. This key is used in conjunction with `check-ans` and is designed to be used with nested levels of `enumext` in which the `\anskey` command will not be used.

4.4 The command \anskey

`\anskey {content}`

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

Example

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

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

4.5 The environment keyans

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

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

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

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

4.5.1 The `\item*` in `keyans`

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

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

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

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

Example

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

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

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

1. Text containing a question.

(A) Choice

* (B) Correct choice

(C) Choice

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

(A) Choice

(B) Choice

(C) Choice

(D) Choice

* (E) [note] Correct choice



4.6 The environment keyanspic

keyanspic

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

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

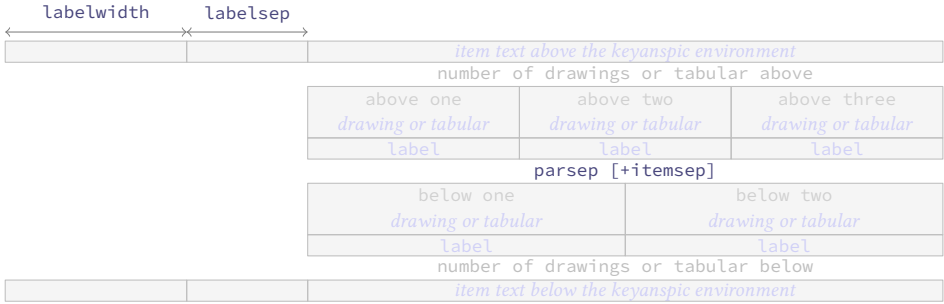


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

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

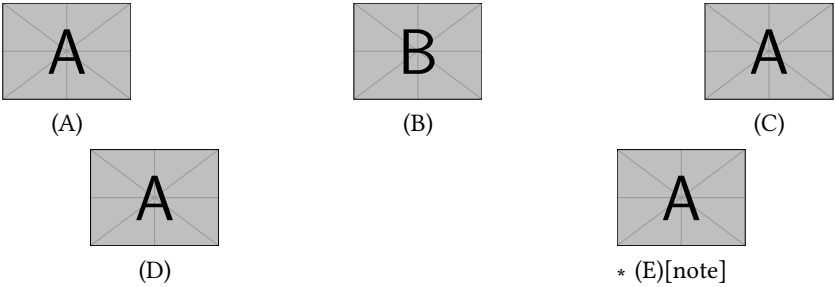
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

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

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1] $3(x + y + z)$

2. True False

(a) ~~LaTeXe~~ is cool?

[2] Very True!

3. Related to Linux

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

(b) Rate the following package and class

i. xsim

[4] very good

ii. exsheets

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

5 Full examples

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

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

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

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D


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

2. A

3. B

4. A

Example 2

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

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

A

 36 km/h.

✓

 B

 360 km/h.

C

 27,8 km/h.

D

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

✓

 A

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

B

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

C

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

D

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

A

 36 km/h.

✓

 B

 360 km/h.

C

 27,8 km/h.

D

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

✓

 A

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

B

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

C

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

D

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

2. A

3. B

4. A

*

*

*

*

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value
- B

 correct
- C

 value
- D

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

 I only
- B

 II only
- C

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

 value
- B

 value
- C

 value
4. Question with image and label below:



A



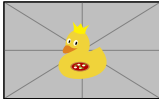
D



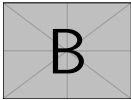
B



C



E



5. Question with image on left side:
- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value

Test keys

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

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

*

*

*

Example 4

A “simple worksheet” using ducks :) 📄.



Factor $x^2 - 2x + 1$



Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)



True False

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



Related to Linux

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

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

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

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

*

*

*

*

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

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

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

Fake description environment

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

- Something** A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

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

- Something** A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

long 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 2024-05-19 – First public release.

9 Index of Documentation

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

C

Document class:

article 2

book 2

exam 3

letter 2

report 2

\columnbreak 5

\columnsep 10

Commands provide by enumext:

\anskey 4, 10–12

\anspic* 4, 10, 11, 13

\anspic 11, 13

\getkeyans 4, 11, 14

\item* 4–7, 10–12

\item 6, 7, 9–12

\miniright 4, 5, 10

\printkeyans 4, 6, 11, 14

\setenumext 4, 6, 7, 10–12, 14

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

enumext* 4, 5, 9, 10

enumext 4–6, 9–12, 14, 17

keyans* 4, 5, 10, 11

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

keyans 4–13, 17

Environments:

enumerate 1, 3, 4, 6, 19

list 4, 9, 19

minipage 3–5, 10, 19

multicols 3, 5, 10

I

\item 4, 5

\itemsep 8

K

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 18

before* 9

before 9

below* 8

below 8

check-ans 11

columns-sep 5, 10

columns 5, 8, 10

first 9

font 7

item-pos* 6

item-sym* 6

itemindent 8

itemsep 8, 13

labelsep 4, 6–11, 18

labelwidth 4, 6, 7, 9–11, 18

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

list-indent 4, 8, 9

list-offset 4, 8, 18

listparindent 8

mark-ans 11

mark-pos 11

mark-ref 11

mini-env 5, 8, 10

mini-sep 5, 10

miniright* 10

miniright 10

no-store 11

noitemsep 8

nosep 8, 17

parsep 8, 13

partopsep 8

ref 5, 7

resume* 9

resume* 9

resume 9

rightmargin 8

save-ans 5, 9–14

save-key 9

save-ref 5, 7, 11, 14

save-sep 11

series 9

show-ans 11

show-length 7

show-pos 11, 14

start 9

topsep 8

widest 7

wrap-ans 10

wrap-label* 7, 18

wrap-label 7

wrap-opt 10

L

\label 5

Labels provide by enumext:

\Alph* 7, 12

\Roman* 7

\alph* 7

\arabic* 7

\roman* 7

\labelsep 4, 7

\labelwidth 4, 7

\linewidth 10

\listparindent 8

P

Packages:

enumerate 18

enumext 1–4, 13, 18, 19

enumitem 4, 5, 9, 18, 19

©2024 by Pablo González L

20 / 120

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

10 Implementation

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

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

10.1 General conventions

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

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

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

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

10.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

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

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

10.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-19}
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_starred_level_one_bool
28 \bool_new:N \l__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_level_one_bool
31 \bool_new:N \l__enumext_keyans_env_bool

```

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

```

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

```

Variables to store the “*name of the counters*” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment.

The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments.

The initial values of these variables are set by the function `__enumext_define_counters:Nn` (§10.8) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§10.11).

```

32 \cs_set_protected:Npn \__enumext_tmp:n #1
33 {
34   \tl_new:c { \l__enumext_counter_#1_tl }
35 }
36 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

```

Internal variables used by `ref` key (§10.11).

```

37 \tl_const:Nn \c__enumext_counter_style_tl
38 { { arabic } { roman } { Roman } { alph } { Alph } }
39 \tl_new:N \l__enumext_ref_key_arg_tl
40 \tl_new:N \l__enumext_ref_the_count_tl
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { \l__enumext_renew_the_count_#1_tl }
44   \tl_new:c { \l__enumext_the_counter_#1_tl }
45   \tl_set:ce { \l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
46 }
47 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\l__enumext_resume_name_tl
\l__enumext_resume_active_bool
\g__enumext_item_symbol_tl
\g__enumext_standar_series_tl
\g__enumext_starred_series_tl

```

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` (§?). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.27).

```

48 \int_new:N \g__enumext_resume_int
49 \int_new:N \g__enumext_resume_vii_int
50 \tl_new:N \l__enumext_resume_name_tl
51 \bool_new:N \l__enumext_resume_active_bool
52 \tl_new:N \g__enumext_item_symbol_tl
53 \tl_new:N \g__enumext_standar_series_tl
54 \tl_new:N \g__enumext_starred_series_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

55 \dim_new:N \l__enumext_current_widest_dim
56 \tl_new:N \g__enumext_counter_styles_tl
57 \tl_new:N \g__enumext_widest_label_tl
58 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

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

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

```

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

```

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

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
```

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

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

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

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

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

```
73 \int_new:N \g__enumext_minipage_stat_int
74 \skip_new:N \l__enumext_minipage_left_skip
75 \skip_new:N \l__enumext_minipage_right_skip
76 \skip_new:N \l__enumext_minipage_after_skip
77 \skip_new:N \g__enumext_minipage_right_skip
78 \skip_new:N \g__enumext_minipage_after_skip
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \dim_new:c { \l__enumext_minipage_left_#1_dim }
82   \bool_new:c { \l__enumext_minipage_active_#1_bool }
83 }
84 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_wrap_label_X_bool
\l__enumext_wrap_label_opt_X_bool
\l__enumext_start_X_int
\l__enumext_fake_item_indent_X_tl
\l__enumext_label_fill_left_X_tl
\l__enumext_label_fill_right_X_tl
\l__enumext_vspace_a_star_X_bool
\l__enumext_vspace_b_star_X_bool
```

The integer variable `\l__enumext_start_X_int` are used by the `start` key (§10.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§10.10). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```
85 \cs_set_protected:Npn \__enumext_tmp:n #1
86 {
87   \bool_new:c { \l__enumext_wrap_label_#1_bool }
88   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
89   \int_new:c { \l__enumext_start_#1_int }
90   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
91   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
92   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
93   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
94   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
95 }
96 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_columns_join_int
\l__enumext_store_keyans_label_tl
\l__enumext_store_keyans_item_opt_tl
\l__enumext_keyans_item_opt_tl
\l__enumext_keyans_tmpa_tl
\l__enumext_keyans_tmpb_tl
\l__enumext_keyans_tmpa_dim
```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§??) activates all the mechanism related to `\anskey`, `keyans`, `keyans*` and `keyanspic`.

The variable `\l__enumext_store_name_tl` sets the name for the storage in `⟨sequence⟩` and `⟨prop list⟩`, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the `check-ans` key (§??).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.25) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.29.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§10.34.1) for the `keyanspic` environment.

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

```
97 \bool_new:N \l__enumext_store_active_bool
98 \tl_new:N \l__enumext_store_name_tl
99 \tl_new:N \g__enumext_store_name_tl
100 \tl_new:N \l__enumext_store_anskey_arg_tl
101 \int_new:N \l__enumext_store_columns_join_int
102 \tl_new:N \l__enumext_store_keyans_label_tl
103 \tl_new:N \l__enumext_store_keyans_item_opt_tl
104 \tl_new:N \l__enumext_keyans_item_opt_tl
105 \tl_new:N \l__enumext_keyans_tmpa_tl
106 \tl_new:N \l__enumext_keyans_tmpb_tl
107 \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).

```
108 \tl_new:N \l__enumext_setkey_tmpa_tl
109 \tl_new:N \l__enumext_setkey_tmpb_tl
110 \int_new:N \l__enumext_setkey_tmpa_int
111 \seq_new:N \l__enumext_setkey_tmpa_seq
112 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

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

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

```
113 \cs_set_protected:Npn \l__enumext_tmp:n #1
114 {
115   \tl_new:c { l__enumext_store_opt_#1_tl }
116   \tl_new:c { l__enumext_print_keyans_#1_tl }
117   \bool_new:c { l__enumext_store_columns_#1_bool }
118   \int_new:c { l__enumext_store_columns_#1_int }
119   \bool_new:c { l__enumext_store_columns_sep_#1_bool }
120   \dim_new:c { l__enumext_store_columns_sep_#1_dim }
121   \bool_new:c { l__enumext_store_upper_level_#1_bool }
122 }
123 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

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

```
\l__enumext_show_answer_bool
\l__enumext_show_position_bool
\l__enumext_mark_ref_sym_tl
\l__enumext_mark_answer_sym_tl
\l__enumext_mark_position_str
```

Internal variables for “storage system” mechanism used by `\anskey` (§10.25), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.24).

```
124 \bool_new:N \l__enumext_show_answer_bool
125 \bool_new:N \l__enumext_show_position_bool
126 \tl_new:N \l__enumext_mark_ref_sym_tl
127 \tl_new:N \l__enumext_mark_answer_sym_tl
128 \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).

```
129 \seq_new:N \l__enumext_keyans_pic_body_seq
130 \dim_new:N \l__enumext_keyans_pic_width_dim
131 \int_new:N \l__enumext_keyans_pic_above_int
132 \int_new:N \l__enumext_keyans_pic_below_int
133 \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 (§10.23) controlled by the `check-ans` and `no-store` keys.

```
134 \bool_new:N \l__enumext_store_ans_bool
135 \bool_new:N \l__enumext_check_ans_bool
136 \bool_new:N \g__enumext_check_ans_show_bool
137 \bool_new:N \g__enumext_check_ans_show_h_bool
138 \tl_new:N \g__enumext_check_ans_item_tl
139 \int_new:N \g__enumext_count_item_anskey_int
140 \int_new:N \g__enumext_count_item_number_int
141 \int_new:N \g__enumext_standar_star_env_int
142 \int_new:N \g__enumext_starred_star_env_int
143 \int_new:N \g__enumext_starred_keyans_star_env_int
144 \int_new:N \g__enumext_standar_keyans_star_env_int
145 \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`.

```
146 \bool_new:N \l__enumext_hyperref_bool
147 \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.

```
148 \tl_new:N \l__enumext_newlabel_arg_one_tl
149 \tl_new:N \l__enumext_newlabel_arg_two_tl
150 \tl_new:N \l__enumext_store_write_aux_file_tl
151 \cs_set_protected:Npn \__enumext_tmp:n #1
152 {
153   \tl_new:c { l__enumext_label_copy_#1_tl }
154 }
155 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```
\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq
```

Internal variables used for redefinition of `\footnote`.

```
156 \int_new:N \g__enumext_footnote_int
157 \seq_new:N \g__enumext_footnote_arg_seq
158 \seq_new:N \g__enumext_footnote_int_seq
```

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

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

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

```
159 \cs_set_protected:Npn \__enumext_tmp:n #1
160 {
161   \bool_new:c { l__enumext_item_starred_#1_bool }
162   \int_new:c { l__enumext_item_column_pos_#1_int }
163   \int_new:c { g__enumext_item_count_all_#1_int }
164   \int_new:c { l__enumext_joined_item_#1_int }
165   \int_new:c { l__enumext_joined_item_aux_#1_int }
166   \int_new:c { l__enumext_tmpa_#1_int }
167   \box_new:c { l__enumext_item_text_#1_box }
168   \dim_new:c { l__enumext_joined_width_#1_dim }
169   \dim_new:c { l__enumext_item_width_#1_dim }
170   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
171   \str_new:c { l__enumext_align_label_#1_str }
172   \bool_new:c { g__enumext_minipage_active_#1_bool }
173   \tl_new:c { g__enumext_miniright_code_#1_tl }
174   \bool_new:c { g__enumext_minipage_center_#1_bool }
175   \dim_new:c { g__enumext_minipage_right_#1_dim }
176   \skip_new:c { g__enumext_minipage_right_#1_skip }
177 }
178 \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`.

```
179 \clist_const:Nn \c__enumext_all_envs_clist
180 {
181   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
182   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
183 }
```

(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.

```
184 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
185 {
186   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
187 }
```

(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.

```

188 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
189 {
190   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
191 }

```

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

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

```

192 \cs_new:Nn \__enumext_level:
193 {
194   \int_to_roman:n { \l__enumext_level_int }
195 }

```

(End of definition for `__enumext_level:`.)

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

```

\__enumext_if_is_int:nF
\__enumext_if_is_int:nTF
196 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
197 {
198   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
199   { \prg_return_true: }
200   { \prg_return_false: }
201 }

```

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

`__enumext_regex_counter_syle:` The internal function `__enumext_regex_counter_syle:` replace the ‘`*`’ with the actual counter of the running level and is used by the `ref` key. It loops through the defined counter styles in `\c__enumext_counter_style_tl` and replace ‘`*`’ by real command, for example, looking for `\arabic*` and replacing that by `\arabic{<counter>}` defined on the current level.

```

202 \cs_new_protected:Nn \__enumext_regex_counter_syle:
203 {
204   \tl_map_inline:Nn \c__enumext_counter_style_tl
205   {
206     \regex_replace_once:nnN { \c{##1}\* }
207     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
208   }
209 }

```

(End of definition for `__enumext_regex_counter_syle:`.)

`__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.

```

210 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
211 {
212   * ~ #2
213   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
214   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
215 }

```

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

`__enumext_zero_count_level:` Internal function used by `check-ans` key.

```

216 \cs_set_protected:Nn \__enumext_zero_count_level:
217 {
218   \cs_set_protected:Npn \__enumext_tmp:n ##1
219   {
220     \int_gzero:c { g__enumext_count_level_##1_int }
221   }
222   \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {##1} }
223 }

```

(End of definition for `__enumext_zero_count_level:`.)

`__enumext_current_env_set_bool:` The function `__enumext_current_env_set_bool:` 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. This function is passed to the `__enumext_safe_exec:` function in the definition of the `enumext` environment (pag 77) and to the `__enumext_safe_exec_vii:` function in the definition of the `enumext*` environment (pag 89).

```

224 \cs_new_protected:Nn \__enumext_current_env_set_bool:
225 {
226   \str_case:en { \@currenvir }
227   {
228     {enumext}
229     {
230       \bool_lazy_and:nnT
231       { \bool_not_p:n { \g__enumext_standar_bool } }
232       { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
233       {
234         \bool_gset_true:N \g__enumext_standar_bool
235         \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
236         \typeout{working-on-enumext}
237       }
238     }
239     {enumext*}
240     {
241       \bool_lazy_and:nnT
242       { \bool_not_p:n { \g__enumext_starred_bool } }
243       { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
244       {
245         \bool_gset_true:N \g__enumext_starred_bool
246         \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
247         \typeout{working-on-enumext*}
248       }
249     }
250   }
251 }

```

(End of definition for `__enumext_current_env_set_bool:`.)

10.6 Copying list and minipage environments

The `list` environment provided by L^AT_EX has the following plain form:

```

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

```

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

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

```

252 \__enumext_at_begin_document:n
253 {
254   \cs_new_eq:NN \__enumext_start_list:nn \list
255   \cs_new_eq:NN \__enumext_stop_list: \endlist
256   \cs_new_eq:NN \__enumext_item_std:w \item
257 }

```

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

The `minipage` environment provided by L^AT_EX has the following (simplified) plain form:

```

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

```

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

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

```

258 \__enumext_at_begin_document:n
259 {
260   \cs_new_eq:NN \__enumext_minipage:w \minipage
261   \cs_new_eq:NN \__enumext_endminipage: \endminipage
262 }

```

(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.

```
263 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
264 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

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

```
265 \cs_new_protected:Nn \__enumext_after_hyperref:
266 {
267   \IfPackageLoadedTF { hyperref }
268   {
269     \msg_info:nnn { enumext } { package-load } { hyperref }
270     \bool_set_true:N \l__enumext_hyperref_bool
271     \IfHyperBoolean{hyperfootnotes}
272     {
273       \typeout{hyperfootnotes=true}
274       \bool_set_true:N \l__enumext_footnotes_key_bool
275     }
276     { \typeout{hyperfootnotes=false} }
277   }
278   { }
```

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`.

```
279 \bool_if:NT \l__enumext_footnotes_key_bool
280 {
281   \IfPackageLoadedTF { footnotehyper }
282   {
283     \msg_info:nnn { enumext } { package-load } { footnotehyper }
284   }
285   {
286     \typeout{No ~ footnotehyper ~ load}
287     \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
288     \bool_set_false:N \l__enumext_footnotes_key_bool
289   }
290 }
```

The functions `__enumext_hypertarget:nn` and `__enumext_phantomsection:` correspond to the internal copies of `\hypertarget` and `\phantomsection`. If the boolean variable `\l__enumext_hy-
perref_bool` is false the functions `__enumext_hypertarget:nn` and `__enumext_phantomsection:` will be disabled.

```
291 \bool_if:NTF \l__enumext_hyperref_bool
292 {
293   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
294   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
295 }
296 {
297   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
298   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
299 }
300 }
```

(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.

```
301 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
302 {
303   \protected@write \@auxout { }
304   {
```

```

305     \token_to_str:N \newlabel {#1}
306     {
307         {#2}
308         \bool_if:NT \l__enumext_hyperref_bool
309         { { \thepage } {#2} {#1} }
310         { }
311     }
312 }
313 \__enumext_hypertarget:nn {#1} { }
314 \__enumext_phantomsection:
315 }

```

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

10.8 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

```

316 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
317 {
318     \cs_if_exist:cTF { c@ #2 }
319     { \msg_fatal:nnn { enumext } { counters }{ #2 } }
320     {
321         \tl_set:Nn #1 { #2 }
322         \newcounter { #2 }
323     }
324 }

```

(End of definition for `__enumext_define_counters:Nn`.)

```

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

```

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.

```

325 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
326 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
327 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
328 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
329 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
330 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
331 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
332 \__enumext_define_counters:Nn \l__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.

```

333 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
334 {
335     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
336     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
337 }
338 \__enumext_register_counter_style:Nn \arabic { 0 }
339 \__enumext_register_counter_style:Nn \Alph { M }
340 \__enumext_register_counter_style:Nn \alph { m }
341 \__enumext_register_counter_style:Nn \Roman { VIII }
342 \__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.

```

343 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
344 {
345   \hbox_set:Nn \__enumext_label_width_by_box {#2}
346   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
347 }
348 \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.

```

349 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
350 {
351   \tl_clear_new:N #1
352   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
353   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
354   \tl_map_inline:Nn \g__enumext_counter_styles_tl
355   {
356     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
357     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
358     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
359   }
360   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
361   { \tl_use:N \g__enumext_widest_label_tl }
362   \tl_set_eq:cN { the #2 } #1
363 }
364 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

10.10 Setting keys associated with label

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

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

```

365 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
366 {
367   \keys_define:nn { enumext / #1 }
368   {
369     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
370     font      .value_required:n = true,
371     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
372     labelsep  .initial:n   = { 0.3333em },
373     labelsep  .value_required:n = true,
374     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
375     labelwidth .value_required:n = true,
376     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
377     wrap-label .initial:n   = { ##1 },
378     wrap-label .value_required:n = true,
379     wrap-label* .code:n = {
380       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
381       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
382     },
383     wrap-label* .value_required:n = true,
384   }
385 }
386 \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.

```

387 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
388 {

```

```

389 \keys_define:nn { enumext / #1 }
390 {
391   align .choice:,
392   align / left .code:n =
393     {
394       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
395       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
396     },
397   align / right .code:n =
398     {
399       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
400       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
401     },
402   align / center .code:n =
403     {
404       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
405       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
406     },
407   align .initial:n = left,
408   align .value_required:n = true,
409 }
410 }
411 \clist_map_inline:nn
412 {
413   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
414 }
415 { \__enumext_tmp:nn #1 }

416 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
417 {
418   \keys_define:nn { enumext / #1 }
419   {
420     align .choice:,
421     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
422     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
423     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
424     align .initial:n = left,
425     align .value_required:n = true,
426   }
427 }
428 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for *align*.)

10.11 Setting label and ref keys

10.11.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

429 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
430 {
431   \keys_define:nn { enumext / #1 }
432   {
433     label .code:n = {
434       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
435       { l__enumext_counter_#2_tl } {##1}
436       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
437       \l__enumext_current_widest_dim
438     },
439     label .initial:n = #3,
440     label .value_required:n = true,
441     ref .code:n = \__enumext_standar_ref:n {##1},
442     ref .value_required:n = true,
443   }
444 }
445 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
446 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
447 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
448 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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


```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

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

```

449 \cs_new_protected:Npn \__enumext_standar_ref:n #1
450 {
451   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
452   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
453   {
454     \typeout{EMPTY}
455   }
456   {
457     \tl_set_eq:Nc
458     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
459     \__enumext_regex_counter_syle:
460     \tl_set_eq:Nc
461     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
462     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
463     {
464       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
465       { \exp_not:V \l__enumext_ref_key_arg_tl }
466     }
467   }
468 }

```

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

```

469 \cs_new_protected:Nn \__enumext_standar_ref:
470 {
471   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
472   {
473     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
474   }
475 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
476 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
477 {
478   \keys_define:nn { enumext / #1 }
479   {
480     label .code:n = {
481       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
482       { \l__enumext_counter_#2_tl } {##1}
483       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
484       \l__enumext_current_widest_dim
485     },
486     label .initial:n = #3,
487     label .value_required:n = true,
488     ref .code:n = \__enumext_starred_ref:n {##1},
489     ref .value_required:n = true,
490   }
491 }
492 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
493 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

```

\__enumext_starred_ref:n
\__enumext_starred_ref:

```

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

```

494 \cs_new_protected:Npn \__enumext_starred_ref:n #1
495 {
496   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
497   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }

```

```

498   {
499     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
500     \__enumext_regex_counter_style:
501     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
502     \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
503     {
504       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
505       { \exp_not:V \l__enumext_ref_key_arg_tl }
506     }
507   }
508   {
509     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
510     \__enumext_regex_counter_style:
511     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
512     \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
513     {
514       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
515       { \exp_not:V \l__enumext_ref_key_arg_tl }
516     }
517   }
518 }

```

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

```

519 \cs_new_protected:Nn \__enumext_starred_ref:
520 {
521   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
522   {
523     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
524     {
525       \tl_use:N \l__enumext_renew_the_count_vii_tl
526     }
527   }
528   {
529     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
530     {
531       \tl_use:N \l__enumext_renew_the_count_viii_tl
532     }
533   }
534 }

```

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

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

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

```

label
ref
\l__enumext_label_v_tl
\l__enumext_label_vi_tl
535 \keys_define:nn { enumext / keyans }
536 {
537   label .code:n = {
538     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
539     { \l__enumext_counter_v_tl } {#1}
540     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
541     \l__enumext_current_widest_dim
542     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
543     { \l__enumext_counter_vi_tl } {#1}
544     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
545     \l__enumext_current_widest_dim
546   },
547   label .initial:n = (\Alph*),
548   label .value_required:n = true,
549   ref .code:n = \__enumext_keyans_ref:n {#1},
550   ref .value_required:n = true,
551 }

```

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

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

```

552 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
553 {
554   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}

```

```

555     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
556     \__enumext_regex_counter_style:
557     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
558     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
559     {
560         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
561         { \exp_not:V \l__enumext_ref_key_arg_tl }
562     }
563
564 }

```

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

```

565 \cs_new_protected:Nn \__enumext_keyans_ref:
566 {
567     \tl_if_empty:NF \l__enumext_renew_the_count_vi_tl
568     {
569         \tl_use:N \l__enumext_renew_the_count_v_tl
570     }
571 }

```

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

10.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

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

```

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

```

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

```

572 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
573 {
574     \__enumext_if_is_int:nTF { #3 }
575     {
576         \int_set:Nn #2 {#3}
577     }
578     {
579         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
580         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
581         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
582         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
583     }
584 }
585 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

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

```

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

```

586 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
587 {
588     \__enumext_if_is_int:nTF {#4}
589     {
590         \setcounter{enumX#1} { #4 }
591     }
592     {
593         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
594         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
595         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
596         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
597     }
598     \__enumext_label_width_by_box:cv

```

```

599         { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
600     }
601 \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
602 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
603 {
604     \keys_define:nn { enumext / #1 }
605     {
606         start .code:n = {
607             \__enumext_start_from:ccn
608             { l__enumext_label_#2_tl }
609             { l__enumext_start_#2_int } {##1}
610         },
611         start .initial:n = 1,
612         widest .code:n = {
613             \__enumext_widest_from:nccn {#2}
614             { l__enumext_label_#2_tl }
615             { l__enumext_labelwidth_#2_dim } {##1}
616         },
617         widest .value_required:n = true,
618         start .value_required:n = true,
619     }
620 }
621 \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 nosepp
622 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
623 {
624     \keys_define:nn { enumext / #1 }
625     {
626         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
627         topsep .initial:n = {#3},
628         topsep .value_required:n = true,
629         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
630         partopsep .initial:n = {#4},
631         partopsep .value_required:n = true,
632         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
633         parsep .initial:n = {#5},
634         parsep .value_required:n = true,
635         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
636         itemsep .initial:n = {#6},
637         itemsep .value_required:n = true,
638         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
639         noitemsep .value_forbidden:n = true,
640         nosepp .meta:n = {
641             itemsep = 0pt, parsep = 0pt,
642             topsep = 0pt, partopsep = 0pt,
643         },
644         nosepp .value_forbidden:n = true,
645     }
646 }

```

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

```

647 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
648 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
649 { 4.0pt plus 2.0pt minus 1.0pt }
650 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
651 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
652 { 2.0pt plus 1.0pt minus 1.0pt }
653 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
654 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
655 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
656 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }

```

```

657 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
658 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
659 { 2.0pt plus 1.0pt minus 1.0pt }
660 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
661 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
662 { 4.0pt plus 2.0pt minus 1.0pt }
663 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
664 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
665 { 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.

```

666 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
667 {
668   \keys_define:nn { enumext / #1 }
669   {
670     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
671     itemindent .value_required:n = true,
672     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
673     rightmargin .value_required:n = true,
674     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
675     listparindent .value_required:n = true,
676     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
677     list-offset .value_required:n = true,
678     list-indent .code:n =
679       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
680       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
681     list-indent .value_required:n = true,
682   }
683 }
684 \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.

```

685 \cs_set_protected:Npn \__enumext_tmp:n #1
686 {
687   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
688 }
689 \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.

```

690 \cs_set_protected:Nn \__enumext_fake_item:
691 {
692   \dim_compare:nNnT
693     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
694     >
695     { \c_zero_dim }
696     {
697       \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
698       {
699         \exp_not:N \mode_leave_vertical:
700         \exp_not:n { \skip_horizontal:n }
701         { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
702         \ignorespaces
703       }
704     }
705 }
706 \cs_set_protected:Nn \__enumext_keyans_fake_item:
707 {
708   \dim_compare:nNnT
709     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
710     {

```

```

711         \tl_set:Nc \l__enumext_fake_item_indent_v_tl
712         {
713             \exp_not:N \mode_leave_vertical:
714             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
715         }
716     }
717 }
718 \cs_set_protected:Nn \__enumext_fake_item_vii:
719 {
720     \dim_compare:nNnT
721     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
722     {
723         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
724         {
725             \exp_not:N \mode_leave_vertical:
726             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
727         }
728     }
729 }
730 \cs_set_protected:Nn \__enumext_fake_item_viii:
731 {
732     \dim_compare:nNnT
733     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
734     {
735         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
736         {
737             \exp_not:N \mode_leave_vertical:
738             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
739         }
740     }
741 }

```

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

10.15 Setting show-length key

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

```

742 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
743 {
744     \keys_define:nn { enumext / #1 }
745     {
746         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
747         show-length .initial:n = false,
748     }
749 }
750 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

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

```

751 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
752 {
753     \keys_define:nn { enumext / #1 }
754     {
755         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
756         before .value_required:n = true,
757         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
758         before* .value_required:n = true,
759         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
760         after .value_required:n = true,
761         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
762         first .value_required:n = true,
763     }
764 }
765 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16.1 Functions for before, after and first keys in enumext

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

```
766 \cs_new_protected:Nn \__enumext_before_args_exec:
767 {
768   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
769 }
```

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⟩`.

```
770 \cs_new_protected:Nn \__enumext_before_keys_exec:
771 {
772   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
773 }
```

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

```
774 \cs_new_protected:Nn \__enumext_after_stop_list:
775 {
776   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
777 }
```

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

```
778 \cs_new_protected:Nn \__enumext_after_args_exec:
779 {
780   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
781 }
```

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

10.16.2 Functions for before, after and first keys in keyans

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

```
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
782 \cs_new_protected:Nn \__enumext_before_args_exec_v:
783 {
784   \tl_use:N \l__enumext_before_starred_key_v_tl
785 }
```

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

```
786 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
787 {
788   \tl_use:N \l__enumext_before_no_starred_key_v_tl
789 }
```

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

```
790 \cs_new_protected:Nn \__enumext_after_stop_list_v:
791 {
792   \tl_use:N \l__enumext_after_stop_list_v_tl
793 }
```

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.`

```
794 \cs_new_protected:Nn \__enumext_after_args_exec_v:
795 {
796   \tl_use:N \l__enumext_after_list_args_v_tl
797 }
```

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

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

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

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

```
798 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
799 {
800   \tl_use:N \l__enumext_before_starred_key_vii_tl
801 }
802 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
803 {
804   \tl_use:N \l__enumext_before_starred_key_viii_tl
805 }
```

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

```
806 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
807 {
808   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
809 }
810 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
811 {
812   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
813 }
```

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

```
814 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
815 {
816   \tl_use:N \l__enumext_after_stop_list_vii_tl
817 }
818 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
819 {
820   \tl_use:N \l__enumext_after_stop_list_viii_tl
821 }
```

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`.

```
822 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
823 {
824   \tl_use:N \l__enumext_after_list_args_vii_tl
825 }
826 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
827 {
828   \tl_use:N \l__enumext_after_list_args_viii_tl
829 }
```

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

10.17 Setting keys for multicol and minipage

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

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

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

```
830 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
831 {
832   \keys_define:nn { enumext / #1 }
833   {
834     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
835     mini-env .value_required:n = true,
836     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
837     mini-sep .initial:n = 0.3333em,
838     mini-sep .value_required:n = true,
839     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
840     columns-sep .value_required:n = true,
841     columns .int_set:c = { l__enumext_columns_#2_int },
842     columns .initial:n = 1,
843     columns .value_required:n = true,
844   }
```

```

845 }
846 \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.

847 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
848 {
849   \keys_define:nn { enumext / #1 }
850   {
851     columns      .initial:n = 2,
852     miniright    .tl_gset:c = { g__enumext_miniright_code_#2_tl },
853     miniright    .value_required:n = true,
854     miniright*   .code:n     = {
855                           \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
856                           \keys_set:nn { enumext / #1 } { miniright = {##1} }
857                         },
858     miniright*   .value_required:n = true,
859   }
860 }
861 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for multicols

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

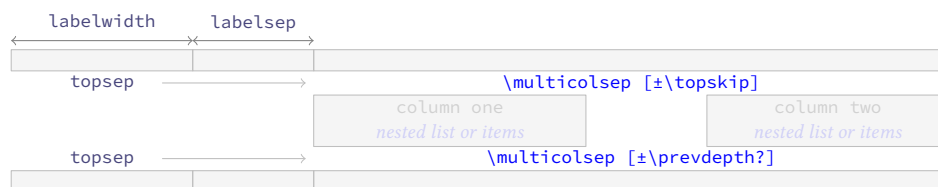


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

10.18.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

862 \cs_new_protected:Nn \__enumext_multi_set_vskip:
863 {
864   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
865   {
866     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
867   }
868   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
869   {
870     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
871   }
872   \__enumext_add_pre_parsep:
873 }

```

(End of definition for `__enumext_multi_set_vskip:`.)

`__enumext_add_pre_parsep:` The function `__enumext_add_pre_parsep:` “*adjusted*” the value of `\l__enumext_multicols_` above `_X_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces*.

```

874 \cs_new_protected:Nn \__enumext_add_pre_parsep:
875 {
876   \int_case:nn { \l__enumext_level_int }
877   {
878     { 2 }{
879       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
880       {
881         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
882       }
883     }
884     { 3 }{
885       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
886       {
887         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
888       }
889     }
890     { 4 }{
891       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
892       {
893         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
894       }
895     }
896   }
897 }

```

(End of definition for `__enumext_add_pre_parsep:`.)

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

```

898 \cs_new_protected:Nn \__enumext_multi_addvspace:
899 {
900   \__enumext_multi_set_vskip:
901   \mode_if_vertical:T
902   {
903     \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
904     {
905       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
906     }
907     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
908     {
909       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
910     }
911   }
912   \par\nopagebreak
913   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
914 }

```

(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:`

```

915 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
916 {
917   \skip_set:Nn \l__enumext_multicols_above_v_skip
918   {
919     \l__enumext_topsep_v_skip
920   }
921   \skip_set:Nn \l__enumext_multicols_below_v_skip
922   {
923     \l__enumext_topsep_v_skip
924   }
925 }
926 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
927 {

```

```

928 \__enumext_keyans_multi_set_vskip:
929 \mode_if_vertical:T
930 {
931   \skip_add:Nn \l__enumext_multicols_above_v_skip
932   {
933     \skip_use:N \l__enumext_partopsep_v_skip
934   }
935   \skip_add:Nn \l__enumext_multicols_below_v_skip
936   {
937     \skip_use:N \l__enumext_partopsep_v_skip
938   }
939 }
940 \par\nopagebreak
941 \addvspace{ \l__enumext_multicols_above_v_skip }
942 }

```

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

10.19 Adjustment of vertical spaces for minipage

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

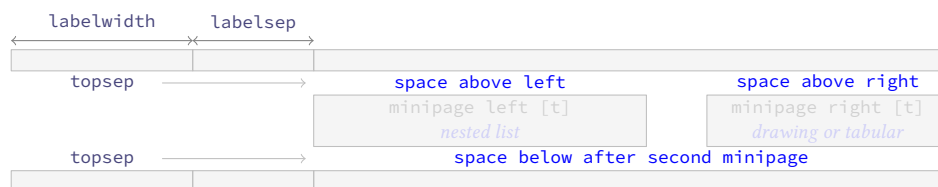


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

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

```

943 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
944 {
945   \__enumext_minipage:w [ t ] { #1 }
946   \legacy_if_gset_false:n { @minipage }
947   \vspace { 0pt }
948 }
949 { \__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`.

```

950 \cs_new_protected:Nn \__enumext_mini_set_vskip:
951 {

```

```

952 \int_compare:nNnTF
953 { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
954 {

```

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

```

955 \skip_if_eq:nnTF
956 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
957 {
958   \skip_set:Nn \l__enumext_minipage_left_skip
959   {
960     -0.150\box_dp:N \strutbox
961   }
962   \skip_set:Nn \l__enumext_minipage_right_skip
963   {
964     0.695\box_dp:N \strutbox
965   }
966   \skip_set:Nn \l__enumext_minipage_after_skip
967   {
968     \box_dp:N \strutbox
969   }
970   \__enumext_zero_parsep:
971 }
972 {
973   \skip_set:Nn \l__enumext_minipage_left_skip
974   {
975     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
976   }
977   \skip_set:Nn \l__enumext_minipage_right_skip
978   {
979     0.695\box_dp:N \strutbox
980   }
981   \skip_set:Nn \l__enumext_minipage_after_skip
982   {
983     1.85\box_dp:N \strutbox
984     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
985   }
986 }
987 }
988 {

```

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.

```

989 \skip_if_eq:nnTF
990 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
991 {
992   \skip_set:Nn \l__enumext_minipage_left_skip
993   {
994     0.5\box_dp:N \strutbox
995     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
996   }
997   \skip_set:Nn \l__enumext_minipage_right_skip
998   {
999     \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1000   }
1001   \skip_set:Nn \l__enumext_minipage_after_skip
1002   {
1003     1.6\box_dp:N \strutbox
1004   }
1005 }
1006 {
1007   \skip_set:Nn \l__enumext_minipage_left_skip
1008   {
1009     0.5875\box_dp:N \strutbox
1010     - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1011   }
1012   \skip_set:Nn \l__enumext_minipage_right_skip
1013   {
1014     + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1015     + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }

```

```

1016         }
1017         \skip_set:Nn \l__enumext_minipage_after_skip
1018         {
1019             0.325\box_dp:N \strutbox
1020             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1021         }
1022     }
1023 }
1024 }

```

(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.

```

1025 \cs_new_protected:Nn \__enumext_zero_parsep:
1026 {
1027     \int_case:nn { \l__enumext_level_int }
1028     {
1029         { 2 }{
1030             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1031             {
1032                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1033             }
1034         }
1035         { 3 }{
1036             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1037             {
1038                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1039             }
1040         }
1041         { 4 }{
1042             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1043             {
1044                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1045             }
1046         }
1047     }
1048 }

```

(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*.

```

1049 \cs_new_protected:Nn \__enumext_mini_addvspace:
1050 {
1051     \__enumext_mini_set_vskip:
1052     \mode_if_vertical:T
1053     {
1054         \skip_add:Nn \l__enumext_minipage_left_skip
1055         {
1056             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1057         }
1058         \skip_add:Nn \l__enumext_minipage_after_skip
1059         {
1060             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1061         }
1062     }
1063     \par\nopagebreak
1064     \addvspace { \l__enumext_minipage_left_skip }
1065 }

```

(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`.

```

1066 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1067 {
1068   \skip_zero_new:N \l__enumext_minipage_after_skip
1069   \skip_zero_new:N \l__enumext_minipage_left_skip
1070   \skip_zero_new:N \l__enumext_minipage_right_skip
1071   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1072   {
1073     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1074     {
1075       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1076       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1077       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1078       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1079       {
1080         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1081       }
1082     }
1083     {
1084       \skip_set:Nn \l__enumext_minipage_left_skip
1085       {
1086         \skip_use:N \l__enumext_topsep_v_skip
1087       }
1088       \skip_set:Nn \l__enumext_minipage_right_skip
1089       {
1090         0.705\box_dp:N \strutbox
1091       }
1092       \skip_set:Nn \l__enumext_minipage_after_skip
1093       {
1094         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1095       }
1096     }
1097   }
1098   {
1099     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1100     {
1101       \skip_set:Nn \l__enumext_minipage_left_skip
1102       {
1103         0.5\box_dp:N \strutbox
1104         + \l__enumext_partopsep_v_skip
1105       }
1106       \skip_set:Nn \l__enumext_minipage_right_skip
1107       {
1108         \l__enumext_partopsep_v_skip
1109       }
1110       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1111     }
1112     {
1113       \skip_set:Nn \l__enumext_minipage_left_skip
1114       {
1115         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1116       }
1117       \skip_set:Nn \l__enumext_minipage_right_skip
1118       {
1119         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1120       }
1121       \skip_set:Nn \l__enumext_minipage_after_skip
1122       {
1123         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1124       }
1125     }
1126   }
1127 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`.)

`__enumext_keyans_mini_addvspace:`

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

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

```

1128 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1129 {
1130   \__enumext_keyans_mini_set_vskip:
1131   \mode_if_vertical:T
1132   {
1133     \skip_add:Nn \l__enumext_minipage_left_skip
1134     {
1135       \l__enumext_partopsep_v_skip
1136     }
1137     \skip_add:Nn \l__enumext_minipage_after_skip
1138     {
1139       \l__enumext_partopsep_v_skip
1140     }
1141   }
1142   \par\nopagebreak
1143   \addvspace { \l__enumext_minipage_left_skip }
1144 }

```

(End of definition for `__enumext_keyans_mini_addvspace:.`)

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1145 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1146 {
1147   \skip_zero_new:N \l__enumext_minipage_left_skip
1148   \skip_gzero_new:N \g__enumext_minipage_right_skip
1149   \skip_gzero_new:N \g__enumext_minipage_after_skip
1150   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1151   {
1152     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1153     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1154   }
1155   {
1156     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1157     \skip_gset:Nn \g__enumext_minipage_right_skip
1158     {
1159       \l__enumext_topsep_vii_skip
1160     }
1161     \skip_gset:Nn \g__enumext_minipage_after_skip
1162     {
1163       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1164     }
1165   }
1166 }
1167 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1168 {
1169   \skip_zero_new:N \l__enumext_minipage_after_skip
1170   \skip_zero_new:N \l__enumext_minipage_left_skip
1171   \skip_zero_new:N \l__enumext_minipage_right_skip
1172   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1173   {
1174     \skip_set:Nn \l__enumext_minipage_left_skip
1175     {
1176       0.5\box_dp:N \strutbox
1177     }
1178     \skip_set:Nn \l__enumext_minipage_right_skip
1179     {
1180       \l__enumext_partopsep_viii_skip
1181     }
1182     \skip_set:Nn \l__enumext_minipage_after_skip
1183     {
1184       1.6\box_dp:N \strutbox
1185     }
1186   }
1187 }

```

```

1188     \skip_set:Nn \l__enumext_minipage_left_skip
1189     {
1190         0.5875\box_dp:N \strutbox
1191     }
1192     \skip_set:Nn \l__enumext_minipage_right_skip
1193     {
1194         \l__enumext_topsep_viii_skip
1195     }
1196     \skip_set:Nn \l__enumext_minipage_after_skip
1197     {
1198         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1199     }
1200 }
1201 }

```

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

`\l__enumext_mini_addvspace_vii:`
`\l__enumext_mini_addvspace_viii:`

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

```

1202 \cs_new_protected:Nn \l__enumext_mini_addvspace_vii:
1203 {
1204     \l__enumext_mini_set_vskip_vii:
1205     \par\nopagebreak
1206     \addvspace { \l__enumext_minipage_left_skip }
1207 }
1208 \cs_new_protected:Nn \l__enumext_mini_addvspace_viii:
1209 {
1210     \l__enumext_mini_set_vskip_viii:
1211     \par\nopagebreak
1212     \addvspace { \l__enumext_minipage_left_skip }
1213 }

```

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

10.19.4 The command `\miniright`

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

`\miniright`

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

```

1214 \NewDocumentCommand \miniright { s }
1215 {
1216     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1217     {
1218         \msg_error:nnn { enumext } { wrong-miniright-place }
1219     }
1220     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1221     {
1222         \msg_error:nnn { enumext } { wrong-miniright-place }
1223     }
1224     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1225     {
1226         \l__enumext_keyans_mini_right_cmd:n {#1}
1227     }
1228     { \l__enumext_mini_right_cmd:n {#1} }
1229 }

```

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

`\l__enumext_mini_right_cmd:n`

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

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

```

1230 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1231 {
1232   \dim_compare:nNnTF
1233   { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1234   {
1235     \__enumext_multicols_stop:
1236     \end{__enumext_mini_env*}
1237     \hfill
1238     \begin{__enumext_mini_env*}
1239     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1240     \par\addvspace { \l__enumext_minipage_right_skip }
1241     \bool_if:nF {#1}
1242     {
1243       \centering
1244     }
1245     \int_gzero:N \g__enumext_minipage_stat_int
1246   }
1247   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1248 }

```

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

`__enumext_keyans_mini_right_cmd:n`

The function `__enumext_keyans_mini_right_cmd:n` takes as argument the *starred version* ‘***’ of the `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1249 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1250 {
1251   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1252   {
1253     \__enumext_keyans_multicols_stop:
1254     \end{__enumext_mini_env*}
1255     \hfill
1256     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1257     \par\addvspace { \l__enumext_minipage_right_skip }
1258     \bool_if:nF {#1}
1259     {
1260       \centering
1261     }
1262     \int_gzero:N \g__enumext_minipage_stat_int
1263   }
1264   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1265 }

```

(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* 1266 \cs_set_protected:Npn \__enumext_tmp:n #1 #2
below 1267 {
below* 1268 \keys_define:nn { enumext / #1 }
1269 {
1270   above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1271   above .value_required:n = true,
1272   above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1273               \keys_set:nn { enumext / #1 } { above = {##1} },
1274   above* .value_required:n = true,
1275   below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1276   below .value_required:n = true,
1277   below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1278               \keys_set:nn { enumext / #1 } { below = {##1} },
1279   below* .value_required:n = true,

```

```

1280     }
1281 }
1282 \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.

```

1283 \cs_new_protected:Nn \__enumext_vspace_above:
1284 {
1285     \skip_if_eq:nnF
1286     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1287     {
1288         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1289         {
1290             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1291         }
1292         {
1293             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1294         }
1295     }
1296 }

```

(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.

```

1297 \cs_new_protected:Nn \__enumext_vspace_below:
1298 {
1299     \skip_if_eq:nnF
1300     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1301     {
1302         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1303         {
1304             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1305         }
1306         {
1307             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1308         }
1309     }
1310 }

```

(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.

```

1311 \cs_new_protected:Nn \__enumext_vspace_above_v:
1312 {
1313     \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1314     {
1315         \bool_if:NTF \__enumext_vspace_a_star_v_bool
1316         {
1317             \vspace*{ \__enumext_vspace_above_v_skip }
1318         }
1319         { \vspace { \__enumext_vspace_above_v_skip } }
1320     }
1321 }

```

(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.

```

1322 \cs_new_protected:Nn \__enumext_vspace_below_v:
1323 {
1324     \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1325     {
1326         \bool_if:NTF \__enumext_vspace_b_star_v_bool
1327         {
1328             \vspace*{ \__enumext_vspace_below_v_skip }

```

```

1329     }
1330     { \vspace { \l__enumext_vspace_below_v_skip } }
1331   }
1332 }

```

(End of definition for `__enumext_vspace_below_v:`)

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

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

```

1333 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1334 {
1335   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1336   {
1337     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1338     {
1339       \vspace*{ \l__enumext_vspace_above_vii_skip }
1340     }
1341     { \vspace { \l__enumext_vspace_above_vii_skip } }
1342   }
1343 }
1344 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1345 {
1346   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1347   {
1348     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1349     {
1350       \vspace*{ \l__enumext_vspace_above_viii_skip }
1351     }
1352     { \vspace { \l__enumext_vspace_above_viii_skip } }
1353   }
1354 }

```

(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.

```

1355 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1356 {
1357   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1358   {
1359     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1360     {
1361       \vspace*{ \l__enumext_vspace_below_vii_skip }
1362     }
1363     { \vspace { \l__enumext_vspace_below_vii_skip } }
1364   }
1365 }
1366 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1367 {
1368   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1369   {
1370     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1371     {
1372       \vspace*{ \l__enumext_vspace_below_viii_skip }
1373     }
1374     { \vspace { \l__enumext_vspace_below_viii_skip } }
1375   }
1376 }

```

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

10.21 Setting series, resume and resume* keys

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

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

```

series
resume
resume*
1377 \cs_set_protected:Npn \__enumext_tmp:n #1
1378 {
1379   \keys_define:nn { enumext / #1 }

```

```

1380     {
1381     series .str_set:N = \__enumext_series_str,
1382     series .value_required:n = true,
1383     resume .code:n = \__enumext_resume_series:n {##1},
1384     resume* .code:n = \__enumext_resume_starred:,
1385     resume* .value_forbidden:n = true,
1386     }
1387   }
1388   \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.21.1 Internal functions for series key

```

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

```

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

```

1389 \cs_new:Npn \__enumext_filter_series:n #1
1390   {
1391   \use:e
1392   {
1393     \keyval_parse:NNn
1394     \__enumext_filter_series_key:n
1395     \__enumext_filter_series_pair:nn {#1}
1396   }
1397   }

```

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

```

1398 \cs_new:Npn \__enumext_filter_series_key:n #1
1399   {
1400   \str_case:nnF {#1}
1401   {
1402     { resume } {}
1403     { resume* } {}
1404   }
1405   { , { \exp_not:n {#1} } }
1406   }

```

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

```

1407 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1408   {
1409   \str_case:nnF {#1}
1410   {
1411     { series } {}
1412     { resume } {}
1413     { start } {}
1414     { save-ans } {}
1415     { save-key } {}
1416   }
1417   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1418   }

```

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

```

\__enumext_parse_series:n
  \__enumext_resume_last:n

```

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

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

```

1419 \cs_new_protected:Npn \__enumext_parse_series:n #1
1420   {
1421   \str_if_empty:NTF \l__enumext_series_str
1422   {
1423     \bool_if:NF \l__enumext_resume_active_bool
1424     {

```

```

1425         \__enumext_resume_last:n {#1}
1426     }
1427 }
1428 {
1429     \tl_gclear_new:c { g__enumext_series_ \__enumext_series_str_tl }
1430     \tl_gset:ce { g__enumext_series_ \__enumext_series_str_tl }
1431     { \__enumext_filter_series:n {#1} }
1432     \int_if_exist:cF { g__enumext_series_ \__enumext_series_str_int }
1433     {
1434         \int_new:c { g__enumext_series_ \__enumext_series_str_int }
1435     }
1436 }
1437 }

```

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

```

1438 \cs_new_protected:Npn \__enumext_resume_last:n #1
1439 {
1440     \bool_if:NT \__enumext_standar_level_one_bool
1441     {
1442         \tl_gclear:N \g__enumext_standar_series_tl
1443         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1444     }
1445     \bool_if:NT \__enumext_starred_level_one_bool
1446     {
1447         \tl_gclear:N \g__enumext_starred_series_tl
1448         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1449     }
1450 }

```

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

10.21.2 Internal function to save counter value

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

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

```

1451 \cs_new_protected:Nn \__enumext_resume_save_counter:
1452 {
1453     \bool_if:NT \g__enumext_standar_bool
1454     {
1455         \tl_if_empty:NF \l__enumext_series_str
1456         {
1457             \int_gset_eq:cN
1458             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1459         }
1460         \tl_if_empty:NTF \l__enumext_resume_name_tl
1461         {
1462             \str_if_empty:NT \l__enumext_series_str
1463             {
1464                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1465             }
1466         }
1467         {
1468             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1469             {
1470                 \int_gset_eq:cN
1471                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1472             }
1473         }
1474         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }

```



```

1475         {
1476             \int_gset_eq:cN
1477             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1478         }
1479     }
1480     \bool_if:NT \g__enumext_starred_bool
1481     {
1482         \tl_if_empty:NF \l__enumext_series_str
1483         {
1484             \int_gset_eq:cN
1485             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1486         }
1487         \tl_if_empty:NTF \l__enumext_resume_name_tl
1488         {
1489             \str_if_empty:NT \l__enumext_series_str
1490             {
1491                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1492             }
1493         }
1494         {
1495             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1496             {
1497                 \int_gset_eq:cN
1498                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1499             }
1500         }
1501         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1502         {
1503             \int_gset_eq:cN
1504             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1505         }
1506     }
1507 }

```

(End of definition for __enumext_resume_save_counter:.)

10.21.3 Internal functions for resume key

__enumext_resume_series:n

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

```

1508 \cs_new_protected:Npn \__enumext_resume_series:n #1
1509 {
1510     \tl_if_empty:NTF {#1}
1511     {
1512         \__enumext_resume_counter:n { }
1513     }
1514     {
1515         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1516         {
1517             \__enumext_resume_counter:n {#1}
1518             \bool_if:NT \g__enumext_standar_bool
1519             {
1520                 \keys_set:nv { enumext / level-1 }
1521                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1522             }
1523             \bool_if:NT \g__enumext_starred_bool
1524             {
1525                 \keys_set:nv { enumext / enumext* }
1526                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1527             }
1528         }
1529         {
1530             \bool_if:NT \g__enumext_standar_bool
1531             {
1532                 \msg_error:nnn { enumext } { unknown-series } {#1}
1533             }
1534             \bool_if:NT \g__enumext_starred_bool

```

```

1535         {
1536             \msg_error:nnn { enumext } { unknown-series } {#1}
1537         }
1538     }
1539 }
1540 }

```

(End of definition for __enumext_resume_series:n.)

```

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

```

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

```

1541 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1542 {
1543   \bool_set_true:N \l__enumext_resume_active_bool
1544   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1545   \tl_if_empty:NTF \l__enumext_resume_name_tl
1546   {
1547     \__enumext_resume_counter:
1548   }
1549   {
1550     \__enumext_resume_counter_series:
1551   }
1552   \__enumext_resume_counter_save_ans:
1553 }

```

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

```

1554 \cs_new_protected:Nn \__enumext_resume_counter:
1555 {
1556   \bool_if:NT \g__enumext_standar_bool
1557   {
1558     \int_gincr:N \g__enumext_resume_int
1559     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1560   }
1561   \bool_if:NT \g__enumext_starred_bool
1562   {
1563     \int_gincr:N \g__enumext_resume_vii_int
1564     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1565   }
1566 }

```

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

```

1567 \cs_new_protected:Nn \__enumext_resume_counter_series:
1568 {
1569   \bool_if:NT \g__enumext_standar_bool
1570   {
1571     \int_set:Nn \l__enumext_start_i_int
1572     {
1573       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1574     }
1575   }
1576   \bool_if:NT \g__enumext_starred_bool
1577   {
1578     \int_set:Nn \l__enumext_start_vii_int
1579     {
1580       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1581     }
1582   }
1583 }

```

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

```

1584 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:

```

```

1585 {
1586   \bool_lazy_and:nnT
1587   { \bool_if_p:N \l__enumext_standar_level_one_bool }
1588   { \bool_if_p:N \l__enumext_store_active_bool }
1589   {
1590     \int_set:Nn \l__enumext_start_i_int
1591     {
1592       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1593     }
1594   }
1595   \bool_lazy_and:nnT
1596   { \bool_if_p:N \l__enumext_starred_level_one_bool }
1597   { \bool_if_p:N \l__enumext_store_active_bool }
1598   {
1599     \int_set:Nn \l__enumext_start_vii_int
1600     {
1601       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1602     }
1603   }
1604 }

```

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

10.21.4 Internal function for resume* key

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

```

1605 \cs_new_protected:Nn \__enumext_resume_starred:
1606 {
1607   \bool_if:NT \g__enumext_standar_bool
1608   {
1609     \tl_if_empty:NF \g__enumext_standar_series_tl
1610     {
1611       \__enumext_resume_counter:n { }
1612       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1613     }
1614   }
1615   \bool_if:NT \g__enumext_starred_bool
1616   {
1617     \tl_if_empty:NF \g__enumext_starred_series_tl
1618     {
1619       \__enumext_resume_counter:n { }
1620       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1621     }
1622   }
1623 }

```

(End of definition for `__enumext_resume_starred:`)

10.22 Setting save-ans key

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

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

```

1624 \cs_set_protected:Npn \__enumext_tmp:n #1
1625 {
1626   \keys_define:nn { enumext / #1 }
1627   {
1628     save-ans .code:n = \__enumext_storing_set:n {##1},
1629     save-ans .value_required:n = true,
1630   }
1631 }
1632 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.22.1 Internal functions for save-ans key

__enumext_storing_set:n
__enumext_storing_exec:

The function __enumext_storing_set:n first pass the value of the *save-ans* key to the variable \l__enumext_store_name_tl which will contain the “store name” of the *⟨sequence⟩* and *⟨prop list⟩* we will use. If \l__enumext_store_name_tl is empty we return an error message, otherwise we proceed to execute the function __enumext_storing_exec: for *enumext* and *enumext** environments.

```

1633 \cs_new_protected:Npn \__enumext_storing_set:n #1
1634 {
1635   \tl_set:Nx \l__enumext_store_name_tl {#1}
1636   \tl_if_empty:NTF \l__enumext_store_name_tl
1637   {
1638     \bool_if:NT \l__enumext_standar_level_one_bool
1639     {
1640       \msg_error:nnnV
1641       { enumext } { save-ans-empty } { enumext } \g__enumext_standar_star_env_int
1642     }
1643     \bool_if:NT \l__enumext_starred_level_one_bool
1644     {
1645       \msg_error:nnnV
1646       { enumext } { save-ans-empty } { enumext* } \g__enumext_starred_star_env_int
1647     }
1648   }
1649   {
1650     \bool_if:NT \l__enumext_standar_level_one_bool
1651     {
1652       \msg_note:nnnee
1653       { enumext } { save-ans-ok } { enumext }
1654       { \l__enumext_store_name_tl } { \int_use:N \g__enumext_standar_star_env_int }
1655       \__enumext_storing_exec:
1656     }
1657     \bool_if:NT \l__enumext_starred_level_one_bool
1658     {
1659       \msg_note:nnnee
1660       { enumext } { save-ans-ok } { enumext* }
1661       { \l__enumext_store_name_tl } { \int_use:N \g__enumext_starred_star_env_int }
1662       \__enumext_storing_exec:
1663     }
1664   }
1665 }

```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_bool which activates the use of the \anskey command and the *keyans*, *keyans** and *keyanspic* environments and will set to true the variable \l__enumext_store_ans_bool used for checking answers by the *check-ans* and *no-store* keys. The *⟨prop list⟩* \g__enumext_series_⟨store name⟩_prop and the *⟨sequence⟩* \g__enumext_series_⟨store name⟩_seq will be created globally to “store content” in case they do not exist together with the integer variable \g__enumext_series_⟨store name⟩_int used by the keys *resume* and *resume**.

```

1666 \cs_new_protected:Npn \__enumext_storing_exec:
1667 {
1668   \bool_set_true:N \l__enumext_store_active_bool
1669   \bool_set_true:N \l__enumext_store_ans_bool
1670   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1671   {
1672     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1673   }
1674   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1675   {
1676     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1677   }
1678   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1679   {
1680     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1681   }
1682 }

```

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

10.23 The check answer mechanism

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

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

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

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

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

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

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

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

10.23.1 Setting check-ans key

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

```

check-ans 1683 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1684 {
1685   \keys_define:nn { enumext / #1 }
1686   {
1687     check-ans .bool_set:N = \__enumext_check_ans_bool,
1688     check-ans .initial:n = false,
1689     no-store .code:n = {
1690       \bool_set_false:N \__enumext_store_ans_bool
1691       \bool_set_false:N \__enumext_check_ans_bool
1692     },
1693     no-store .value_forbidden:n = true,
1694   }
1695 }
1696 \clist_map_inline:nn
1697 {
1698   level-1, level-2, level-3, level-4, enumext*
1699 }
1700 { \__enumext_tmp:n {#1} }
```

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

10.23.2 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.

```

1701 \cs_new_protected:Nn \__enumext_check_ans_set:
1702 {
1703   \int_case:nn { \__enumext_level_int }
1704   {
1705     { 1 }{
1706       \bool_lazy_all:nT
1707       {
1708         { \bool_if_p:N \g__enumext_starred_bool }
1709         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1710       }
1711       {
1712         \int_gdecr:N \g__enumext_count_item_number_int
1713         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1714       }

```

```

1715         }
1716     { 2 }{
1717         \int_gdecr:N \g__enumext_count_item_number_int
1718     }
1719     { 3 }{
1720         \int_gdecr:N \g__enumext_count_item_number_int
1721     }
1722     { 4 }{
1723         \int_gdecr:N \g__enumext_count_item_number_int
1724     }
1725 }
1726 \int_case:nn { \l__enumext_level_h_int }
1727 {
1728     { 1 }{
1729         \bool_if:NT \g__enumext_standar_bool
1730         {
1731             \int_gdecr:N \g__enumext_count_item_number_int
1732             \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1733         }
1734     }
1735 }
1736 }

```

(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.

```

1737 \cs_new_protected:Nn \__enumext_check_ans_exec:
1738 {
1739     \bool_if:NT \l__enumext_check_ans_bool
1740     {
1741         \__enumext_check_ans_set:
1742     }
1743 }

```

(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.

```

1744 \cs_new_protected:Nn \__enumext_check_ans_show:
1745 {
1746     \int_compare:nNnTF
1747     { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1748     {
1749         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1750     }
1751     {
1752         \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1753     }
1754     \int_gzero:N \g__enumext_count_item_number_int
1755     \int_gzero:N \g__enumext_count_item_anskey_int
1756 }

```

(End of definition for `__enumext_check_ans_show:`.)

10.24 Keys and functions associated with storage

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “*storage system*” and internal mechanism of “*label and ref*” only at the *first level* of `enumext` and `enumext*`.

```

1757 \cs_set_protected:Npn \__enumext_tmp:n #1
1758 {
1759     \keys_define:nn { enumext / #1 }
1760     {
1761         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1762         wrap-ans .initial:n = \fbox{##1},
1763         wrap-ans .value_required:n = true,
1764         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1765         wrap-opt .initial:n = [{##1}],

```

```

1766     wrap-opt    .value_required:n = true,
1767     save-sep    .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1768     save-sep    .initial:n = {, ~ },
1769     save-sep    .value_required:n = true,
1770     mark-ans    .tl_set:N = \__enumext_mark_answer_sym_tl,
1771     mark-ans    .initial:n = \textasteriskcentered,
1772     mark-ans    .value_required:n = true,
1773     mark-pos    .choice:,
1774     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1775     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1776     mark-pos    .initial:n = right,
1777     mark-pos    .value_required:n = true,
1778     show-ans    .bool_set:N = \__enumext_show_answer_bool,
1779     show-ans    .initial:n = false,
1780     show-ans    .value_required:n = true,
1781     show-pos    .bool_set:N = \__enumext_show_position_bool,
1782     show-pos    .initial:n = false,
1783     show-pos    .value_required:n = true,
1784     mark-ref    .tl_set:N = \__enumext_mark_ref_sym_tl,
1785     mark-ref    .initial:n = \textasteriskcentered,
1786     mark-ref    .value_required:n = true,
1787     save-ref    .bool_set:N = \__enumext_store_ref_key_bool,
1788     save-ref    .initial:n = false,
1789     save-ref    .value_required:n = true,
1790   }
1791 }
1792 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1793 \cs_set_protected:Npn \__enumext_tmp:n #1
1794 {
1795   \keys_define:nn { enumext / #1 }
1796   {
1797     mark-pos .choice:,
1798     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1799     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1800     mark-pos .initial:n = right,
1801     mark-pos .value_required:n = true,
1802     show-ans .bool_set:N = \__enumext_show_answer_bool,
1803     show-ans .initial:n = false,
1804     show-ans .value_required:n = true,
1805     show-pos .bool_set:N = \__enumext_show_position_bool,
1806     show-pos .initial:n = false,
1807     show-pos .value_required:n = true,
1808   }
1809 }
1810 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1811 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1812 {
1813   \keys_define:nn { enumext / #1 }
1814   {
1815     columns* .code:n = \bool_set_true:c { \__enumext_store_columns_#2_bool }
1816               \int_set:cn { \__enumext_store_columns_#2_int } {##1}
1817               \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1818               {
1819                 columns = \exp_not:v { \__enumext_store_columns_#2_int },
1820               },
1821     columns* .value_required:n = true,
1822     columns-sep* .code:n = \bool_set_true:c { \__enumext_store_columns_sep_#2_bool }
1823                       \dim_set:cn { \__enumext_store_columns_sep_#2_dim } {##1}
1824                       \tl_put_right:ce { \__enumext_store_opt_#2_tl }

```



```

1825         {
1826             columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dir
1827         },
1828         columns-sep* .value_required:n = true,
1829     }
1830 }
1831 \clist_map_inline:nn
1832 {
1833     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1834 }
1835 { \__enumext_tmp:nn #1 }

```

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

10.24.1 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

1836 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1837 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1838 {
1839     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1840     {
1841         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1842     }
1843     { #1 }
1844 }
1845 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

10.24.2 Function for storing content in sequence

```

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

```

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

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

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

```

1846 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1847 {
1848     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1849 }
1850 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

10.24.3 Functions for storing the list structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

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

```

1851 \cs_new_protected:Nn \__enumext_store_level_open:
1852 {
1853     \bool_if:NT \l__enumext_store_ans_bool
1854     {
1855         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1856         {
1857             \__enumext_store_addto_seq:n
1858             {
1859                 \item \begin{enumext}
1860             }
1861         }
1862         {
1863             \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1864             {

```

```

1865         \item \begin{enumext} [
1866         ]
1867         \tl_put_right:cn { \l__enumext_store_opt_ \l__enumext_level: _tl }
1868         {
1869         ]
1870         }
1871         \__enumext_store_addto_seq:v { \l__enumext_store_opt_ \l__enumext_level: _tl }
1872     }
1873 }
1874 }
1875 \cs_new_protected:Nn \__enumext_store_level_close:
1876 {
1877     \bool_if:NT \l__enumext_store_ans_bool
1878     {
1879         \__enumext_store_addto_seq:n { \end{enumext} }
1880     }
1881 }

```

(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.

```

1882 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1883 {
1884     \bool_if:NT \l__enumext_store_ans_bool
1885     {
1886         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1887         {
1888             \__enumext_store_addto_seq:n
1889             {
1890                 \item \mode_leave_vertical:
1891                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1892                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1893             }
1894         }
1895         {
1896             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1897             {
1898                 \item \mode_leave_vertical:
1899                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1900                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1901                 ]
1902             }
1903             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1904             {
1905             ]
1906             }
1907             \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1908         }
1909     }
1910 }
1911 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1912 {
1913     \bool_if:NT \l__enumext_store_ans_bool
1914     {
1915         \__enumext_store_addto_seq:n { \end{enumext*} }
1916     }
1917 }

```

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

10.24.4 Function for show marks and position

`__enumext_print_keyans_box:NN`
`__enumext_print_keyans_box:cc`

The function `__enumext_print_keyans_box:NN` print a box in the left margin with `\l__enumext_mark_answer_sym_tl` used by the `wrap-ans`, `show-ans` and `show-pos` keys. The function takes two arguments:

- #1: `\l__enumext_labelwidth_X_dim`
- #2: `\l__enumext_labelsep_X_dim`

```

1917 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1918 {
1919   \mode_leave_vertical:
1920   \skip_horizontal:n { -\dim_use:N #2 }
1921   \makebox[\opt][ r ]
1922   {
1923     \makebox[ \dim_use:N #1 ][ \__enumext_mark_position_str ]
1924     {
1925       \tl_use:N \__enumext_mark_answer_sym_tl
1926     }
1927   }
1928   \skip_horizontal:n { \dim_use:N #2 }
1929 }
1930 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

10.25 The command \anskey and internal label and ref

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

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

```

1931 \keys_define:nn { enumext / anskey }
1932 {
1933   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
1934   item-sym* .value_required:n = true,
1935   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
1936   item-pos* .value_required:n = true,
1937   store-brk .bool_set:N = \__enumext_store_columns_break_bool,
1938   store-brk .default:n = true,
1939   store-brk .value_forbidden:n = true,
1940 }

```

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)*.

```

1941 \NewDocumentCommand \anskey { d() s o +m }
1942 {
1943   \bool_if:NF \__enumext_store_active_bool
1944   {
1945     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1946   }
1947   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
1948   {
1949     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1950   }
1951   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1952   {
1953     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1954   }
1955   \group_begin:
1956     \bool_if:NT \__enumext_store_ans_bool
1957     {
1958       \bool_if:NT \__enumext_check_ans_bool
1959       {
1960         \int_gincr:N \g__enumext_count_item_anskey_int
1961       }
1962       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1963     }
1964   \group_end:
1965 }

```

(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 `__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.

```

1966 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1967 {
1968   \__enumext_store_addto_prop:n {#4}
1969   \bool_if:NT \l__enumext_store_ref_key_bool
1970   {
1971     \__enumext_store_internal_ref:
1972   }
1973   \__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`.

```

1974   \tl_clear:N \l__enumext_store_anskey_arg_tl
1975   \tl_if_novalue:nF {#3}
1976   {
1977     \keys_set:nn { enumext / anskey } {#3}
1978   }
1979   \bool_lazy_and:nnT
1980   { \bool_if_p:N \l__enumext_store_columns_break_bool }
1981   { \bool_not_p:n { \l__enumext_starred_bool } }
1982   {
1983     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1984   }
1985   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

1986   \tl_if_novalue:nF {#1}
1987   {
1988     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1989     \bool_if:NT \l__enumext_starred_bool
1990     {
1991       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1992       {
1993         ( \exp_not:V \l__enumext_store_columns_join_int )
1994       }
1995     }
1996   }

```

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`.

```

1997   \bool_if:nTF {#2}
1998   {
1999     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2000     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2001     {
2002       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2003       {
2004         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2005       }
2006     }
2007     \dim_compare:nT
2008     {
2009       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2010     }
2011     {
2012       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2013       {
2014         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2015       }
2016     }
2017     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2018   }
2019   {
2020     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2021   }

```

Finally we check if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store in `\sequence`.

```

2022 \bool_lazy_and:nnT
2023   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2024   { \bool_if_p:N \l__enumext_hyperref_bool }
2025   {
2026     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2027     {
2028       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2029       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2030     }
2031   }
2032 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2033 }

```

(End of definition for `__enumext_store_anskey_code:nnnn`.)

`__enumext_store_internal_ref:`

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

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

```

2034 \cs_new_protected:Nn \__enumext_store_internal_ref:
2035 {
2036   \cs_set_protected:Npn \__enumext_tmp:n ##1
2037   {
2038     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2039     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2040     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2041     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2042   }
2043   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2044   \cs_set:Npn \__enumext_tmp:n ##1
2045   { . \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.

```

2046 \bool_lazy_all:nT
2047 {
2048   { \bool_if_p:N \g__enumext_starred_bool }
2049   { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2050 }
2051 {
2052   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2053   { \tl_use:N \l__enumext_label_copy_vii_tl }
2054 }
2055 \bool_lazy_all:nT
2056 {
2057   { \bool_if_p:N \l__enumext_standar_bool }
2058   { \bool_if_p:N \g__enumext_starred_bool }
2059   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2060 }
2061 {
2062   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2063   {
2064     \tl_use:N \l__enumext_label_copy_vii_tl
2065     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2066   }
2067 }

```

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.

```

2068 \bool_lazy_all:nT
2069 {
2070   { \bool_if_p:N \l__enumext_standar_bool }
2071   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2072   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2073   { \bool_not_p:n { \l__enumext_starred_bool } }
2074 }
2075 {
2076   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

2077     {
2078         \tl_use:N \l__enumext_label_copy_i_tl
2079         \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2080     }
2081 }
2082 \cs_set:Npn \__enumext_tmp:n ##1
2083 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2084 \bool_lazy_all:nT
2085 {
2086     { \bool_if_p:N \l__enumext_standar_bool }
2087     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2088     { \bool_not_p:n { \g__enumext_starred_bool } }
2089     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2090 }
2091 {
2092     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2093     {
2094         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2095         . \tl_use:N \l__enumext_label_copy_vii_tl
2096     }
2097 }

```

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

```

2098     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2099     {
2100         \l__enumext_store_name_tl \c_colon_str
2101         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2102     }

```

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.

```

2103     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2104     {
2105         \__enumext_newlabel:nn
2106         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2107         { \l__enumext_newlabel_arg_two_tl }
2108     }
2109     \l__enumext_store_write_aux_file_tl
2110 }

```

(End of definition for `__enumext_store_internal_ref:`)

`__enumext_store_anskey_show_wrap:n`

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

```

2111 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2112 {
2113     \par
2114     \bool_if:NT \l__enumext_starred_bool
2115     {
2116         \cs_set:Nn \__enumext_level: { vii }
2117     }
2118     \__enumext_print_keyans_box:cc
2119     { l__enumext_labelwidth_ \__enumext_level: _dim }
2120     { l__enumext_labelsep_ \__enumext_level: _dim }
2121     \__enumext_anskey_wrapper:n { #1 }
2122 }

```

(End of definition for `__enumext_store_anskey_show_wrap:n`)

`__enumext_store_anskey_show_left:n`

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

```

2123 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2124 {
2125     \bool_if:NT \l__enumext_show_answer_bool
2126     {
2127         \__enumext_store_anskey_show_wrap:n { #1 }
2128     }
2129     \bool_if:NT \l__enumext_show_position_bool

```

```

2130     {
2131       \tl_set:Nc \l__enumext_mark_answer_sym_tl
2132       {
2133         \group_begin:
2134         \exp_not:N \normalfont
2135         \exp_not:N \footnotesize [ \int_eval:n
2136         {
2137           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2138         }
2139       ]
2140       \group_end:
2141     }
2142     \__enumext_store_anskey_show_wrap:n { #1 }
2143   }
2144 }

```

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

10.26 Common functions for keyans, keyans* and keyanspic

10.26.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

2145 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2146 {
2147   \tl_clear:N \l__enumext_store_keyans_label_tl
2148   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2149   {
2150     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2151   }
2152   {
2153     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2154   }
2155   \tl_if_novalue:nF { #1 }
2156   {
2157     % Set save-sep
2158     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2159     {
2160       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2161     }
2162     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2163   }
2164   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2165 }

```

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

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

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

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`__enumext_keyans_store_ref_aux_ii:`

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

```

2166 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2167 {
2168   \bool_if:NT \l__enumext_store_ref_key_bool
2169   {
2170     \cs_set_protected:Npn \__enumext_tmp:n ##1
2171     {
2172       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2173       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2174       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2175       \tl_reverse:c { \l__enumext_label_copy_##1_tl }

```



```

2176     }
2177     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2178     \__enumext_keyans_store_ref_aux_i:
2179 }
2180 }

```

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

```

2181 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2182 {
2183   \bool_if:NT \g__enumext_starred_bool
2184   {
2185     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2186   }
2187   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2188   {
2189     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2190     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2191   }
2192   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2193   {
2194     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2195     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2196   }
2197   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2198   {
2199     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2200     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2201   }
2202   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2203   {
2204     \l__enumext_store_name_tl \c_colon_str
2205     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2206   }
2207   \__enumext_keyans_store_ref_aux_ii:
2208 }

```

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.

```

2209 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2210 {
2211   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2212   {
2213     \__enumext_newlabel:nn
2214     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2215     { \l__enumext_newlabel_arg_two_tl }
2216   }
2217   \l__enumext_store_write_aux_file_tl
2218 }

```

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

10.26.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

2219 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2220 {
2221   \tl_clear:N \l__enumext_store_keyans_label_tl
2222   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2223   {
2224     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2225   }
2226   {
2227     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2228   }
2229   \tl_if_no_value:nF { #1 }

```

```

2230     {
2231         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2232         {
2233             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2234         }
2235         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2236     }
2237     \__enumext_keyans_addto_seq_link:
2238 }

```

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.

```

2239 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2240 {
2241     \bool_lazy_and:nnT
2242     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2243     { \bool_if_p:N \l__enumext_hyperref_bool }
2244     {
2245         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2246         {
2247             \hfill \exp_not:N \hyperlink
2248             {
2249                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2250             }
2251             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2252         }
2253     }
2254     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2255     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2256     \bool_if:NT \l__enumext_check_ans_bool
2257     {
2258         \int_gincr:N \g__enumext_count_item_anskey_int
2259     }
2260 }

```

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

10.26.4 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.

```

2261 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2262 {
2263     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2264     {
2265         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2266     }
2267     { \tl_gclear:N \g__enumext_check_ans_item_tl }
2268 }

```

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

10.26.5 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`
`__enumext_keyans_show_ans:`
`__enumext_keyans_show_pos:`
`__enumext_keyans_show_item_opt:`

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

```

2269 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2270 {
2271     \tl_if_novalue:nF { #1 }
2272     {
2273         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2274     }
2275     \bool_if:NT \l__enumext_show_answer_bool
2276     {

```

```

2277     \__enumext_keyans_show_ans:
2278   }
2279   \bool_if:NT \l__enumext_show_position_bool
2280   {
2281     \__enumext_keyans_show_pos:
2282   }
2283 }
2284 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2285 {
2286   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2287   {
2288     \bool_lazy_or:nnT
2289     { \bool_if_p:N \l__enumext_show_answer_bool }
2290     { \bool_if_p:N \l__enumext_show_position_bool }
2291     {
2292       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2293     }
2294   }
2295 }
2296 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2297 {
2298   \tl_put_left:Nn \l__enumext_label_v_tl
2299   {
2300     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2301   }
2302 }
2303 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2304 {
2305   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2306   {
2307     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2308     {
2309       \group_begin:
2310       \exp_not:N \normalfont
2311       \exp_not:N \footnotesize [ \int_eval:n
2312         {
2313           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2314         }
2315       ]
2316       \group_end:
2317     }
2318   }
2319   {
2320     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2321     {
2322       \group_begin:
2323       \exp_not:N \normalfont
2324       \exp_not:N \footnotesize [ \int_eval:n
2325         {
2326           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2327         }
2328       ]
2329       \group_end:
2330     }
2331   }
2332   \tl_put_left:Nn \l__enumext_label_v_tl
2333   {
2334     \__enumext_print_keyans_box:NN
2335     \l__enumext_labelwidth_i_dim
2336     \l__enumext_labelsep_i_dim
2337   }
2338 }

```

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

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

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

`item-sym*` Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

`item-pos*` 2339 \cs_set_protected:Npn __enumext_tmp:nn #1 #2

```

2340 {
2341   \keys_define:nn { enumext / #1 }
2342   {
2343     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2344     item-sym* .value_required:n = true,
2345     item-sym* .initial:n = { $\star$ },
2346     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2347     item-pos* .value_required:n = true,
2348   }
2349 }
2350 \clist_map_inline:nn
2351 {
2352   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2353 }
2354 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

2355 \cs_new_protected:Nn \__enumext_footnotetext:nn
2356 {
2357   \footnotetext[#1]{#2}
2358 }
2359 \cs_new_protected:Nn \__enumext_renew_footnote:
2360 {
2361   \seq_gclear:N \g__enumext_footnote_arg_seq
2362   \seq_gclear:N \g__enumext_footnote_int_seq
2363   \RenewDocumentCommand \footnote { o +m }
2364   {
2365     \tl_if_novalue:nTF {##1}
2366     {
2367       \stepcounter{footnote}
2368       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2369     }
2370     {
2371       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2372     }
2373     \footnotemark [ \g__enumext_footnote_int ]
2374     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2375     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2376   }
2377 }
2378 \cs_new_protected:Nn \__enumext_print_footnote:
2379 {
2380   \seq_if_empty:NF \g__enumext_footnote_int_seq
2381   {
2382     \seq_map_pairwise_function:NNN
2383     \g__enumext_footnote_int_seq
2384     \g__enumext_footnote_arg_seq
2385     \__enumext_footnotetext:nn
2386   }
2387 }

```

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

10.29 Redefining `\item` command

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

10.29.1 The `\item` command in `enumext`

```

\__enumext_default_item:n

```

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

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `\l__enumext_check_ans_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `__enumext_item_std:w`.

Otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the optional argument.

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `__enumext_make_label:` (§10.30).

```

2388 \cs_new_protected:Npn \__enumext_default_item:n #1
2389 {
2390   \tl_if_novalue:nTF {#1}
2391   {
2392     \bool_if:NT \l__enumext_check_ans_bool
2393     {
2394       \int_gincr:N \g__enumext_count_item_number_int
2395     }
2396     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2397     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2398   }
2399   {
2400     \bool_set_eq:cc
2401     { \l__enumext_wrap_label_ \__enumext_level: _bool }
2402     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2403     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2404   }
2405 }

```

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

`__enumext_starred_item:nn`

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

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

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

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

```

2406 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2407 {
2408   \tl_if_novalue:nF {#1}
2409   {
2410     \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2411   }
2412   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2413   \tl_if_novalue:nTF {#2}
2414   {
2415     \dim_set_eq:cc
2416     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2417     { \l__enumext_labelsep_ \__enumext_level: _dim }
2418   }
2419   {
2420     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2421   }
2422   \bool_if:NT \l__enumext_check_ans_bool
2423   {
2424     \int_gincr:N \g__enumext_count_item_number_int
2425   }
2426   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2427   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2428 }

```

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

`__enumext_redefine_item:` The function `__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment for the internal mechanism of check-answers for `check-ans` key and adding the starred `\item*` version.

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

```

2429 \cs_new_protected:Nn \__enumext_redefine_item:
2430 {
2431   \RenewDocumentCommand \item { s o o }
2432   {
2433     \bool_if:nTF {##1}
2434     {
2435       \__enumext_starred_item:nn {##2} {##3}
2436     }
2437     { \__enumext_default_item:n {##2} }
2438   }
2439 }

```

(End of definition for `__enumext_redefine_item:`)

10.29.2 The `\item` command in keyans

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

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

```

2440 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2441 {
2442   \tl_if_novalue:nTF { #1 }
2443   {
2444     \bool_set_true:N \l__enumext_wrap_label_v_bool
2445     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2446   }
2447   {
2448     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2449     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2450   }
2451 }

```

(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 $\langle label \rangle$.

```

2452 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2453 {
2454   \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2455   \__enumext_keyans_show_left:n { #1 }
2456   \bool_set_true:N \l__enumext_wrap_label_v_bool
2457   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

Recover the original value of the current $\langle label \rangle$ and *store* it first in the $\langle prop list \rangle$ (including the optional argument), run the internal “*label and ref*” system if the `save-ref` key is active and finally *store* it in the $\langle sequence \rangle$.

```

2458   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2459   \__enumext_keyans_addto_prop:n { #1 }
2460   \__enumext_keyans_store_ref:
2461   \__enumext_keyans_addto_seq:n { #1 }
2462 }

```

(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.

`__enumext_keyans_redefine_item:`

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

```

2463 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2464 {

```

```

2465 \RenewDocumentCommand \item { s o }
2466 {
2467   \bool_if:nTF {##1}
2468   {
2469     \peek_remove_spaces:n
2470     {
2471       \__enumext_keyans_starred_item:n {##2}
2472     }
2473   }
2474   {
2475     \__enumext_keyans_default_item:n {##2}
2476   }
2477 }
2478 }

```

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

10.30 Redefining `\makeLabel` command

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

10.30.1 Redefining `\makeLabel` for `enumext`

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

```

2479 \cs_new_protected:Nn \__enumext_item_starred:
2480 {
2481   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2482   {
2483     \mode_leave_vertical:
2484     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2485     \makebox[ 0pt ]{ r }{ \g__enumext_item_symbol_tl }
2486     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2487   }
2488 }

```

(End of definition for `__enumext_item_starred:`.)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makeLabel` for the `enumext` environment.

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

```

2489 \cs_new_protected:Nn \__enumext_make_label:
2490 {
2491   \RenewDocumentCommand \makeLabel { m }
2492   {
2493     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2494     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2495     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2496     {
2497       \__enumext_item_starred:
2498       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2499     }
2500     { ##1 }
2501     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2502     \tl_gclear:N \g__enumext_item_symbol_tl
2503   }
2504 }

```

(End of definition for `__enumext_make_label:`.)

10.30.2 Redefining `\makeLabel` for `keyans`

`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment.

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

```

2505 \cs_new_protected:Nn \__enumext_keyans_make_label:
2506 {
2507   \RenewDocumentCommand \makeLabel { m }
2508   {
2509     \tl_use:N \l__enumext_label_fill_left_v_tl
2510     \tl_use:N \l__enumext_label_font_style_v_tl
2511     \bool_if:NTF \l__enumext_wrap_label_v_bool

```



```

2512     {
2513         \__enumext_wrapper_label_v:n { ##1 }
2514     }
2515     { ##1 }
2516     \tl_use:N \l__enumext_label_fill_right_v_tl
2517 }
2518 }

```

(End of definition for `__enumext_keyans_make_label:`.)

10.31 Second argument of the lists

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

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

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

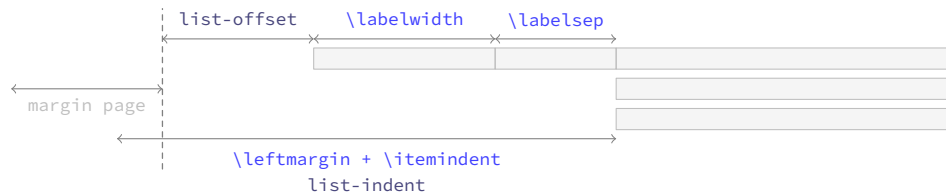


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

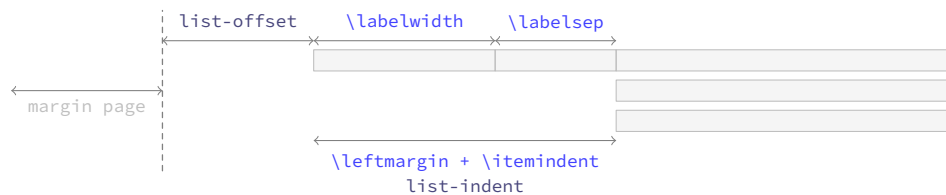


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

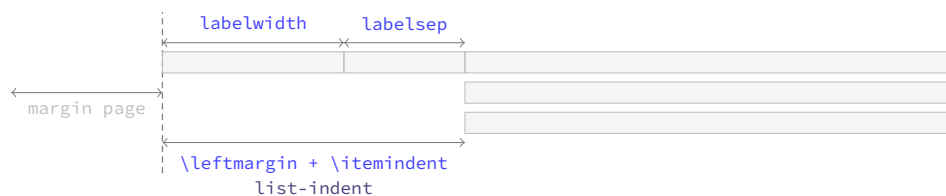


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

```

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

```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§10.31.2).

```

2519 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2520 {
2521     \dim_compare:nNt { #1 } < { \c_zero_dim }
2522     {
2523         \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2524         \dim_set:Nn #1 { \dim_abs:n { #1 } }
2525     }
2526     \dim_compare:nNt { #2 } < { \c_zero_dim }
2527     {

```

```

2528         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2529         \dim_set:Nn #2 { \dim_abs:n { #2 } }
2530     }

```

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

```

2531     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2532     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2533     {
2534         \dim_set:Nn #6 { #1 + #2 - #4 }
2535         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2536     }
2537     {
2538         \dim_compare:nNnT { #4 } = { #1 + #2 }
2539         { \dim_set:Nn #6 { \c_zero_dim } }
2540         \dim_compare:nNnT { #4 } < { #1 + #2 }
2541         { \dim_set:Nn #6 { #1 + #2 - #4 } }
2542         \dim_compare:nNnT { #4 } > { #1 + #2 }
2543         {
2544             \dim_set:Nn #6 { -#1 - #2 + #4 }
2545             \dim_set:Nn #6 { #6*-1 }
2546         }
2547         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2548     }
2549 }
2550 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

10.31.2 Setting second argument of the lists

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

```

2551 \cs_set_protected:Npn \__enumext_tmp:n #1
2552 {
2553     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2554     {
2555         \__enumext_calc_hspace:ccccc
2556         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2557         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2558         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2559         { \__enumext_leftmargin_tmp_#1_bool }
2560         \clist_map_inline:nn
2561         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2562         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2563         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2564         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2565         \usecounter { enumX#1 }
2566         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2567         \str_if_eq:nnTF { #1 } { v }
2568         {
2569             \__enumext_keyans_redefine_item:
2570             \__enumext_keyans_make_label:
2571             \__enumext_keyans_ref:
2572             \__enumext_keyans_fake_item:
2573             \bool_if:cT { \__enumext_show_length_#1_bool }
2574             {
2575                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2576             }
2577         }
2578         {
2579             \__enumext_redefine_item:
2580             \__enumext_make_label:
2581             \__enumext_standar_ref:
2582             \__enumext_fake_item:
2583             \bool_if:cT { \__enumext_show_length_#1_bool }
2584             {
2585                 \msg_term:nnne { enumext } { list-lengths } { #1 } { \int_use:N \__enumext_level_int }
2586             }
2587         }

```

```

2588     }
2589   }
2590   \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 `0pt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `parindent` to make it set the value of `listparindent` and `parsep` to set the value of `\parskip` locally.

```

2591   \cs_set_protected:Npn \__enumext_tmp:n #1
2592   {
2593     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2594     {
2595       \__enumext_calc_hspace:ccccc
2596       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2597       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2598       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2599       { \__enumext_leftmargin_tmp_#1_bool }
2600       \clist_map_inline:nn
2601       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2602       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2603       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2604       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2605       \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2606       \skip_zero:N \partopsep
2607       \usecounter { enumX#1 }
2608       \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2609       \__enumext_starred_ref:
2610       \str_if_eq:nnTF {#1} { vii }
2611       {
2612         \__enumext_fake_item_vii:
2613         \bool_if:cT { \__enumext_show_length_vii_bool }
2614         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2615       }
2616       {
2617         \__enumext_fake_item_viii:
2618         \bool_if:cT { \__enumext_show_length_#1_bool }
2619         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2620       }
2621     }
2622   }
2623   \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.

```

2624   \NewDocumentEnvironment{enumext}{0}{}
2625   {
2626     \__enumext_safe_exec:
2627     \__enumext_parse_keys:n {#1}
2628     \__enumext_before_list:
2629     \__enumext_start_store_level:
2630     \__enumext_start_list:nn
2631     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2632     {
2633       \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2634       \__enumext_before_keys_exec:
2635     }
2636     \__enumext_after_args_exec:
2637   }
2638   {
2639     \__enumext_stop_list:
2640     \__enumext_stop_store_level:
2641     \__enumext_after_list:
2642   }

```

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

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first execute the function `__enumext_current_env_set_-bool:` which will set the variable `\g__enumext_standard_bool` to “true” if the environment is not nested in `enumext*`, we increment the variable `\l__enumext_level_int` for the nesting levels and set the `\l__enumext_standard_bool` variable to “true”. Finally we set the variable `\l__enumext_standar_level_one_bool` to “true” only if the environment is not nested and we are at the first level of it.

```

2643 \cs_new_protected:Nn \__enumext_safe_exec:
2644 {
2645   \__enumext_current_env_set_bool:
2646   \int_incr:N \l__enumext_level_int
2647   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2648     { \msg_fatal:nn { enumext } { list-too-deep } }
2649   \bool_set_true:N \l__enumext_standar_bool
2650   \bool_lazy_all:nT
2651     {
2652       { \bool_if_p:N \g__enumext_standar_bool }
2653       { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2654       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2655     }
2656     {
2657       %%\typeout{[[ON-FIRST-LEVEL-ENUMEXT-NOT-NESTED]]}
2658       \bool_set_true:N \l__enumext_standar_level_one_bool
2659     }
2660 }

```

(End of definition for `__enumext_safe_exec:`)

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

```

2661 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2662 {
2663   \tl_if_novalue:nF {#1}
2664   {
2665     \str_clear:N \l__enumext_series_str
2666     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2667     {
2668       \keys_set:nn { enumext / level-1 } {#1}
2669       \__enumext_parse_series:n {#1}
2670     }
2671     {
2672       \exp_args:Ne \keys_set:nn
2673       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2674     }
2675     \bool_if:NT \l__enumext_store_active_bool
2676     {
2677       \__enumext_parse_store_keys:n {#1}
2678     }
2679   }
2680 }

```

(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:`.

```

2681 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2682 {
2683   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2684   {
2685     \regex_match:nnT { \b columns\b } {#1}
2686     {
2687       \int_set_eq:cc
2688       { \l__enumext_store_columns_ \__enumext_level: _int }
2689       { \l__enumext_columns_ \__enumext_level: _int }
2690       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }

```

```

2691         {
2692             columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2693         }
2694     }
2695 }
2696 \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2697 {
2698     \regex_match:nnT { \b columns-sep \b} {#1}
2699     {
2700         \dim_set_eq:cc
2701         { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2702         { l__enumext_columns_sep_ \__enumext_level: _dim }
2703         \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2704         {
2705             columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2706         }
2707     }
2708 }
2709 }

```

(End of definition for __enumext_parse_store_keys:n.)

__enumext_start_store_level:
 __enumext_stop_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.

```

2710 \cs_new_protected:Nn \__enumext_start_store_level:
2711 {
2712     \bool_lazy_all:nT
2713     {
2714         { \bool_if_p:N \__enumext_store_active_bool }
2715         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2716         { \bool_not_p:n { \g__enumext_starred_bool } }
2717     }
2718     {
2719         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2720         {
2721             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2722             \__enumext_store_level_open:
2723         }
2724     }
2725     \bool_lazy_all:nT
2726     {
2727         { \bool_if_p:N \__enumext_store_active_bool }
2728         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2729         { \bool_if_p:N \g__enumext_starred_bool }
2730     }
2731     {
2732         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2733         {
2734             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2735             \__enumext_store_level_open:
2736         }
2737     }
2738 }
2739 \cs_new_protected:Nn \__enumext_stop_store_level:
2740 {
2741     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2742     {
2743         \__enumext_store_level_close:
2744     }
2745 }

```

(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.

```

2746 \cs_new_protected:Nn \__enumext_before_list:
2747 {
2748     \__enumext_vspace_above:
2749     \__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.

```
2750 \__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.

```
2751 \dim_compare:nNnT
2752 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2753 {
2754   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2755   {
2756     \linewidth
2757     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2758     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2759   }
2760 }
```

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

```
2760 \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
2761 \int_gincr:N \g__enumext_minipage_stat_int
2762 \__enumext_mini_addvspace:
2763 \nointerlineskip\noindent
2764 \begin{\__enumext_mini_env*}
2765 { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
2766 }
2767 \__enumext_multicols_start:
2768 }
```

(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.

```
2769 \cs_new_protected:Nn \__enumext_multicols_start:
2770 {
2771   \int_compare:nNnT
2772   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2773   {
2774     \dim_compare:nNnT
2775     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2776     {
2777       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
2778       {
2779         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
2780           + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2781         ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
2782         - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
2783       }
2784     }
2785     \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
2786     \skip_zero:N \multicolsep
2787     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2788     {
2789       \dim_zero:N \columnseprule
2790     }
2791   }
2792 }
```

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.

```
2791 \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2792 {
```

```

2793         \__enumext_multi_addvspace:
2794     }
2795     \raggedcolumns
2796     \begin{multicols}{\int_use:c { \l__enumext_columns_ \__enumext_level: _int } }
2797 }
2798 }

```

(End of definition for __enumext_multicols_start:.)

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

```

2799 \cs_new_protected:Nn \__enumext_multicols_stop:
2800 {
2801     \int_compare:nNtT
2802     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2803     {
2804         \end{multicols}
2805         \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2806         {
2807             \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \__enumext_level: _skip } }
2808         }
2809     }
2810 }

```

(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.

```

2811 \cs_new_protected:Nn \__enumext_after_list:
2812 {
2813     \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2814     {
2815         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2816         {
2817             \msg_warning:nn { enumext } { missing-miniright }
2818             \miniright
2819         }
2820         \int_gzero:N \g__enumext_minipage_stat_int
2821         \end{__enumext_mini_env*}
2822         \par\addvspace { \l__enumext_minipage_after_skip }
2823     }
2824     { \__enumext_multicols_stop: }

```

If the `check-ans` key is active, we set the boolean variable \g__enumext_check_ans_show_bool to true and copy the “store name” to the variable \g__enumext_store_name_tl.

```

2825     \bool_lazy_and:nnT
2826     { \bool_if_p:N \l__enumext_check_ans_bool }
2827     { \bool_not_p:n { \g__enumext_starred_bool } }
2828     {
2829         \bool_gset_true:N \g__enumext_check_ans_show_bool
2830         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2831     }

```

Now apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present, set \l__enumext_standar_bool to false and save the *current value* of the counter for `series`, `resume` and `resume*` keys.

```

2832     \__enumext_after_stop_list:
2833     \__enumext_vspace_below:
2834     \bool_set_false:N \l__enumext_standar_bool
2835     \__enumext_resume_save_counter:
2836 }

```

(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.

```

2837 \__enumext_after_env:nn {enumext}
2838 {

```



```

2839 \int_compare:nNt { \l__enumext_level_int } = { 0 }
2840 {
2841   \bool_if:NT \g__enumext_check_ans_show_bool
2842   {
2843     \__enumext_check_ans_show:
2844   }
2845   \bool_gset_false:N \g__enumext_standar_bool
2846   \bool_gset_false:N \g__enumext_check_ans_show_bool
2847   \tl_gclear:N \g__enumext_store_name_tl
2848 }
2849 }

```

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.

```

2850 \NewDocumentEnvironment{keyans}{ 0{} }
2851 {
2852   \__enumext_keyans_safe_exec:
2853   \__enumext_keyans_parse_keys:n {#1}
2854   \__enumext_before_list_v:
2855   \__enumext_start_list:nn
2856   { \tl_use:N \l__enumext_label_v_tl }
2857   {
2858     \__enumext_list_arg_two_v:
2859     \__enumext_before_keys_exec_v:
2860   }
2861   \__enumext_after_args_exec_v:
2862 }
2863 {
2864   \__enumext_keyans_check_ans:nn { item }{ keyans }
2865   \__enumext_stop_list:
2866   \__enumext_after_list_v:
2867 }

```

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

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

```

2868 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2869 {
2870   \bool_if:NF \l__enumext_store_active_bool
2871   {
2872     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2873   }
2874   \int_incr:N \l__enumext_keyans_level_int
2875   \bool_set_true:N \l__enumext_keyans_env_bool
2876   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2877   \bool_set_false:N \l__enumext_store_active_bool
2878   \int_compare:nNt { \l__enumext_keyans_level_int } > { 1 }
2879   {
2880     \msg_error:nn { enumext } { keyans-nested }
2881   }
2882   \int_compare:nNt { \l__enumext_level_int } > { 1 }
2883   {
2884     \msg_error:nn { enumext } { keyans-wrong-level }
2885   }
2886 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```

2887 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2888 {
2889   \keys_set:nn { enumext / keyans } {#1}
2890 }

```

(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.

```

2891 \cs_new_protected:Nn \__enumext_before_list_v:
2892 {
2893   \__enumext_vspace_above_v:
2894   \__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.

```

2895   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2896   {
2897     \dim_set:Nn \l__enumext_minipage_left_v_dim
2898     {
2899       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2900     }

```

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

```

2901     \bool_set_true:N \l__enumext_minipage_active_v_bool
2902     \int_gincr:N \g__enumext_minipage_stat_int
2903     \__enumext_keyans_mini_addvspace:
2904     \nointerlineskip\noindent
2905     \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2906   }

```

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

```

2907   \__enumext_keyans_multicols_start:
2908 }

```

(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.

```

2909 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2910 {
2911   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2912   {

```

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

```

2913     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2914     {
2915       \dim_set:Nn \l__enumext_columns_sep_v_dim
2916       {
2917         (
2918           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2919         ) / \l__enumext_columns_v_int
2920         - \l__enumext_listoffset_v_dim
2921       }
2922     }
2923     \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).

```

2924     \skip_zero:N \multicolsep
2925     \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.

```

2926     \bool_if:NF \l__enumext_minipage_active_v_bool
2927     {
2928       \__enumext_keyans_multi_addvspace:
2929     }
2930     \raggedcolumns

```

```

2931         \begin{multicols}{\l__enumext_columns_v_int }
2932     }
2933 }

```

(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.

```

2934 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2935 {
2936     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2937     {
2938         \end{multicols}
2939         \bool_if:NF \l__enumext_minipage_active_v_bool
2940         {
2941             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2942         }
2943     }
2944 }

```

(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.

```

2945 \cs_new_protected:Nn \__enumext_after_list_v:
2946 {
2947     \bool_if:NTF \l__enumext_minipage_active_v_bool
2948     {
2949         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2950         {
2951             \msg_warning:nn { enumext } { missing-miniright }
2952             \miniright
2953         }
2954         \int_gzero:N \g__enumext_minipage_stat_int
2955         \end{\__enumext_mini-env*}
2956         \par\addvspace{ \l__enumext_minipage_after_skip }
2957     }
2958     { \__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.

```

2959     \bool_set_false:N \l__enumext_keyans_env_bool
2960     \__enumext_after_stop_list_v:
2961     \__enumext_vspace_below_v:
2962 }

```

(End of definition for `__enumext_after_list_v:`)

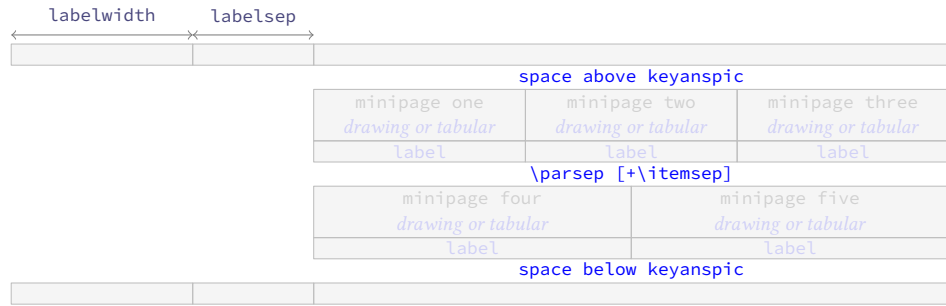
10.34 The environment `keyanspic` and `\anspic`

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.

```
2963 \NewDocumentCommand \anspic { s o +m }
2964 {
```

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

```
2965   \bool_if:NF \l__enumext_store_active_bool
2966   {
2967     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2968   }
2969   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2970   {
2971     \msg_error:nn { enumext } { keyanspic-wrong-level }
2972   }
2973   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2974   {
2975     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2976   }
```

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.

```
2977   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2978   {
2979     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2980   }
2981 }
```

(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.

```
2982 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2983 {
2984   \stepcounter { enumXvi }
2985   #3 \\\
2986   \bool_if:nT { #1 }
2987   {
2988     \__enumext_keyans_addto_prop:n { #2 }
2989     \__enumext_keyans_store_ref:
2990     \__enumext_keyans_addto_seq:n { #2 }
2991     \bool_lazy_or:nnT
2992     { \bool_if_p:N \l__enumext_show_answer_bool }
2993     { \bool_if_p:N \l__enumext_show_position_bool }
2994     {
2995       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2996       \__enumext_keyans_show_left:n { #2 }
2997       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2998     }
2999   }
3000   \tl_use:N \l__enumext_label_font_style_v_tl
3001   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3002 }
```

(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.

```

3003 \NewDocumentEnvironment{keyanspic}{ o }
3004 {
3005   \__enumext_keyans_pic_safe_exec:
3006   \__enumext_start_list:nn
3007   { }
3008   {
3009     \__enumext_keyans_pic_arg_two:
3010   }

```

We apply the “adjusted” vertical spacing above the environment

```

3011   \vspace { \__enumext_keyans_pic_above_skip }
3012 }

```

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.

```

3013 {
3014   \tl_if_novalue:nTF { #1 }
3015   {
3016     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3017   }
3018   { \__enumext_keyans_pic_do:n { #1 } }
3019   \__enumext_stop_list:
3020   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
3021   \setcounter { enumXvi } { 0 }
3022   \vspace { \__enumext_topsep_v_skip }
3023   %\bool_set_false:N \l__enumext_store_active_bool
3024 }

```

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

`__enumext_keyans_pic_safe_exec:`

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

```

3025 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3026 {
3027   \int_incr:N \l__enumext_keyans_pic_level_int
3028   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3029   {
3030     \msg_error:nn { enumext } { keyanspic-nested }
3031   }
3032 }

```

(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`.

```

3033 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3034 {
3035   \dim_compare:nNnT { #1 } < { 0pt }
3036   { \skip_set:Nn #1 { -#1 } }
3037 }

```

(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.

```

3038 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3039 {

```

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.

```

3040   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3041   \__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.

```

3042 \skip_add:Nn \parsep { \itemsep }
3043 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3044 \dim_zero:N \labelwidth
3045 \dim_zero:N \listparindent
3046 \dim_zero:N \labelsep
3047 \skip_zero:N \partopsep
3048 \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.

```

3049 \__enumxt_keyans_pic_skip_abs:N \parsep
3050 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
3051 {
3052   \box_dp:N \strutbox
3053   + \l__enumxt_topsep_v_skip
3054   - \parsep
3055 }
3056 \__enumxt_item_std:w \scan_stop:
3057 }

```

(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`.

```

3058 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
3059 {
3060   \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
3061 }
3062 \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.

```

3063 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
3064 {
3065   \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
3066   \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
3067   \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
3068   \int_step_inline:nnn
3069     { \l__enumxt_keyans_pic_above_int + 1 }
3070     { \l__enumxt_keyans_pic_below_int }
3071     {
3072       \__enumxt_minipage:w [ b ]{ \l__enumxt_keyans_pic_width_dim }
3073       \centering
3074       \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
3075       \__enumxt_endminipage:
3076     }
3077   \par
3078 }

```

(End of definition for `__enumxt_keyans_pic_row:n`)

10.35 The environment `enumxt*`

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.

```

3079 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3080 {
3081   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3082   {
3083     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3084     {
3085       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3086       / \l__enumext_columns_vii_int
3087     }
3088   }
3089   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3090   \dim_set:Nn \l__enumext_item_width_vii_dim
3091   {
3092     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3093     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3094     - \l__enumext_labelsep_vii_dim
3095   }
3096   \dim_zero_new:N \itemwidth
3097 }

```

(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`.

```

3098 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3099 {
3100   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3101   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3102   {
3103     \msg_warning:nnee { enumext } { item-joined }
3104     { \int_use:N \l__enumext_joined_item_vii_int }
3105     { \int_use:N \l__enumext_columns_vii_int }
3106     \int_set:Nn \l__enumext_joined_item_vii_int
3107     {
3108       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3109     }
3110   }
3111   \int_compare:nNnT
3112   { \l__enumext_joined_item_vii_int }
3113   >
3114   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3115   {
3116     \msg_warning:nnee { enumext } { item-joined-columns }
3117     { \int_use:N \l__enumext_joined_item_vii_int }
3118     {
3119       \int_eval:n
3120       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3121     }
3122     \int_set:Nn \l__enumext_joined_item_vii_int
3123     {
3124       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3125     }
3126   }

```

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

```

3127   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3128   {
3129     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3130     \int_decr:N \l__enumext_joined_item_aux_vii_int
3131     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3132     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3133     \dim_set:Nn \l__enumext_joined_width_vii_dim
3134     {
3135       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3136       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3137         + \l__enumext_columns_sep_vii_dim
3138         ) * \l__enumext_joined_item_aux_vii_int

```



```

3139     }
3140     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3141   }
3142   {
3143     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3144     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3145   }
3146 }

```

(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`.

```

3147 \cs_new_protected:Nn \__enumext_start_mini_vii:
3148 {
3149   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3150   {
3151     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3152     {
3153       \linewidth
3154       - \l__enumext_minipage_right_vii_dim
3155       - \l__enumext_minipage_hsep_vii_dim
3156     }
3157     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3158     \dim_gset_eq:NN
3159       \g__enumext_minipage_right_vii_dim
3160       \l__enumext_minipage_right_vii_dim
3161     \__enumext_mini_addvspace_vii:
3162     \nointerlineskip\noindent
3163     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3164   }
3165 }

```

(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*”.

```

3166 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3167 {
3168   \bool_if:NT \l__enumext_minipage_active_vii_bool
3169   {
3170     \end{__enumext_mini_env*}
3171     \hfill
3172     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3173   }
3174 }

```

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*”.

```

3175 \__enumext_after_env:nn {enumext*}
3176 {
3177   \bool_if:NT \g__enumext_minipage_active_vii_bool
3178   {
3179     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3180     \par\addvspace { \g__enumext_minipage_right_skip }
3181     \bool_if:NF \g__enumext_minipage_center_vii_bool
3182     {
3183       \centering
3184     }
3185     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3186     \end{__enumext_mini_env*}
3187     \par\addvspace{ \g__enumext_minipage_after_skip }
3188   }
3189   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3190   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3191   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3192   \dim_gzero:N \g__enumext_minipage_right_vii_dim

```

```

3193     \bool_gset_false:N \g__enumext_starred_bool
3194 }

```

(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.

```

3195 \NewDocumentEnvironment{enumext*}{ o }
3196 {
3197     \__enumext_safe_exec_vii:
3198     \__enumext_parse_keys_vii:n {#1}
3199     \__enumext_before_list_vii:
3200     \__enumext_start_store_level_vii:
3201     \__enumext_start_list:nn { }
3202     {
3203         \__enumext_list_arg_two_vii:
3204         \__enumext_before_keys_exec_vii:
3205     }
3206     \__enumext_starred_columns_set_vii:
3207     \item[] \scan_stop:
3208     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3209     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3210 }
3211 {
3212     \__enumext_stop_item_tmp_vii:
3213     \__enumext_remove_extra_parsep_vii:
3214     \__enumext_stop_list:
3215     \__enumext_stop_store_level_vii:
3216     \__enumext_after_list_vii:
3217 }

```

(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`.

```

3218 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3219 {
3220     \__enumext_current_env_set_bool:
3221     \int_incr:N \l__enumext_level_h_int
3222     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3223     {
3224         \msg_error:nn { enumext } { nested }
3225     }
3226     \bool_set_true:N \l__enumext_starred_bool
3227     \bool_lazy_all:nT
3228     {
3229         { \bool_if_p:N \g__enumext_starred_bool }
3230         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3231         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
3232     }
3233     {
3234         \typeout{[[ON-FIRST-LEVEL-ENUMEXT*-NOT-NESTED]]}
3235         \bool_set_true:N \l__enumext_starred_level_one_bool
3236     }
3237 }

```

(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.

```

3238 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3239 {
3240     \tl_if_novalue:nF {#1}
3241     {
3242         \str_clear:N \l__enumext_series_str
3243         \keys_set:nn { enumext / enumext* } {#1}
3244         \__enumext_parse_series:n {#1}
3245         \bool_if:NT \l__enumext_store_active_bool
3246         {

```

```

3247         \__enumext_parse_store_keys_vii:n {#1}
3248     }
3249 }
3250 }

```

(End of definition for __enumext_parse_store_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:.

```

3251 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3252 {
3253     \bool_if:NF \l__enumext_store_columns_vii_bool
3254     {
3255         \regex_match:nnT { \b columns\b } {#1}
3256         {
3257             \int_set_eq:NN
3258             \l__enumext_store_columns_vii_int
3259             \l__enumext_columns_vii_int
3260             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3261             {
3262                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3263             }
3264         }
3265     }
3266     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3267     {
3268         \regex_match:nnT { \b columns-sep\b } {#1}
3269         {
3270             \dim_set_eq:NN
3271             \l__enumext_store_columns_sep_vii_dim
3272             \l__enumext_columns_sep_vii_dim
3273             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3274             {
3275                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3276             }
3277         }
3278     }
3279 }

```

(End of definition for __enumext_parse_store_keys_vii:n.)

__enumext_before_list_vii:

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

```

3280 \cs_new_protected:Nn \__enumext_before_list_vii:
3281 {
3282     \__enumext_vspace_above_vii:
3283     \__enumext_check_ans_exec: % need by chek-ans
3284     \__enumext_before_args_exec_vii:
3285     \__enumext_start_mini_vii:
3286 }

```

(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.

```

3287 \cs_new_protected:Nn \__enumext_after_list_vii:
3288 {
3289     \__enumext_stop_mini_vii:
3290     \__enumext_after_stop_list_vii:
3291     \__enumext_vspace_below_vii:
3292     \bool_set_false:N \l__enumext_starred_bool
3293     \__enumext_resume_save_counter:
3294 }

```

(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`.

```

3295 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3296 {
3297   \bool_if:NT \l__enumext_store_active_bool
3298   {
3299     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3300     {
3301       \__enumext_store_level_open_vii:
3302     }
3303   }
3304 }
3305 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3306 {
3307   \bool_if:NT \l__enumext_store_active_bool
3308   {
3309     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3310     {
3311       \__enumext_store_level_close_vii:
3312     }
3313   }
3314 }

```

(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`.

```

3315 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3316 {
3317   \__enumext_stop_item_tmp_vii:
3318   \int_incr:N \l__enumext_item_column_pos_vii_int
3319   \int_gincr:N \g__enumext_item_count_all_vii_int
3320   \__enumext_item_peek_args_vii:
3321 }

```

(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).

```

3322 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3323 {
3324   \peek_meaning:NTF (
3325     { \__enumext_joined_item_vii:w }
3326     { \__enumext_joined_item_vii:w (1) }
3327   }

```

(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`.

```

3328 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3329 {
3330   \__enumext_starred_joined_item_vii:n {#1}
3331   \peek_meaning_remove:NTF *
3332     { \__enumext_starred_item_vii:w }
3333     { \__enumext_standard_item_vii:w }
3334 }

```

(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].

```

3335 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3336 {
3337   \bool_set_false:N \l__enumext_item_starred_vii_bool
3338   \peek_meaning:NTF [
3339     {
3340       \bool_set_eq:NN
3341         \l__enumext_wrap_label_vii_bool
3342         \l__enumext_wrap_label_opt_vii_bool
3343       \__enumext_start_item_vii:w
3344     }
3345     {
3346       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3347       \legacy_if_set_true:n { @noitemarg }
3348       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3349     }
3350   }

```

(End of definition for __enumext_standard_item_vii:w.)

__enumext_starred_item_vii:w

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

__enumext_starred_item_vii_aux_i:w

__enumext_starred_item_vii_aux_ii:w

__enumext_starred_item_vii_aux_iii:w

```

3351 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3352 {
3353   \bool_set_true:N \l__enumext_item_starred_vii_bool
3354   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3355   \peek_meaning:NTF [
3356     { \__enumext_starred_item_vii_aux_i:w }
3357     { \__enumext_starred_item_vii_aux_ii:w }
3358   }
3359   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3360   {
3361     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3362     \__enumext_starred_item_vii_aux_ii:w
3363   }
3364   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3365   {
3366     \peek_meaning:NTF [
3367       { \__enumext_starred_item_vii_aux_iii:w }
3368       {
3369         \dim_set_eq:NN
3370           \l__enumext_item_symbol_sep_vii_dim
3371           \l__enumext_labelsep_vii_dim
3372         \legacy_if_set_true:n { @noitemarg }
3373         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3374       }
3375     }
3376   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3377   {
3378     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3379     \legacy_if_set_true:n { @noitemarg }
3380     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3381   }

```

(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.

```

3382 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3383 {
3384   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3385   \legacy_if:nT { @noitemarg }
3386   {
3387     \legacy_if_set_false:n { @noitemarg }
3388     \legacy_if:nT { @nmbrlist }
3389     {
3390       \bool_if:NT \__enumext_hyperref_bool
3391       {
3392         \legacy_if_set_true:n { @hyper@item }
3393       }
3394       \refstepcounter{enumXvii}
3395       \bool_if:NT \__enumext_check_ans_bool
3396       {
3397         \int_gincr:N \g__enumext_count_item_number_int
3398       }
3399     }
3400   }

```

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 `__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.

```

3401   \group_begin:
3402   \lrbox{ \__enumext_item_text_vii_box }
3403   \bool_if:NF \__enumext_footnotes_key_bool
3404   {
3405     \__enumext_renew_footnote:
3406   }
3407   \bool_if:NT \__enumext_item_starred_vii_bool
3408   {
3409     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3410     {
3411       \tl_gset_eq:NN
3412       \g__enumext_item_symbol_aux_vii_tl \__enumext_item_symbol_vii_tl
3413     }
3414     \mode_leave_vertical:
3415     \skip_horizontal:n { -\__enumext_item_symbol_sep_vii_dim }
3416     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3417     \skip_horizontal:N \__enumext_item_symbol_sep_vii_dim
3418     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3419   }
3420   \group_begin:
3421   \tl_use:N \__enumext_label_font_style_vii_tl
3422   \bool_if:NTF \__enumext_wrap_label_vii_bool
3423   {
3424     \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]
3425     { \__enumext_wrapper_label_vii:n {#1} }
3426   }
3427   {
3428     \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3429   }
3430   \group_end:
3431   \skip_horizontal:N \__enumext_labelsep_vii_dim
3432   \tl_use:N \__enumext_after_list_args_vii_tl
3433   \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3434   \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3435   \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3436   \tl_use:N \__enumext_fake_item_indent_vii_tl
3437 }

```

(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.

```

3438 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3439 {
3440     \__enumext_endminipage:
3441     \endlrbox
3442     \group_end:
3443     \box_set_wd:Nn \l__enumext_item_text_vii_box
3444     {
3445         \l__enumext_joined_width_vii_dim
3446         + \l__enumext_labelwidth_vii_dim
3447         + \l__enumext_labelsep_vii_dim
3448     }
3449     \int_set:Nn \hbadness { 10000 }
3450     \box_use:N \l__enumext_item_text_vii_box
3451     \bool_if:NF \l__enumext_footnotes_key_bool
3452     {
3453         \__enumext_print_footnote:
3454     }
3455     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3456     {
3457         \par\noindent
3458         \int_zero:N \l__enumext_item_column_pos_vii_int
3459     }
3460     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3461 }

```

(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.

```

3462 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3463 {
3464     \int_compare:nNnT
3465     {
3466         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3467     }
3468     =
3469     { \c_zero_int }
3470     {
3471         \par
3472         \vspace{ -\l__enumext_itemsep_vii_skip }
3473         \int_gzero:N \g__enumext_item_count_all_vii_int
3474     }
3475 }

```

(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`.

```

3476 \__enumext_after_env:nn {enumext*}
3477 {
3478     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3479     {
3480         \bool_if:NT \g__enumext_check_ans_show_h_bool
3481         {
3482             \__enumext_check_ans_show:
3483         }
3484         \bool_gset_false:N \g__enumext_starred_bool
3485         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3486         \tl_gclear:N \g__enumext_store_name_tl
3487     }
3488 }

```

10.36 The environment `keyans*`

10.36.1 Functions for item box width

`__enumext_starred_columns_set_viii:`

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

```

3489 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3490 {
3491     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }

```

```

3492     {
3493         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3494         {
3495             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3496             / \l__enumext_columns_viii_int
3497         }
3498     }
3499     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3500     \dim_set:Nn \l__enumext_item_width_viii_dim
3501     {
3502         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3503         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3504         - \l__enumext_labelsep_viii_dim
3505     }
3506     \dim_zero_new:N \itemwidth
3507 }

```

(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.

```

3508 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3509 {
3510     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3511     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3512     {
3513         \msg_warning:nnee { enumext } { item-joined }
3514         { \int_use:N \l__enumext_joined_item_viii_int }
3515         { \int_use:N \l__enumext_columns_viii_int }
3516         \int_set:Nn \l__enumext_joined_item_viii_int
3517         {
3518             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3519         }
3520     }
3521     \int_compare:nNnT
3522     { \l__enumext_joined_item_viii_int }
3523     >
3524     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3525     {
3526         \msg_warning:nnee { enumext } { item-joined-columns }
3527         { \int_use:N \l__enumext_joined_item_viii_int }
3528         {
3529             \int_eval:n
3530             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3531         }
3532         \int_set:Nn \l__enumext_joined_item_viii_int
3533         {
3534             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3535         }
3536     }
3537     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3538     {
3539         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3540         \int_decr:N \l__enumext_joined_item_aux_viii_int
3541         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3542         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3543         \dim_set:Nn \l__enumext_joined_width_viii_dim
3544         {
3545             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3546             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3547                 + \l__enumext_columns_sep_viii_dim
3548             ) * \l__enumext_joined_item_aux_viii_int
3549         }
3550         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3551     }
3552     {
3553         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3554         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3555     }
3556 }

```


(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.

```

3557 \cs_new_protected:Nn \__enumext_start_mini_viii:
3558 {
3559   \dim_compare:nNt { \__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3560   {
3561     \dim_set:Nn \__enumext_minipage_left_viii_dim
3562     {
3563       \linewidth
3564       - \__enumext_minipage_right_viii_dim
3565       - \__enumext_minipage_hsep_viii_dim
3566     }
3567     \bool_set_true:N \__enumext_minipage_active_viii_bool
3568     \dim_gset_eq:NN
3569       \g__enumext_minipage_right_viii_dim
3570       \l__enumext_minipage_right_viii_dim
3571     \__enumext_mini_addvspace_viii:
3572     \nointerlineskip\noindent
3573     \begin{__enumext_mini_env*}{ \__enumext_minipage_left_viii_dim }
3574   }
3575 }
3576 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3577 {
3578   \bool_if:NT \__enumext_minipage_active_viii_bool
3579   {
3580     \end{__enumext_mini_env*}
3581     \hfill
3582     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3583   }
3584 }
3585 \__enumext_after_env:nn {keyans*}
3586 {
3587   \bool_if:NT \g__enumext_minipage_active_viii_bool
3588   {
3589     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3590     \par\addvspace { \g__enumext_minipage_right_skip }
3591     \bool_if:NF \g__enumext_minipage_center_viii_bool
3592     {
3593       \centering
3594     }
3595     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3596     \end{__enumext_mini_env*}
3597     \par\addvspace{ \g__enumext_minipage_after_skip }
3598   }
3599   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3600   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3601   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3602   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3603 }

```

(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.

```

3604 \NewDocumentEnvironment{keyans*}{ o }
3605 {
3606   \__enumext_safe_exec_viii:
3607   \__enumext_parse_keys_viii:n {#1}
3608   \__enumext_before_list_viii:
3609   \__enumext_start_list:nn { }
3610   {
3611     \__enumext_list_arg_two_viii:
3612     \__enumext_before_keys_exec_viii:
3613   }
3614   \__enumext_starred_columns_set_viii:
3615   \item[] \scan_stop:
3616   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3617   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3618 }

```

```

3619 {
3620   \__enumext_stop_item_tmp_viii:
3621   \__enumext_remove_extra_parsep_viii:
3622   \__enumext_keyans_check_ans:nn { item }{ keyans* }
3623   \__enumext_stop_list:
3624   \__enumext_after_list_viii:
3625 }

```

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

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

```

3626 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3627 {
3628   \int_incr:N \l__enumext_keyans_level_h_int
3629   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3630   {
3631     \msg_error:nn { enumext } { nested }
3632   }
3633   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3634   \bool_set_false:N \l__enumext_store_active_bool
3635   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3636   {
3637     \msg_error:nn { enumext } { keyans-wrong-level }
3638   }
3639 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

3640 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3641 {
3642   \tl_if_novalue:nF {#1}
3643   {
3644     \keys_set:nn { enumext / keyans* } {#1}
3645   }
3646 }

```

(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`.

```

3647 \cs_new_protected:Nn \__enumext_before_list_viii:
3648 {
3649   \__enumext_vspace_above_viii:
3650   \__enumext_before_args_exec_viii:
3651   \__enumext_start_mini_viii:
3652 }

```

(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.

```

3653 \cs_new_protected:Nn \__enumext_after_list_viii:
3654 {
3655   \__enumext_stop_mini_viii:
3656   \__enumext_after_stop_list_viii:
3657   \__enumext_vspace_below_viii:
3658 }

```

(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`.

```
3659 \cs_new_protected_nopar:Nn \_enumext_start_item_tmp_viii:
3660 {
3661   \_enumext_stop_item_tmp_viii:
3662   \int_incr:N \l\_enumext_item_column_pos_viii_int
3663   \int_gincr:N \g\_enumext_item_count_all_viii_int
3664   \_enumext_item_peek_args_viii:
3665 }
```

(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).

```
3666 \cs_new_protected:Nn \_enumext_item_peek_args_viii:
3667 {
3668   \peek_meaning:NTF (
3669     { \_enumext_joined_item_viii:w }
3670     { \_enumext_joined_item_viii:w (1) }
3671 }
```

(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`.

```
3672 \cs_new_protected:Npn \_enumext_joined_item_viii:w (#1)
3673 {
3674   \_enumext_starred_joined_item_viii:n {#1}
3675   \peek_meaning_remove:NTF *
3676     { \_enumext_starred_item_viii:w }
3677     { \_enumext_standard_item_viii:w }
3678 }
```

(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]`.

```
3679 \cs_new_protected:Npn \_enumext_standard_item_viii:w
3680 {
3681   \bool_set_false:N \l\_enumext_item_starred_viii_bool
3682   \peek_meaning:NTF [
3683     {
3684       \bool_set_eq:NN
3685         \l\_enumext_wrap_label_viii_bool
3686         \l\_enumext_wrap_label_opt_viii_bool
3687       \_enumext_start_item_viii:w
3688     }
3689     {
3690       \bool_set_true:N \l\_enumext_wrap_label_viii_bool
```

```

3691         \legacy_if_set_true:n { @noitemarg }
3692         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3693     }
3694 }

```

(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].

```

3695 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3696 {
3697     \bool_set_true:N \__enumext_item_starred_viii_bool
3698     \bool_set_true:N \__enumext_wrap_label_viii_bool
3699     \peek_meaning:NTF [
3700         { \__enumext_starred_item_viii_aux_i:w }
3701         { \__enumext_starred_item_viii_aux_ii:w }
3702     }

```

The optional argument will be captured in the variables __enumext_keyans_tmpa_tl and __enumext_keyans_tmpb_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.

```

3703 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3704 {
3705     \tl_clear:N \__enumext_store_keyans_label_tl
3706     \tl_if_no_value:nF { #1 }
3707     {
3708         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3709         {
3710             \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
3711             \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
3712         }
3713         \tl_set:Ne \__enumext_keyans_item_opt_tl { #1 }
3714     }
3715     \__enumext_starred_item_viii_aux_ii:w
3716 }
3717 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3718 {
3719     \legacy_if_set_true:n { @noitemarg }
3720     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3721 }

```

(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.

```

3722 \cs_new_protected:Nn \__enumext_starred_item_exec:
3723 {
3724     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3725     \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3726     \__enumext_keyans_store_ref:
3727     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3728     \__enumext_keyans_addto_seq_link:
3729     \bool_if:NT \__enumext_show_answer_bool
3730     {
3731         \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3732     }
3733     \bool_if:NT \__enumext_show_position_bool
3734     {
3735         \tl_set:Ne \__enumext_mark_answer_sym_tl
3736         {
3737             \group_begin:
3738             \exp_not:N \normalfont
3739             \exp_not:N \footnotesize [ \int_eval:n
3740                 {
3741                     \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3742                 }
3743             ]
3744             \group_end:

```

```

3745     }
3746     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3747   }
3748 }

```

(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.

```

3749 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3750 {
3751   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3752   \legacy_if:nT { @noitemarg }
3753   {
3754     \legacy_if_set_false:n { @noitemarg }
3755     \legacy_if:nT { @nmbrrlist }
3756     {
3757       \bool_if:NT \l__enumext_hyperref_bool
3758       {
3759         \legacy_if_set_true:n { @hyper@item }
3760       }
3761       \refstepcounter{enumXviii}
3762     }
3763   }

```

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

```

3764   \group_begin:
3765   \lrbox{ \l__enumext_item_text_viii_box }
3766   \bool_if:NF \l__enumext_footnotes_key_bool
3767   {
3768     \__enumext_renew_footnote:
3769   }
3770   \bool_if:NT \l__enumext_item_starred_viii_bool
3771   {
3772     \__enumext_starred_item_exec:
3773   }
3774   \group_begin:
3775   \tl_use:N \l__enumext_label_font_style_viii_tl
3776   \bool_if:NTF \l__enumext_wrap_label_viii_bool
3777   {
3778     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3779     { \__enumext_wrapper_label_viii:n {#1} }
3780   }
3781   {
3782     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
3783   }
3784   \group_end:
3785   \skip_horizontal:N \l__enumext_labelsep_viii_dim
3786   \tl_use:N \l__enumext_after_list_args_viii_tl
3787   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3788   \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3789   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3790   \bool_if:NT \l__enumext_item_starred_viii_bool
3791   {
3792     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3793     \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3794   }
3795   {
3796     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3797   }
3798 }

```

(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.

```

3799 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3800 {
3801   \__enumext_endminipage:
3802   \endlrbox

```

```

3803 \group_end:
3804 \box_set_wd:Nn \l__enumext_item_text_viii_box
3805 {
3806   \l__enumext_joined_width_viii_dim
3807   + \l__enumext_labelwidth_viii_dim
3808   + \l__enumext_labelsep_viii_dim
3809 }
3810 \int_set:Nn \hbadness { 10000 }
3811 \box_use:N \l__enumext_item_text_viii_box
3812 \bool_if:NF \l__enumext_footnotes_key_bool
3813 {
3814   \__enumext_print_footnote:
3815 }
3816 \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3817 {
3818   \par\noindent
3819   \int_zero:N \l__enumext_item_column_pos_viii_int
3820 }
3821 { \hspace{ \l__enumext_columns_sep_viii_dim } }
3822 }

```

(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.

```

3823 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3824 {
3825   \int_compare:nNnT
3826   {
3827     \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3828   }
3829   =
3830   { \c_zero_int }
3831   {
3832     \par
3833     \vspace{ -\l__enumext_itemsep_viii_skip }
3834     \int_gzero:N \g__enumext_item_count_all_viii_int
3835   }
3836 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

10.37 The command \getkeyans

\getkeyans

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

```

3837 \NewDocumentCommand \getkeyans { m }
3838 {
3839   \exp_args:Ne \__enumext_getkeyans_aux:n
3840   { \tl_to_str:e { \text_expand:n {#1} } }
3841 }

```

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

__enumext_getkeyans_aux:n

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

```

3842 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3843 {
3844   \str_if_in:nnTF {#1} { : }
3845   {
3846     \use:e
3847     {
3848       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3849       { {##1} {##2} }
3850     }
3851     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3852   }
3853   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3854 }

```

(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)*.

```

3855 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3856 {
3857   \prop_if_exist:cF { g__enumext_#1_prop }
3858   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3859   \group_begin:
3860     \prop_item:cn { g__enumext_#1_prop }{#2}
3861   \group_end:
3862 }

```

(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`.

```

3863 \keys_define:nn { keyanskey / print }
3864 {
3865   level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3866                       {
3867                         \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3868                       },
3869   level-1 .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3870   level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3871                       {
3872                         \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3873                       },
3874   level-2 .initial:n   = { nosep, label=(\alph*. , first=\small, font=\small },
3875   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3876                       {
3877                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3878                       },
3879   level-3 .initial:n   = { nosep, label=\roman*. , first=\small, font=\small },
3880   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3881                       {
3882                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3883                       },
3884   level-4 .initial:n   = { nosep, label=\Alph*. , first=\small, font=\small },
3885   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3886                       {
3887                         \setenumext[enumext*] {#1} %\setenumext[print,*] {#1}
3888                       },
3889   level-* .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3890 }

```

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

```

3891 \NewDocumentCommand \printkeyans { s O{ } m }
3892 {
3893   \group_begin:
3894     \tl_use:N \l__enumext_print_keyans_i_tl
3895     \tl_use:N \l__enumext_print_keyans_ii_tl
3896     \tl_use:N \l__enumext_print_keyans_iii_tl
3897     \tl_use:N \l__enumext_print_keyans_iv_tl
3898     \tl_use:N \l__enumext_print_keyans_vii_tl
3899     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3900   \group_end:
3901 }

```

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

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

```

#1:  starred
#2:  key-val
#3:  seq-name

```

```

3902 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3903 {
3904   \seq_if_exist:cTF { g__enumext_#3_seq }
3905   {
3906     \seq_if_empty:cF { g__enumext_#3_seq }
3907     {
3908       %%\seq_show:c { g__enumext_#3_seq }
3909       \bool_if:nTF {#1}
3910       {
3911         \begin{enumext*}[#2]
3912           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3913           \end{enumext*}
3914         }
3915       {
3916         \begin{enumext}[#2]
3917           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3918           \end{enumext}
3919         }
3920       }
3921     }
3922     {
3923       \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3924     }
3925   }

```

(End of definition for __enumext_printkeyans:nnn.)

10.39 The command \setenumext

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

```

3926 \keys_define:nn { enumext / meta-families }
3927 {
3928   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3929   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3930   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3931   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3932   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3933   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3934   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3935   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3936   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3937   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3938   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3939   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3940   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3941 }

```

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

```

3942 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3943 {
3944   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3945   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3946 }

```

\setenumext Now we define the user command \setenumext.

```

3947 \NewDocumentCommand \setenumext { o +m }
3948 {
3949   \tl_if_novalue:nTF {#1}
3950   {
3951     \seq_map_inline:Nn \c__enumext_all_families_seq
3952     }
3953   {
3954     \seq_clear:N \l__enumext_setkey_tmpa_seq
3955     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3956     \int_set:Nn \l__enumext_setkey_tmpa_int
3957     {
3958       \seq_count:N \l__enumext_setkey_tmpb_seq
3959     }
3960     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3961     {
3962       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3963       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n

```



```

3964         \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3965         {
3966             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3967         }
3968     }
3969     {
3970         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3971     }
3972     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3973     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3974     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3975 }
3976 {
3977     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3978 }
3979 }

```

(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.

```

3980 \cs_new_protected:Npn \__enumext_set_parse:n #1
3981 {
3982     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3983     \int_step_inline:nnn { 0 } { 4 } % <- max level
3984     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3985     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3986     {
3987         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3988         { \tl_trim_spaces:n {#1} }
3989     }
3990     { \__enumext_set_error:nn {#1} { } }
3991 }
3992 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3993 { \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.

```

3994 \msg_new:nnn { enumext } { package-load }
3995 {
3996     The ~ '#1' ~ package ~ is ~ already ~ loaded.
3997 }
3998 \msg_new:nnn { enumext } { package-not-load }
3999 {
4000     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4001 }
4002 \msg_new:nnn { enumext } { package-load-foot }
4003 {
4004     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4005 }

```

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

```

4006 \msg_new:nnn { enumext } { counters }
4007 {
4008     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
4009     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4010 }

```

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

```

4011 \msg_new:nnn { enumext } { invalid-key }
4012 {
4013     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4014 }
4015 \msg_new:nnn { enumext } { unknown-key-family }
4016 {
4017     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4018 }

```

Messages used in length calculation.

```

4019 \msg_new:nnn { enumext } { width-negative }
4020 {
4021   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4022   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4023 }
4024 \msg_new:nnn { enumext } { width-zero }
4025 {
4026   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4027   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4028 }

```

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

```

4029 \msg_new:nnn { enumext } { list-lengths }
4030 {
4031   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4032   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4033   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4034   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4035   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4036   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4037   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4038   \__enumext_show_length:nnn { skip } { topsep } {#1}
4039   \__enumext_show_length:nnn { skip } { parsep } {#1}
4040   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4041   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4042   ****
4043 }

```

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

```

4044 \msg_new:nnn { enumext } { list-lengths-not-nested }
4045 {
4046   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4047   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4048   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4049   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4050   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4051   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4052   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4053   \__enumext_show_length:nnn { skip } { topsep } {#1}
4054   \__enumext_show_length:nnn { skip } { parsep } {#1}
4055   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4056   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4057   ****
4058 }

```

Messages used by `save-ans` key.

```

4059 \msg_new:nnn { enumext } { save-ans-empty }
4060 {
4061   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1' ~ on ~ line ~ #2.
4062 }
4063 \msg_new:nnn { enumext } { save-ans-ok }
4064 {
4065   Set ~ 'save-ans=#2' ~ in ~ '#1' ~ on ~ line ~ #3.
4066 }

```

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

```

4067 \msg_new:nnn { enumext } { items-same-answer }
4068 {
4069   *****~Checking~answers~on~'#1'~OK~*****\
4070   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \
4071   *****
4072   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4073 }
4074 \msg_new:nnn { enumext } { item-different-answer }
4075 {
4076   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4077   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4078 }

```

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

```

4079 \msg_new:nnn { enumext } { missing-starred }
4080 {

```

```

4081     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4082 }

```

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

```

4083 \msg_new:nnn { enumext } { list-too-deep }
4084 {
4085     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4086     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4087 }

```

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

```

4088 \msg_new:nnn { enumext } { anskey-wrong-place }
4089 {
4090     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4091     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4092 }
4093 \msg_new:nnn { enumext } { anspic-wrong-place }
4094 {
4095     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4096     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4097 }
4098 \msg_new:nnn { enumext } { command-wrong-place }
4099 {
4100     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4101     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4102 }

```

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

```

4103 \msg_new:nnn { enumext } { keyans-nested }
4104 {
4105     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4106 }
4107 \msg_new:nnn { enumext } { keyans-wrong-level }
4108 {
4109     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4110     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4111 }
4112 \msg_new:nnn { enumext } { wrong-place }
4113 {
4114     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4115     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4116 }
4117 \msg_new:nnn { enumext } { keyanspic-nested }
4118 {
4119     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4120 }
4121 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4122 {
4123     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4124     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4125 }

```

Messages used by `\getkeyans` command.

```

4126 \msg_new:nnn { enumext } { undefined-storage-anskey }
4127 {
4128     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4129 }

```

Messages used by `\miniright` command.

```

4130 \msg_new:nnn { enumext } { missing-miniright }
4131 {
4132     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4133     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4134 }
4135 \msg_new:nnn { enumext } { wrong-miniright-place }
4136 {
4137     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4138     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4139 }
4140 \msg_new:nnn { enumext } { wrong-miniright-use }
4141 {
4142     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4143     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4144 }

```

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

```
4145 \msg_new:nnn { enumext } { nested }
4146 {
4147   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4148 }
4149 \msg_new:nnn { enumext } { item-joined }
4150 {
4151   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4152 }
4153 \msg_new:nnn { enumext } { item-joined-columns }
4154 {
4155   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4156 }
```

10.41 Finish package

Finish package implementation.

```
4157 \file_input_stop:
4158 </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

*

.....

206

\+

.....

198

\-

.....

198

\\

214, 2985, 4008, 4021, 4026, 4031, 4046, 4069, 4070, 4085, 4090, 4095, 4100, 4109, 4114, 4123, 4132, 4137, 4142

A

above

.....

1266

above*

.....

1266

\addvspace

913, 941, 1064, 1143, 1206, 1212, 1240, 1257, 2807, 2822, 2941, 2956, 3180, 3187, 3590, 3597

after

.....

751

align

.....

387

\Alph

.....

31, 35

\Alph

339, 448, 493, 547, 3884

\alph

.....

31, 35

\alph

.....

340, 446, 3874

\anskey

.....

11, 63, 1931

\anspic

.....

13, 84, 85, 2963

\arabic

.....

27, 31

\arabic

338, 445, 492, 3869, 3889

B

\b

.....

2685, 2698, 3255, 3268

\baselineskip

.....

43

\baselineskip

1891, 1899

before

.....

751

before*

.....

751

below

.....

1266

below*

.....

1266

bool commands:

\bool_gset_false:N

2845, 2846, 3189, 3193, 3484, 3485, 3599

\bool_gset_true:N

234, 245, 855, 2829, 3172, 3190, 3582, 3600

\bool_if:NTF

279, 291, 308, 1288, 1302, 1315, 1326, 1337, 1348, 1359, 1370, 1423, 1440, 1445, 1453, 1480, 1518, 1523, 1530, 1534, 1556, 1561, 1569, 1576, 1607, 1615, 1638, 1643, 1650, 1657, 1729, 1739, 1853, 1877, 1884, 1912, 1943, 1956, 1958, 1969, 1989, 2114, 2125, 2129, 2168, 2183, 2256, 2275, 2279, 2392, 2422, 2495, 2511, 2573, 2583, 2613, 2618, 2675, 2683, 2696, 2741, 2791, 2805, 2813, 2841, 2870, 2926, 2939, 2947, 2965, 3168, 3177, 3181, 3245, 3253, 3266, 3297, 3307, 3390, 3395, 3403, 3407, 3422, 3451, 3480, 3578, 3587, 3591, 3729, 3733, 3757, 3766, 3770, 3776, 3790, 3812

\bool_if:nTF

1241, 1258, 1997, 2433, 2467, 2531, 2986, 3909

\bool_if_p:N

1587, 1588, 1596, 1597, 1708, 1980, 2023, 2024, 2048, 2057, 2058, 2070, 2086, 2242, 2243, 2289, 2290, 2652, 2714, 2727, 2729, 2826, 2992, 2993, 3229

\bool_lazy_all:nTF

1706, 2046, 2055, 2068, 2084, 2650, 2712, 2725, 3227

\bool_lazy_and:nnTF

230, 241, 1586, 1595, 1979, 2022, 2241, 2825

\bool_lazy_or:nnTF

2288, 2991

\bool_new:N

25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88, 93, 94, 97, 117, 119, 121, 124, 125, 134, 135, 136, 137, 146, 147, 161, 172, 174

\bool_not_p:n

231, 242, 1981, 2073, 2088, 2715, 2716, 2728, 2827

\bool_set_eq:NN

2400, 2448, 3340, 3684

\bool_set_false:N

288, 1690, 1691, 2834, 2877, 2959, 3023, 3040, 3292, 3337, 3634, 3681

\bool_set_true:N

270, 274, 380, 679, 1272, 1277, 1543, 1668, 1669, 1815, 1822, 2396, 2426, 2444, 2456, 2649, 2658, 2721, 2734, 2760, 2875, 2901, 3157, 3226, 3235, 3346, 3353, 3354, 3567, 3690, 3697, 3698

box commands:

\box_dp:N

960, 964, 968, 979, 983, 994, 1003, 1009, 1019, 1032, 1038, 1044, 1075, 1076, 1077, 1080, 1090, 1094, 1103, 1110, 1115, 1123, 1152, 1153, 1156, 1163, 1176, 1184, 1190, 1198, 3052

\box_new:N

58, 167

\box_set_wd:Nn

3443, 3804

\box_use:N

3450, 3811

\box_wd:N

346

C

\c

.....

206, 207, 579, 581, 593, 595

\cB

.....

207

\cE

.....

207

\centering

.....

1243, 1260, 3073, 3183, 3593

check-ans

.....

1683

Document class:

article

.....

36

clist commands:

\clist_const:Nn

.....

179

\clist_map_function:nN

.....

3060

\clist_map_inline:Nn

386, 621, 684, 750, 765, 846, 1282

\clist_map_inline:nn

36, 47, 66, 72, 84, 96, 123, 155, 178, 222, 411, 428, 689, 861, 1388, 1632, 1696, 1792, 1810, 1831, 2043, 2177, 2350, 2560, 2563, 2590, 2600, 2603, 2623

\columnbreak

.....

64

\columnbreak

.....

1983

columns

.....

830

columns*

.....

1811

columns-sep

.....

830

columns-sep*

.....

1811

\columnsep

.....

80, 83

\columnsep

.....

2785, 2923

\columnseprule

.....

80, 83

\columnseprule

.....

2789, 2925

Commands provide by enumext:

\anskey

24, 25, 57, 58, 61, 63, 65-67, 69, 79, 92, 102, 103, 107

\anspic*

24, 67-69, 85-87, 102, 103

\anspic

61, 84-87, 107

\getkeyans

61, 102, 107

\item*

24, 61, 67-69, 72, 73, 93, 100, 102, 103

\itemwidth

88, 95, 96

\item

71, 73, 88, 92, 93, 96, 99

\miniright

24, 41, 48, 49, 80, 81, 83, 84, 107

\printkeyans

25, 61, 103

\setenumext

25, 104, 105

Counters defined by enumext:

enumXiii

.....

23, 30

©2024 by Pablo González L

109 / 120

enumXii 23, 30
 enumXiv 23, 30
 enumXi 23, 30
 enumXviii 23, 30
 enumXvii 23, 30, 93
 enumXvi 23, 30
 enumXv 23, 30

cs commands:

\cs_generate_variant:Nn 348, 364, 585, 601, 1836,
 1845, 1850, 1930, 2550, 3062
 \cs_if_exist:NTF 318
 \cs_new:Nn 192
 \cs_new:Npn 210, 1389, 1398, 1407
 \cs_new_eq:NN 254, 255, 256, 260, 261, 293, 294, 297,
 298
 \cs_new_protected:Nn . 202, 224, 265, 469, 519, 565,
 766, 770, 774, 778, 782, 786, 790, 794, 798, 802, 806,
 810, 814, 818, 822, 826, 862, 874, 898, 915, 926, 950,
 1025, 1049, 1066, 1128, 1145, 1167, 1202, 1208, 1283,
 1297, 1311, 1322, 1333, 1344, 1355, 1366, 1451, 1554,
 1567, 1584, 1605, 1666, 1701, 1737, 1744, 1851, 1875,
 1882, 1910, 1917, 2034, 2166, 2181, 2209, 2239, 2284,
 2296, 2303, 2355, 2359, 2378, 2429, 2463, 2479, 2489,
 2505, 2643, 2710, 2739, 2746, 2769, 2799, 2811, 2868,
 2891, 2909, 2934, 2945, 2982, 3025, 3038, 3058, 3063,
 3079, 3147, 3166, 3218, 3280, 3287, 3295, 3305, 3322,
 3462, 3489, 3557, 3576, 3626, 3647, 3653, 3666, 3722,
 3823
 \cs_new_protected:Npn 184, 188, 301, 316, 333, 343,
 349, 449, 494, 552, 572, 586, 1230, 1249, 1419, 1438,
 1508, 1541, 1633, 1837, 1846, 1966, 2111, 2123, 2145,
 2219, 2261, 2269, 2388, 2406, 2440, 2452, 2519, 2553,
 2593, 2661, 2681, 2887, 3033, 3098, 3238, 3251, 3328,
 3335, 3351, 3359, 3364, 3376, 3508, 3640, 3672, 3679,
 3695, 3703, 3717, 3842, 3855, 3902, 3980, 3992
 \cs_new_protected_nopar:Nn . . . 3315, 3438, 3659,
 3799
 \cs_new_protected_nopar:Npn 3382, 3749
 \cs_set:Nn 2116
 \cs_set:Npn 2044, 2082, 3848
 \cs_set_eq:NN . . 3208, 3209, 3384, 3616, 3617, 3751
 \cs_set_protected:Nn 216, 690, 706, 718, 730
 \cs_set_protected:Npn . 32, 41, 59, 67, 79, 85, 113,
 151, 159, 218, 365, 387, 416, 429, 476, 602, 622, 666,
 685, 742, 751, 830, 847, 1266, 1377, 1624, 1683, 1757,
 1793, 1811, 2036, 2170, 2339, 2551, 2591
 \cs_to_str:N 335, 358

D

\d 198
 \DeclareDocumentEnvironment 943

dim commands:

\dim_abs:n 2524, 2529
 \dim_add:Nn 3043
 \dim_compare:nNnTF . 692, 708, 720, 732, 1232, 1251,
 2521, 2526, 2532, 2538, 2540, 2542, 2751, 2774, 2895,
 2913, 3035, 3081, 3149, 3491, 3559
 \dim_compare:nTF 2007
 \dim_gset_eq:NN 3158, 3568
 \dim_gzero:N 3192, 3602
 \dim_new:N . 55, 62, 63, 64, 81, 107, 120, 130, 168, 169,
 175
 \dim_set:Nn . . 346, 680, 1823, 2420, 2524, 2529, 2531,
 2534, 2535, 2539, 2541, 2544, 2545, 2547, 2754, 2777,
 2897, 2915, 3065, 3083, 3090, 3133, 3151, 3378, 3493,
 3500, 3543, 3561

\dim_set_eq:NN 436, 483, 540, 544, 2415, 2562, 2602,
 2700, 2785, 2923, 3140, 3143, 3144, 3270, 3369, 3550,
 3553, 3554
 \dim_use:N 693, 701, 1233, 1239, 1920, 1923, 1928, 2484,
 2486, 2752, 2757, 2758, 2765, 2775, 2779, 2780, 2782
 \dim_zero:N 2789, 2925, 3044, 3045, 3046
 \dim_zero_new:N 3096, 3506
 \c_zero_dim 695, 709, 721, 733, 1233, 1251, 2009, 2521,
 2526, 2532, 2539, 2752, 2775, 2895, 2913, 3081, 3149,
 3491, 3559

E

\end . . 1236, 1254, 1879, 1914, 2804, 2821, 2938, 2955, 3170,
 3186, 3580, 3596, 3913, 3918
 \endlist 28
 \endlist 255
 \endlrbox 3441, 3802
 \endminipage 28
 \endminipage 261
 enumext 5, 2624
 enumext internal commands:

\l__enumext_l_ref_the_count_tl 33
 \g__enumext_l__enumext_store_name_tl
 _prop 69
 \l__enumext__resume_name_tl 53
 __enumext_add_pre_parsep: . . . 42, 872, 874, 874
 __enumext_after_args_exec: . 39, 766, 778, 2636
 __enumext_after_args_exec_v: . 39, 40, 782, 794,
 2861
 __enumext_after_args_exec_vii: . . . 798, 822
 __enumext_after_args_exec_viii: 826
 __enumext_after_env:nn . . 81, 95, 188, 188, 2837,
 3175, 3476, 3585
 __enumext_after_hyperref: . . . 29, 263, 265, 265
 __enumext_after_list: 81, 91, 98, 2641, 2811, 2811
 \l__enumext_after_list_args_v_tl 796
 \l__enumext_after_list_args_vii_tl 824, 3432
 \l__enumext_after_list_args_viii_tl 828, 3786
 __enumext_after_list_v: . . 84, 2866, 2945, 2945
 __enumext_after_list_vii: . . 3216, 3287, 3287
 __enumext_after_list_viii: . . 3624, 3653, 3653
 __enumext_after_star_env:nn 89
 __enumext_after_stop_list: . . 39, 40, 766, 774,
 2832
 __enumext_after_stop_list_v: 39, 782, 790, 2960
 \l__enumext_after_stop_list_v_tl 792
 __enumext_after_stop_list_vii: 798, 814, 3290
 \l__enumext_after_stop_list_vii_tl . . . 816
 __enumext_after_stop_list_viii: . 818, 3656
 \l__enumext_after_stop_list_viii_tl . . . 820
 \l__enumext_align_label_vii_str . . 3424, 3428
 \l__enumext_align_label_viii_str . 3778, 3782
 \l__enumext_align_label_X_str 159
 \c__enumext_all_envs_clist . . 179, 386, 621, 684,
 750, 765, 846, 1282
 \c__enumext_all_families_seq . 104, 3942, 3951,
 3973
 __enumext_anskey_wrapper:n 1761, 2121
 __enumext_at_begin_document:n . . 28, 184, 184,
 252, 258
 __enumext_before_args_exec: 39, 766, 766, 2749
 __enumext_before_args_exec_v: 39, 40, 782, 782,
 2894
 __enumext_before_args_exec_vii: . . 798, 798,
 3284

__enumext_before_args_exec_viii: 802, 3650
 __enumext_before_keys_exec: 39, 766, 770, 2634
 __enumext_before_keys_exec_v: .. 39, 782, 786, 2859
 __enumext_before_keys_exec_vii: 798
 __enumext_before_keys_exec_vii: 40, 806, 3204
 __enumext_before_keys_exec_viii: .. 40, 810, 3612
 __enumext_before_list: ... 79, 2628, 2746, 2746
 __enumext_before_list_v: . 83, 2854, 2891, 2891
 __enumext_before_list_vii: 91, 3199, 3280, 3280
 __enumext_before_list_viii: .. 98, 3608, 3647, 3647
 \l__enumext_before_no_starred_key_v_tl 788
 \l__enumext_before_no_starred_key_vii_tl 808
 \l__enumext_before_no_starred_key_viii_tl 812
 \l__enumext_before_starred_key_v_tl ... 784
 \l__enumext_before_starred_key_vii_tl . 800
 \l__enumext_before_starred_key_viii_tl 804
 __enumext_calc_hspace:NNNNNNN 75, 2519, 2519, 2550, 2555, 2595
 \l__enumext_check_ans_bool ... 71, 72, 134, 1687, 1691, 1739, 1958, 2256, 2392, 2422, 2826, 3395
 __enumext_check_ans_exec: .. 59, 80, 1737, 1737, 2750, 3283
 \g__enumext_check_ans_item_tl .. 69, 134, 2255, 2263, 2267
 __enumext_check_ans_set: . 58, 1701, 1701, 1741
 __enumext_check_ans_show: 59, 1744, 1744, 2843, 3482
 \g__enumext_check_ans_show_bool 81, 134, 2829, 2841, 2846
 \g__enumext_check_ans_show_h_bool 134, 3480, 3485
 \l__enumext_columns_sep_v_dim 2913, 2915, 2923
 \l__enumext_columns_sep_vii_dim .. 3081, 3083, 3092, 3137, 3272, 3460
 \l__enumext_columns_sep_viii_dim . 3491, 3493, 3502, 3547, 3821
 \l__enumext_columns_v_int 1071, 2911, 2919, 2931, 2936
 \l__enumext_columns_vii_int .. 3086, 3089, 3093, 3101, 3105, 3108, 3114, 3120, 3124, 3259, 3455, 3466
 \l__enumext_columns_viii_int . 3496, 3499, 3503, 3511, 3515, 3518, 3524, 3530, 3534, 3816, 3827
 \g__enumext_count_item_anskey_int .. 69, 134, 1747, 1755, 1960, 2258
 \g__enumext_count_item_number_int 134, 1712, 1717, 1720, 1723, 1731, 1747, 1754, 2394, 2424, 3397
 \g__enumext_count_item_with_ans_int 63
 \l__enumext_counter_i_tl 32, 325
 \l__enumext_counter_ii_tl 32, 326
 \l__enumext_counter_iii_tl 32, 327
 \l__enumext_counter_iv_tl 32, 328
 \c__enumext_counter_style_tl 27, 37, 204
 \g__enumext_counter_styles_tl . 23, 31, 55, 336, 354
 \l__enumext_counter_v_tl 32, 329, 555
 \l__enumext_counter_vi_tl 32, 330
 \l__enumext_counter_vii_tl 32, 331, 499
 \l__enumext_counter_viii_tl 32, 332, 509
 __enumext_current_env_set_bool: . 28, 78, 224, 224, 2645, 3220
 \l__enumext_current_widest_dim 23, 55, 360, 437, 484, 541, 545
 __enumext_default_item:n ... 2388, 2388, 2437
 __enumext_define_counters:Nn 23, 316, 316, 325, 326, 327, 328, 329, 330, 331, 332
 __enumext_endminipage: . 28, 258, 261, 949, 3075, 3440, 3801
 __enumext_fake_item: 690, 690, 2582
 \l__enumext_fake_item_indent_v_dim 709, 714
 \l__enumext_fake_item_indent_v_tl 711, 2445, 2449, 2457
 \l__enumext_fake_item_indent_vii_dim 721, 726
 \l__enumext_fake_item_indent_vii_tl 723, 3436
 \l__enumext_fake_item_indent_viii_dim . 733, 738, 3793
 \l__enumext_fake_item_indent_viii_tl .. 735, 3792, 3796
 \l__enumext_fake_item_indent_X_tl 85
 __enumext_fake_item_vii: 690, 718, 2612
 __enumext_fake_item_viii: 690, 730, 2617
 __enumext_filter_series:n 52, 1389, 1389, 1431, 1443, 1448
 __enumext_filter_series_key:n 52, 1389, 1394, 1398
 __enumext_filter_series_pair:nn .. 52, 1389, 1395, 1407
 \g__enumext_footnote_arg_seq . 156, 2361, 2374, 2384
 \g__enumext_footnote_int . 156, 2368, 2371, 2373, 2375
 \g__enumext_footnote_int_seq . 156, 2362, 2375, 2380, 2383
 __enumext_footnotes_key_bool 29
 \l__enumext_footnotes_key_bool 25, 29, 94, 146, 274, 279, 288, 3403, 3451, 3766, 3812
 __enumext_footnotetext:nn ... 2355, 2355, 2385
 __enumext_getkeyans:nn .. 103, 3851, 3855, 3855
 __enumext_getkeyans_aux:n 102, 3839, 3842, 3842
 \l__enumext_hyperref_bool 25, 29, 146, 270, 291, 308, 2024, 2243, 3390, 3757
 __enumext_hypertarget:nn 29, 265, 293, 297, 313
 __enumext_if_is_int:n 196
 __enumext_if_is_int:nTF 196, 574, 588
 \l__enumext_item_column_pos_vii_int 92, 3108, 3114, 3120, 3124, 3131, 3318, 3455, 3458
 \l__enumext_item_column_pos_viii_int ... 99, 3518, 3524, 3530, 3534, 3541, 3662, 3816, 3819
 \l__enumext_item_column_pos_X_int 159
 \g__enumext_item_count_all_vii_int 92, 3132, 3319, 3466, 3473
 \g__enumext_item_count_all_viii_int 99, 3542, 3663, 3827, 3834
 \g__enumext_item_count_all_X_int 159
 __enumext_item_peek_args_vii: 92, 3320, 3322, 3322
 __enumext_item_peek_args_viii: 99, 3664, 3666, 3666
 __enumext_item_starred: .. 74, 2479, 2479, 2497
 \l__enumext_item_starred_vii_bool 3337, 3353, 3407
 \l__enumext_item_starred_viii_bool 3681, 3697, 3770, 3790
 \l__enumext_item_starred_X_bool 159


```

\__enumext_item_std:w 28, 71-73, 87, 252, 256, 2397,
    2403, 2427, 2445, 2449, 2457, 3056
\g__enumext_item_symbol_aux_vii_tl 3361, 3409,
    3412, 3416, 3418
\g__enumext_item_symbol_aux_X_tl . . . . . 159
\l__enumext_item_symbol_sep_vii_dim . . 3370,
    3378, 3415, 3417
\g__enumext_item_symbol_tl 23, 72, 48, 2412, 2485,
    2502
\l__enumext_item_symbol_vii_tl . . . . . 3412
\l__enumext_item_text_vii_box 3402, 3443, 3450
\l__enumext_item_text_viii_box 3765, 3804, 3811
\l__enumext_item_text_X_box . . . . . 159
\l__enumext_item_width_vii_dim . . 3090, 3135,
    3143, 3144
\l__enumext_item_width_viii_dim . . 3500, 3545,
    3553, 3554
\l__enumext_item_width_X_dim . . . . . 159
\l__enumext_itemindent_X_dim . . . . . 59
\l__enumext_itemsep_vii_skip . . . . . 3472
\l__enumext_itemsep_viii_skip . . . . . 3833
\l__enumext_joined_item_aux_vii_int . . 3129,
    3130, 3131, 3132, 3138
\l__enumext_joined_item_aux_viii_int . 3539,
    3540, 3541, 3542, 3548
\l__enumext_joined_item_aux_X_int . . . 159
\__enumext_joined_item_vii:w . . 92, 3325, 3326,
    3328, 3328
\l__enumext_joined_item_vii_int . . 3100, 3101,
    3104, 3106, 3112, 3117, 3122, 3127, 3129, 3135
\__enumext_joined_item_viii:w . 99, 3669, 3670,
    3672, 3672
\l__enumext_joined_item_viii_int . 3510, 3511,
    3514, 3516, 3522, 3527, 3532, 3537, 3539, 3545
\l__enumext_joined_item_X_int . . . . . 159
\l__enumext_joined_width_vii_dim . 3133, 3140,
    3143, 3433, 3445
\l__enumext_joined_width_viii_dim 3543, 3550,
    3553, 3787, 3806
\l__enumext_joined_width_X_dim . . . . . 159
\__enumext_keyans_addto_prop:n 67, 2145, 2145,
    2459, 2988
\__enumext_keyans_addto_seq:n . 68, 2219, 2219,
    2461, 2990
\__enumext_keyans_addto_seq_link: 2219, 2237,
    2239, 3728
\__enumext_keyans_anspic_code:nnn . 85, 2979,
    2982, 2982
\__enumext_keyans_check_ans:nn 69, 2261, 2261,
    2864, 3020, 3622
\__enumext_keyans_default_item:n . . 73, 2440,
    2440, 2475
\l__enumext_keyans_env_bool 20, 2715, 2728, 2875,
    2959
\__enumext_keyans_fake_item: . . 690, 706, 2572
\l__enumext_keyans_item_opt_tl 97, 2273, 2286,
    2292, 3713
\l__enumext_keyans_level_h_int 20, 2197, 3628,
    3629
\l__enumext_keyans_level_int . . 20, 1224, 1947,
    2192, 2874, 2878, 2973
\__enumext_keyans_make_label: 31, 74, 2505, 2505,
    2570
\__enumext_keyans_mini_addvspace: 46, 83, 1128,
    1128, 2903
\__enumext_keyans_mini_right_cmd:n 49, 1226,
    1249, 1249
\__enumext_keyans_mini_set_vskip: . 46, 1066,
    1066, 1130
\__enumext_keyans_multi_addvspace: . 83, 915,
    926, 2928
\__enumext_keyans_multi_set_vskip: . 42, 915,
    915, 928
\__enumext_keyans_multicols_start: 83, 2907,
    2909, 2909
\__enumext_keyans_multicols_stop: . 84, 1253,
    2934, 2934, 2958
\__enumext_keyans_parse_keys:n 2853, 2887, 2887
\l__enumext_keyans_pic_above_int . 129, 3066,
    3067, 3069
\l__enumext_keyans_pic_above_skip . . 87, 129,
    3011, 3050
\__enumext_keyans_pic_arg_two: 86, 3009, 3038,
    3038
\l__enumext_keyans_pic_below_int . 129, 3066,
    3067, 3070
\l__enumext_keyans_pic_body_seq . . 85-87, 129,
    2977, 3016, 3074
\__enumext_keyans_pic_do:n 87, 3016, 3018, 3058,
    3058, 3062
\l__enumext_keyans_pic_level_int . . 20, 1216,
    1951, 2148, 2187, 2222, 2305, 3027, 3028
\__enumext_keyans_pic_row:n 87, 3060, 3063, 3063
\__enumext_keyans_pic_safe_exec: . . 86, 3005,
    3025, 3025
\__enumext_keyans_pic_skip_abs:N . . 86, 3033,
    3033, 3049
\l__enumext_keyans_pic_width_dim . 129, 3065,
    3072
\__enumext_keyans_redefine_item: . . 73, 2463,
    2463, 2569
\__enumext_keyans_ref: . . . . . 35, 552, 565, 2571
\__enumext_keyans_ref:n . . . . . 34, 549, 552, 552
\__enumext_keyans_safe_exec: . 2852, 2868, 2868
\__enumext_keyans_show_ans: . . 2269, 2277, 2296
\__enumext_keyans_show_item_opt: . 2269, 2284,
    2457, 3001, 3793
\__enumext_keyans_show_left:n . 73, 2269, 2269,
    2455, 2996
\__enumext_keyans_show_pos: . . 2269, 2281, 2303
\__enumext_keyans_starred_item:n . . 73, 2452,
    2452, 2471
\__enumext_keyans_store_ref: . . 67, 2166, 2166,
    2460, 2989, 3726
\__enumext_keyans_store_ref_aux_i: 68, 2166,
    2178, 2181
\__enumext_keyans_store_ref_aux_ii: 68, 2166,
    2207, 2209
\l__enumext_keyans_tmpa_dim . . . . . 97
\l__enumext_keyans_tmpa_tl 24, 100, 97, 2454, 2458
\l__enumext_keyans_tmpb_tl . . . . . 100, 97
\__enumext_keyans_wrapper_opt:n . . 1764, 2292
\l__enumext_label_copy_i_tl . . 2078, 2185, 2190,
    2195, 2200
\l__enumext_label_copy_v_tl . . . . . 2195
\l__enumext_label_copy_vi_tl . . . . . 2190
\l__enumext_label_copy_vii_tl 2053, 2064, 2095,
    2185

```



```

\l__enumext_label_copy_viii_tl . . . . . 2200
\l__enumext_label_copy_X_tl . . . . . 148
\l__enumext_label_fill_left_v_tl . . . . . 2509
\l__enumext_label_fill_left_X_tl . . . . . 85
\l__enumext_label_fill_right_v_tl . . . . . 2516
\l__enumext_label_fill_right_X_tl . . . . . 85
\l__enumext_label_font_style_v_tl 2510, 3000
\l__enumext_label_font_style_vii_tl . . . 3421
\l__enumext_label_font_style_viii_tl . . 3775
\l__enumext_label_i_tl . . . . . 429
\l__enumext_label_ii_tl . . . . . 429
\l__enumext_label_iii_tl . . . . . 429
\l__enumext_label_iv_tl . . . . . 429
\__enumext_label_style:Nnn 23, 31, 349, 349, 364,
    434, 481, 538, 542
\l__enumext_label_v_tl . . 67, 68, 535, 2153, 2227,
    2298, 2332, 2454, 2458, 2856, 2995, 2997
\l__enumext_label_vi_tl . 67, 68, 535, 2150, 2224,
    2995, 2997, 3001
\l__enumext_label_vii_tl . 476, 3348, 3373, 3380
\l__enumext_label_viii_tl 476, 3692, 3720, 3724
\l__enumext_label_width_by_box . . 55, 345, 346
\__enumext_label_width_by_box:Nn 31, 343, 343,
    348, 360, 598
\l__enumext_labelsep_i_dim . . 2300, 2336, 3731,
    3746
\l__enumext_labelsep_v_dim . . . . . 2918
\l__enumext_labelsep_vii_dim . 3085, 3094, 3136,
    3371, 3431, 3447
\l__enumext_labelsep_viii_dim 3495, 3504, 3546,
    3785, 3808
\l__enumext_labelwidth_i_dim . 2300, 2335, 3731,
    3746
\l__enumext_labelwidth_v_dim . . . . . 2918
\l__enumext_labelwidth_vii_dim . . 3085, 3093,
    3136, 3424, 3428, 3446
\l__enumext_labelwidth_viii_dim . . 3495, 3503,
    3546, 3778, 3782, 3807
\l__enumext_leftmargin_tmp_v_bool . 86, 3040
\l__enumext_leftmargin_tmp_X_bool . . . . . 59
\l__enumext_leftmargin_tmp_X_dim . . . . . 59
\l__enumext_leftmargin_X_dim . . . . . 59
\__enumext_level: 192, 192, 458, 461, 462, 471, 473,
    693, 697, 701, 768, 772, 776, 780, 864, 866, 868, 870,
    903, 905, 907, 909, 913, 953, 956, 975, 984, 990, 995,
    999, 1010, 1014, 1015, 1020, 1056, 1060, 1233, 1239,
    1286, 1288, 1290, 1293, 1300, 1302, 1304, 1307, 1855,
    1863, 1867, 1871, 2116, 2119, 2120, 2396, 2397, 2401,
    2402, 2403, 2410, 2412, 2416, 2417, 2420, 2426, 2427,
    2481, 2484, 2486, 2493, 2494, 2495, 2498, 2501, 2631,
    2633, 2683, 2688, 2689, 2690, 2692, 2696, 2701, 2702,
    2703, 2705, 2721, 2734, 2741, 2752, 2754, 2757, 2758,
    2760, 2765, 2772, 2775, 2777, 2779, 2780, 2781, 2782,
    2785, 2791, 2796, 2802, 2805, 2807, 2813
\l__enumext_level_h_int . 20, 232, 497, 521, 1709,
    1726, 2072, 2089, 2654, 3221, 3222, 3230
\l__enumext_level_int 78, 20, 194, 243, 876, 1027,
    1220, 1703, 2049, 2059, 2065, 2071, 2079, 2087, 2094,
    2585, 2646, 2647, 2653, 2666, 2673, 2719, 2732, 2787,
    2839, 2882, 2969, 3231, 3299, 3309, 3478, 3635
\__enumext_list_arg_two_i: . . . . . 2551
\__enumext_list_arg_two_ii: . . . . . 2551
\__enumext_list_arg_two_iii: . . . . . 2551
\__enumext_list_arg_two_iv: . . . . . 2551
\__enumext_list_arg_two_v: . 73, 2551, 2858, 3041
\__enumext_list_arg_two_vii: . . . . . 2591, 3203
\__enumext_list_arg_two_viii: . . . . . 2591, 3611
\l__enumext_listoffset_v_dim . . . . . 2920
\l__enumext_listparindent_vii_dim . . . . 3434
\l__enumext_listparindent_viii_dim . . . 3788
\__enumext_make_label: 31, 72, 74, 2489, 2489, 2580
\l__enumext_mark_answer_sym_tl . 62, 124, 1770,
    1925, 2131, 2307, 2320, 3735
\l__enumext_mark_position_str 124, 1774, 1775,
    1798, 1799, 1923
\l__enumext_mark_ref_sym_tl . . 124, 1784, 2029,
    2251
\__enumext_mini_addvspace: . . 45, 80, 1049, 1049,
    2762
\__enumext_mini_addvspace_vii: 48, 1202, 1202,
    3161
\__enumext_mini_addvspace_viii: 48, 1202, 1208,
    3571
__enumext_mini_env* . . . . . 943
\__enumext_mini_right_cmd:n . 48, 49, 1228, 1230,
    1230
\__enumext_mini_set_vskip: . . 43, 950, 950, 1051
\__enumext_mini_set_vskip_vii: 47, 1145, 1145,
    1204
\__enumext_mini_set_vskip_viii: 47, 1145, 1167,
    1210
\__enumext_minipage:w 28, 258, 260, 945, 3072, 3433,
    3787
\l__enumext_minipage_active_v_bool . . 83, 84,
    2901, 2926, 2939, 2947
\g__enumext_minipage_active_vii_bool . . . 89,
    3172, 3177, 3189
\l__enumext_minipage_active_vii_bool . 3157,
    3168
\g__enumext_minipage_active_viii_bool 3582,
    3587, 3599
\l__enumext_minipage_active_viii_bool 3567,
    3578
\g__enumext_minipage_active_X_bool . . . 159
\l__enumext_minipage_active_X_bool . . . 73
\g__enumext_minipage_after_skip 73, 1149, 1161,
    3187, 3597
\l__enumext_minipage_after_skip 43, 45, 81, 84,
    73, 966, 981, 1001, 1017, 1032, 1038, 1044, 1058, 1068,
    1077, 1080, 1092, 1110, 1121, 1137, 1169, 1182, 1196,
    2822, 2956
\g__enumext_minipage_center_vii_bool . 3181,
    3190
\g__enumext_minipage_center_viii_bool 3591,
    3600
\g__enumext_minipage_center_X_bool . . . 159
\l__enumext_minipage_hsep_v_dim . . . 83, 2899
\l__enumext_minipage_hsep_vii_dim . . . 3155
\l__enumext_minipage_hsep_viii_dim . . . 3565
\l__enumext_minipage_left_skip 43, 83, 73, 958,
    973, 992, 1007, 1054, 1064, 1069, 1075, 1084, 1101,
    1113, 1133, 1143, 1147, 1152, 1156, 1170, 1174, 1188,
    1206, 1212
\l__enumext_minipage_left_v_dim 83, 2897, 2905
\l__enumext_minipage_left_vii_dim 3151, 3163
\l__enumext_minipage_left_viii_dim 3561, 3573
\l__enumext_minipage_left_X_dim . . . . . 73
\g__enumext_minipage_right_skip 73, 1148, 1153,

```

1157, 3180, 3590
 \l__enumext_minipage_right_skip . . 43, 73, 962,
 977, 997, 1012, 1070, 1076, 1088, 1106, 1117, 1171,
 1178, 1192, 1240, 1257
 \l__enumext_minipage_right_v_dim . . 83, 1251,
 1256, 2895, 2899
 \g__enumext_minipage_right_vii_dim 89, 3159,
 3179, 3192
 \l__enumext_minipage_right_vii_dim 89, 3149,
 3154, 3160
 \g__enumext_minipage_right_viii_dim . . 3569,
 3589, 3602
 \l__enumext_minipage_right_viii_dim . . 3559,
 3564, 3570
 \g__enumext_minipage_right_X_dim 159
 \g__enumext_minipage_right_X_skip 159
 \g__enumext_minipage_stat_int . 80, 83, 73, 1245,
 1262, 2761, 2815, 2820, 2902, 2949, 2954
 \g__enumext_miniright_code_vii_tl . 89, 3185,
 3191
 \g__enumext_miniright_code_viii_tl 3595, 3601
 \g__enumext_miniright_code_X_tl 159
 __enumext_multi_addvspace: . . . 42, 80, 898, 898,
 2793
 __enumext_multi_set_vskip: . . 41, 862, 862, 900
 \l__enumext_multicols_above_ii_skip . . . 881
 \l__enumext_multicols_above_iii_skip . . 887
 \l__enumext_multicols_above_iv_skip . . . 893
 \l__enumext_multicols_above_v_skip 917, 931,
 941
 \l__enumext_multicols_above_X_skip 67
 \l__enumext_multicols_below_v_skip 921, 935,
 2941
 \l__enumext_multicols_below_X_skip 67
 __enumext_multicols_start: 80, 2767, 2769, 2769
 __enumext_multicols_stop: 81, 1235, 2799, 2799,
 2824
 __enumext_newlabel:nn 26, 29, 66, 301, 301, 2105,
 2213
 \l__enumext_newlabel_arg_one_tl 26, 29, 66, 68,
148, 2028, 2098, 2106, 2202, 2214, 2249
 \l__enumext_newlabel_arg_two_tl 26, 29, 65, 148,
 2052, 2062, 2076, 2092, 2107, 2189, 2194, 2199, 2215
 __enumext_parse_keys:n . . . 52, 2627, 2661, 2661
 __enumext_parse_keys_vii:n 52, 3198, 3238, 3238
 __enumext_parse_keys_viii:n . 3607, 3640, 3640
 __enumext_parse_series:n . . 52, 78, 1419, 1419,
 2669, 3244
 __enumext_parse_store_keys:n . 78, 2677, 2681,
 2681
 __enumext_parse_store_keys_vii:n 90, 91, 3247,
3251, 3251
 \l__enumext_parsep_i_skip 879, 881, 1030, 1078
 \l__enumext_parsep_ii_skip . . . 885, 887, 1036
 \l__enumext_parsep_iii_skip . . . 891, 893, 1042
 \l__enumext_parsep_vii_skip 3435
 \l__enumext_parsep_viii_skip 3789
 \l__enumext_partopsep_v_skip . . 933, 937, 1104,
 1108, 1115, 1119, 1135, 1139
 \l__enumext_partopsep_viii_skip 1180
 __enumext_phantomsection: 29, 265, 294, 298, 314
 __enumext_print_footnote: . . . 2355, 2378, 3453,
 3814
 __enumext_print_keyans_box:NN 62, 1917, 1917,
 1930, 2118, 2300, 2334, 3731, 3746
 \l__enumext_print_keyans_i_tl 3865, 3894
 \l__enumext_print_keyans_ii_tl . . . 3870, 3895
 \l__enumext_print_keyans_iii_tl . . 3875, 3896
 \l__enumext_print_keyans_iv_tl . . . 3880, 3897
 \l__enumext_print_keyans_vii_tl . . 3885, 3898
 \l__enumext_print_keyans_X_tl 113
 __enumext_printkeyans:nnn 103, 3899, 3902, 3902
 __enumext_redefine_item: . 73, 2429, 2429, 2579
 \l__enumext_ref_key_arg_tl 33, 37, 207, 451, 452,
 465, 496, 505, 515, 554, 561
 \l__enumext_ref_the_count_tl . 33, 37, 458, 461,
 464, 499, 501, 504, 509, 511, 514, 555, 557, 560
 __enumext_regex_counter_syle: 27, 33, 202, 202,
 459, 500, 510, 556
 __enumext_register_counter_style:Nn . . 333,
 333, 338, 339, 340, 341, 342
 __enumext_remove_extra_parsep_vii: . . 3213,
3462, 3462
 __enumext_remove_extra_parsep_viii: . 3621,
3823, 3823
 __enumext_renew_footnote: . . . 2355, 2359, 3405,
 3768
 \l__enumext_renew_the_count_v_tl . . 558, 569
 \l__enumext_renew_the_count_vi_tl 567
 \l__enumext_renew_the_count_vii_tl 502, 512,
 523, 525
 \l__enumext_renew_the_count_viii_tl 529, 531
 \l__enumext_renew_the_count_X_tl 37
 \l__enumext_resume_active_bool 52, 55, 48, 1423,
 1543
 \l__enumext_resume_bool 23
 __enumext_resume_counter: . . 54, 55, 1541, 1547,
 1554
 __enumext_resume_counter:n . 52, 55, 1512, 1517,
1541, 1541, 1611, 1619
 __enumext_resume_counter_save_ans: 55, 1541,
 1552, 1584
 __enumext_resume_counter_series: . 55, 1541,
 1550, 1567
 \g__enumext_resume_int . 23, 48, 1464, 1558, 1559
 __enumext_resume_last:n 52, 53, 1419, 1425, 1438
 \l__enumext_resume_name_tl 48, 1460, 1468, 1471,
 1487, 1495, 1498, 1544, 1545, 1573, 1580
 __enumext_resume_save_counter: 53, 1451, 1451,
 2835, 3293
 __enumext_resume_series:n . 54, 1383, 1508, 1508
 __enumext_resume_starred: . 56, 1384, 1605, 1605
 \g__enumext_resume_vii_int . . 91, 48, 1491, 1563,
 1564
 __enumext_safe_exec: . . 28, 78, 2626, 2643, 2643
 __enumext_safe_exec_vii: . 28, 3197, 3218, 3218
 __enumext_safe_exec_viii: . . 3606, 3626, 3626
 \l__enumext_series_name_tl 55
 \l__enumext_series_str . 53, 78, 1381, 1421, 1429,
 1430, 1432, 1434, 1455, 1458, 1462, 1482, 1485, 1489,
 2665, 3242
 __enumext_set_error:nn 3980, 3990, 3992
 __enumext_set_parse:n 3963, 3980, 3980
 \l__enumext_setkey_tmpa_int . . . 108, 3956, 3960
 \l__enumext_setkey_tmpa_seq . . 108, 3954, 3964,
 3970, 3972, 3974, 3987
 \l__enumext_setkey_tmpa_tl 108, 3962, 3966

```

\l__enumext_setkey_tmpb_seq .. 108, 3955, 3958,
    3962, 3963
\l__enumext_setkey_tmpb_tl 108, 3982, 3984, 3985
\l__enumext_show_answer_bool . 124, 1778, 1802,
    2125, 2275, 2289, 2992, 3729
\__enumext_show_length:nnn .. 38, 210, 210, 4032,
    4033, 4034, 4035, 4036, 4037, 4038, 4039, 4040, 4041,
    4047, 4048, 4049, 4050, 4051, 4052, 4053, 4054, 4055,
    4056
\l__enumext_show_position_bool 124, 1781, 1805,
    2129, 2279, 2290, 2993, 3733
\g__enumext_standar_bool . 28, 20, 231, 234, 1453,
    1518, 1530, 1556, 1569, 1607, 1729, 2652, 2845
\l__enumext_standar_bool 81, 20, 2057, 2070, 2086,
    2649, 2834
\g__enumext_standar_keyans_pic_star_env_-
    int ..... 145
\g__enumext_standar_keyans_star_env_int 144
\l__enumext_standar_level_one_bool .. 78, 20,
    1440, 1587, 1638, 1650, 2658
\__enumext_standar_ref: .... 33, 449, 469, 2581
\__enumext_standar_ref:n .... 33, 441, 449, 449
\g__enumext_standar_series_tl . 48, 1442, 1443,
    1609, 1612
\g__enumext_standar_star_env_int .. 141, 235,
    1641, 1654
\g__enumext_standard_bool ..... 78
\l__enumext_standard_bool ..... 78
\__enumext_standard_item_vii:w .. 92, 93, 3333,
    3335, 3335
\__enumext_standard_item_viii:w 99, 3677, 3679,
    3679
\g__enumext_starred_bool 28, 90, 91, 20, 242, 245,
    1480, 1523, 1534, 1561, 1576, 1615, 1708, 2048, 2058,
    2088, 2183, 2716, 2729, 2827, 3193, 3229, 3484
\l__enumext_starred_bool . 90, 91, 20, 1981, 1989,
    2073, 2114, 3226, 3292
\__enumext_starred_columns_set_vii: .. 3079,
    3079, 3206
\__enumext_starred_columns_set_viii: . 3489,
    3489, 3614
\__enumext_starred_item:nn ... 2406, 2406, 2435
\__enumext_starred_item_exec: . 100, 3722, 3722,
    3772
\__enumext_starred_item_vii:w 92, 93, 3332, 3351,
    3351
\__enumext_starred_item_vii_aux_i:w .. 3351,
    3356, 3359
\__enumext_starred_item_vii_aux_ii:w . 3351,
    3357, 3362, 3364
\__enumext_starred_item_vii_aux_iii:w 3351,
    3367, 3376
\__enumext_starred_item_viii:w . 99, 100, 3676,
    3695, 3695
\__enumext_starred_item_viii_aux_i:w . 3695,
    3700, 3703
\__enumext_starred_item_viii_aux_ii:w 3695,
    3701, 3715, 3717
\__enumext_starred_joined_item_vii:n . 88, 92,
    3098, 3098, 3330
\__enumext_starred_joined_item_viii:n 96, 99,
    3508, 3508, 3674
\g__enumext_starred_keyans_star_env_int 143
\l__enumext_starred_level_one_bool 20, 1445,
    1596, 1643, 1657, 3235
\__enumext_starred_ref: .... 34, 494, 519, 2609
\__enumext_starred_ref:n .... 33, 488, 494, 494
\g__enumext_starred_series_tl . 48, 1447, 1448,
    1617, 1620
\g__enumext_starred_star_env_int .. 142, 246,
    1646, 1661
\__enumext_start_from:NNn 35, 572, 572, 585, 607
\l__enumext_start_i_int ..... 1559, 1571, 1590
\__enumext_start_item_tmp_vii: 90, 3209, 3315,
    3315
\__enumext_start_item_tmp_viii: 97, 3617, 3659,
    3659
\__enumext_start_item_vii:w 93, 3343, 3348, 3373,
    3380, 3382, 3382
\__enumext_start_item_viii:w .. 99, 3687, 3692,
    3720, 3749, 3749
\__enumext_start_list:nn 28, 75, 86, 252, 254, 2630,
    2855, 3006, 3201, 3609
\__enumext_start_mini_vii: . 91, 3147, 3147, 3285
\__enumext_start_mini_viii: 98, 3557, 3557, 3651
\__enumext_start_store_level: . 79, 2629, 2710,
    2710
\__enumext_start_store_level_vii: . 92, 3200,
    3295, 3295
\l__enumext_start_vii_int ... 1564, 1578, 1599
\l__enumext_start_X_int ..... 85, 602
\__enumext_stop_item_tmp_vii: . 90, 92, 93, 3208,
    3212, 3317, 3384
\__enumext_stop_item_tmp_viii: .. 97, 99, 3616,
    3620, 3661, 3751
\__enumext_stop_item_vii: 93, 94, 3384, 3438, 3438
\__enumext_stop_item_viii: 101, 3751, 3799, 3799
\__enumext_stop_list: .. 28, 252, 255, 2639, 2865,
    3019, 3214, 3623
\__enumext_stop_mini_vii: 89, 91, 3166, 3166, 3289
\__enumext_stop_mini_viii: . 98, 3557, 3576, 3655
\__enumext_stop_store_level: .. 79, 2640, 2710,
    2739
\__enumext_stop_store_level_vii: .. 92, 3215,
    3295, 3305
\l__enumext_store_active_bool 24, 57, 78, 90, 97,
    1588, 1597, 1668, 1943, 2675, 2714, 2727, 2870, 2877,
    2965, 3023, 3245, 3297, 3307, 3634
\__enumext_store_addto_prop:n 61, 67, 1836, 1837,
    1845, 1968, 2164, 3725
\__enumext_store_addto_seq:n 61, 69, 1846, 1846,
    1850, 1857, 1871, 1879, 1888, 1906, 1914, 2032, 2254
\l__enumext_store_ans_bool . 57, 134, 1669, 1690,
    1853, 1877, 1884, 1912, 1956
\l__enumext_store_anskey_arg_tl .. 24, 64, 97,
    1974, 1983, 1985, 1991, 1999, 2002, 2012, 2017, 2020,
    2026, 2032
\__enumext_store_anskey_code:nnnn . 63, 1962,
    1966, 1966
\__enumext_store_anskey_show_left:n 66, 1973,
    2123, 2123
\__enumext_store_anskey_show_wrap:n 66, 2111,
    2111, 2127, 2142
\l__enumext_store_columns_break_bool . 1937,
    1980
\l__enumext_store_columns_join_int 97, 1988,
    1993
\l__enumext_store_columns_sep_vii_bool 3266
\l__enumext_store_columns_sep_vii_dim 3271,

```

3275	
\l__enumext_store_columns_sep_X_bool ..	113
\l__enumext_store_columns_sep_X_dim ...	113
\l__enumext_store_columns_vii_bool ...	3253
\l__enumext_store_columns_vii_int	3258 , 3262
\l__enumext_store_columns_X_bool	113
\l__enumext_store_columns_X_int	113
__enumext_store_internal_ref: ..	63 , 65 , 1971 , 2034 , 2034
\l__enumext_store_item_symbol_sep_dim	1935 , 2009 , 2014
\l__enumext_store_item_symbol_tl .	1933 , 2000 , 2004
\l__enumext_store_keyans_item_opt_sep_- tl	1767 , 2158 , 2160 , 2231 , 2233 , 3708 , 3710
\l__enumext_store_keyans_item_opt_tl ...	97
\l__enumext_store_keyans_label_tl	24 , 67 – 69 , 97 , 2147 , 2150 , 2153 , 2160 , 2162 , 2164 , 2221 , 2224 , 2227 , 2233 , 2235 , 2245 , 2254 , 2255 , 3705 , 3710 , 3711 , 3724 , 3725 , 3727
__enumext_store_level_close: .	61 , 1851 , 1875 , 2743
__enumext_store_level_close_vii:	1882 , 1910 , 3311
__enumext_store_level_open: ..	60 , 61 , 78 , 1851 , 1851 , 2722 , 2735
__enumext_store_level_open_vii: ..	91 , 1882 , 1882 , 3301
\g__enumext_store_name_tl	24 , 81 , 97 , 1749 , 1752 , 2830 , 2847 , 3486
\l__enumext_store_name_tl	24 , 57 , 97 , 1474 , 1477 , 1501 , 1504 , 1592 , 1601 , 1635 , 1636 , 1654 , 1661 , 1670 , 1672 , 1674 , 1676 , 1678 , 1680 , 1839 , 1841 , 1848 , 2100 , 2101 , 2137 , 2204 , 2205 , 2313 , 2326 , 2830 , 3741
\l__enumext_store_opt_vii_tl .	1886 , 1896 , 1902 , 1906 , 3260 , 3273
\l__enumext_store_opt_X_tl	113
\l__enumext_store_ref_key_bool	63 , 1787 , 1969 , 2023 , 2168 , 2242
\l__enumext_store_upper_level_X_bool ..	113
\l__enumext_store_write_aux_file_tl	26 , 66 , 68 , 148 , 2103 , 2109 , 2211 , 2217
__enumext_storing_exec: .	57 , 1633 , 1655 , 1662 , 1666
__enumext_storing_set:n ..	57 , 1628 , 1633 , 1633
\l__enumext_the_counter_v_tl	557
\l__enumext_the_counter_vii_tl	501
\l__enumext_the_counter_viii_tl	511
\l__enumext_the_counter_X_tl	37
__enumext_tmp:n	32 , 36 , 41 , 47 , 59 , 66 , 67 , 72 , 79 , 84 , 85 , 96 , 113 , 123 , 151 , 155 , 159 , 178 , 218 , 222 , 685 , 689 , 1377 , 1388 , 1624 , 1632 , 1683 , 1700 , 1757 , 1792 , 1793 , 1810 , 2036 , 2043 , 2044 , 2065 , 2079 , 2082 , 2094 , 2170 , 2177 , 2551 , 2590 , 2591 , 2623
__enumext_tmp:nn	365 , 386 , 387 , 415 , 416 , 428 , 602 , 621 , 666 , 684 , 742 , 750 , 751 , 765 , 830 , 846 , 847 , 861 , 1266 , 1282 , 1811 , 1835 , 2339 , 2354
__enumext_tmp:nnn	429 , 445 , 446 , 447 , 448 , 476 , 492 , 493
__enumext_tmp:nnnnnn	622 , 647 , 650 , 653 , 655 , 657 , 660 , 663
__enumext_tmp:w	3848 , 3851
\l__enumext_tmpa_vii_int	3089 , 3092
\l__enumext_tmpa_viii_int	3499 , 3502
\l__enumext_tmpa_X_int	159
\l__enumext_topsep_v_skip	919 , 923 , 1073 , 1086 , 1094 , 1099 , 1119 , 1123 , 3022 , 3053
\l__enumext_topsep_vii_skip ..	1150 , 1159 , 1163
\l__enumext_topsep_viii_skip .	1172 , 1194 , 1198
\l__enumext_vspace_a_star_v_bool	1315
\l__enumext_vspace_a_star_vii_bool ...	1337
\l__enumext_vspace_a_star_viii_bool ...	1348
\l__enumext_vspace_a_star_X_bool	85
__enumext_vspace_above: ..	50 , 1283 , 1283 , 2748
__enumext_vspace_above_v: .	50 , 1311 , 1311 , 2893
\l__enumext_vspace_above_v_skip ..	1313 , 1317 , 1319
__enumext_vspace_above_vii: ..	51 , 1333 , 1333 , 3282
\l__enumext_vspace_above_vii_skip	1335 , 1339 , 1341
__enumext_vspace_above_viii: .	51 , 1333 , 1344 , 3649
\l__enumext_vspace_above_viii_skip	1346 , 1350 , 1352
\l__enumext_vspace_b_star_v_bool	1326
\l__enumext_vspace_b_star_vii_bool ...	1359
\l__enumext_vspace_b_star_viii_bool ...	1370
\l__enumext_vspace_b_star_X_bool	85
__enumext_vspace_below: ..	50 , 1297 , 1297 , 2833
__enumext_vspace_below_v: .	50 , 1322 , 1322 , 2961
\l__enumext_vspace_below_v_skip ..	1324 , 1328 , 1330
__enumext_vspace_below_vii: ..	51 , 1355 , 1355 , 3291
\l__enumext_vspace_below_vii_skip	1357 , 1361 , 1363
__enumext_vspace_below_viii: .	51 , 1355 , 1366 , 3657
\l__enumext_vspace_below_viii_skip	1368 , 1372 , 1374
__enumext_widest_from:nnnn ..	35 , 586 , 586 , 601 , 613
\g__enumext_widest_label_tl	23 , 31 , 55 , 353 , 357 , 361
\l__enumext_wrap_label_opt_v_bool	2448
\l__enumext_wrap_label_opt_vii_bool	93 , 3342
\l__enumext_wrap_label_opt_viii_bool	99 , 3686
\l__enumext_wrap_label_opt_X_bool	85
\l__enumext_wrap_label_v_bool	2444 , 2448 , 2456 , 2511
\l__enumext_wrap_label_vii_bool	93 , 3341 , 3346 , 3354 , 3422
\l__enumext_wrap_label_viii_bool ..	99 , 3685 , 3690 , 3698 , 3776
\l__enumext_wrap_label_X_bool	85
__enumext_wrapper_label_v:n	2513 , 3001
__enumext_wrapper_label_vii:n	3425
__enumext_wrapper_label_viii:n	3779
__enumext_zero_count_level:	216 , 216
__enumext_zero_parsep: ...	45 , 970 , 1025 , 1025
enumext*	5, 3195
enumXi	325
enumXii	325
enumXiii	325
enumXiv	325
enumXv	325
enumXvi	325

enumXvii	325
enumXviii	325
Environments provide by enumext :	
enumext*	22, 23, 25–28, 30, 33, 34, 37, 38, 40, 41, 47, 48, 51–54, 56–65, 68, 70, 71, 77–79, 90–93, 95, 97, 99, 101, 103, 106, 108
enumext	22, 23, 25, 27, 28, 30–34, 36–54, 56–65, 68, 70, 71, 73–75, 77–79, 81, 82, 86, 87, 89, 92, 103, 106, 107
keyans*	22–24, 26, 27, 30, 33–35, 37, 38, 40, 41, 47, 48, 51, 57, 58, 60, 61, 67, 71, 77, 98, 106, 108
keyanspic	22–25, 30, 31, 34, 48, 57, 58, 61, 67–69, 84–87, 107
keyans	22–25, 27, 30, 31, 34, 36–40, 42, 46, 48–50, 57, 58, 60, 61, 67–69, 73–75, 82, 84–86, 89, 99, 106, 107
Environments:	
list	26, 28, 75, 77
lrbox	87, 94, 101
minipage	26, 28, 41, 43, 84, 86, 87, 94, 101
multicols	41–44, 48, 80, 81, 83, 84
exp commands:	
\exp_after:wN	3851
\exp_args:Ne	2672, 3839
\exp_not:N	45, 356, 464, 504, 514, 560, 699, 713, 714, 725, 726, 737, 738, 2028, 2134, 2135, 2247, 2310, 2311, 2323, 2324, 3738, 3739, 3848
\exp_not:n	464, 465, 504, 505, 514, 515, 560, 561, 700, 1405, 1417, 1819, 1826, 1993, 2004, 2014, 2028, 2029, 2106, 2214, 2249, 2251, 2692, 2705, 3262, 3275
F	
\fbox	1762
file commands:	
\file_input_stop:	4157
first	751
font	365
\footnote	71
\footnote	71, 2363
\footnotemark	2373
\footnotesize	2135, 2311, 2324, 3739
\footnotetext	2357
G	
\getkeyans	14, 102, 3837
group commands:	
\group_begin:	1955, 2133, 2309, 2322, 3401, 3420, 3737, 3764, 3774, 3859, 3893
\group_end:	1964, 2140, 2316, 2329, 3430, 3442, 3744, 3784, 3803, 3861, 3900
H	
\hbadness	3449, 3810
hbox commands:	
\hbox_set:Nn	345
\hfill	395, 399, 404, 405, 1237, 1255, 2028, 2247, 3171, 3581
hook commands:	
\hook_gput_code:nnn	9, 186, 190, 263
\hook_gset_rule:nnnn	264
\hspace	3460, 3821
\hyperlink	65, 69
\hyperlink	2028, 2247
\hypertarget	29
\hypertarget	293
I	
\IfHyperBoolean	271
\IfPackageLoadedTF	11, 267, 281

\ignorespaces	702
\inputlineno	235, 246
int commands:	
\int_add:Nn	3131, 3541
\int_case:nn	876, 1027, 1703, 1726
\int_compare:nNnTF	497, 521, 952, 1071, 1216, 1220, 1224, 1746, 1947, 1951, 2148, 2187, 2192, 2197, 2222, 2305, 2647, 2666, 2719, 2732, 2771, 2787, 2801, 2815, 2839, 2878, 2882, 2911, 2936, 2949, 2969, 2973, 3028, 3101, 3111, 3127, 3222, 3299, 3309, 3455, 3464, 3478, 3511, 3521, 3537, 3629, 3635, 3816, 3825, 3960
\int_compare_p:nNn	232, 243, 1709, 2049, 2059, 2071, 2072, 2087, 2089, 2653, 2654, 3230, 3231
\int_decr:N	3130, 3540
\int_eval:n	1841, 2101, 2135, 2205, 2311, 2324, 2566, 2608, 3119, 3529, 3739
\int_from_alph:n	580, 594
\int_from_roman:n	582, 596
\int_gadd:Nn	3132, 3542
\int_gdecr:N	1712, 1717, 1720, 1723, 1731
\int_gincr:N	1558, 1563, 1960, 2258, 2394, 2424, 2761, 2902, 3319, 3397, 3663
\int_gset:Nn	235, 246, 2371
\int_gset_eq:NN	1457, 1464, 1470, 1476, 1484, 1491, 1497, 1503, 2368
\int_gzero:N	220, 1245, 1262, 1754, 1755, 2820, 2954, 3473, 3834
\int_if_exist:NTF	1432, 1468, 1474, 1495, 1501, 1678
\int_incr:N	2646, 2874, 3027, 3221, 3318, 3628, 3662
\int_mod:nn	3466, 3827
\int_new:N	20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 110, 118, 131, 132, 139, 140, 141, 142, 143, 144, 145, 156, 162, 163, 164, 165, 166, 1434, 1680
\int_set:Nn	576, 580, 582, 1571, 1578, 1590, 1599, 1816, 1988, 3066, 3067, 3089, 3100, 3106, 3122, 3449, 3499, 3510, 3516, 3532, 3810, 3956
\int_set_eq:NN	1559, 1564, 2687, 3129, 3257, 3539
\int_step_function:nnN	2065, 2079, 2094
\int_step_inline:nnn	3068, 3983
\int_to_roman:n	194, 2045, 2083
\int_use:N	953, 1573, 1580, 1592, 1601, 1654, 1661, 2566, 2585, 2608, 2673, 2772, 2781, 2796, 2802, 3104, 3105, 3117, 3514, 3515, 3527
\int_zero:N	3458, 3819
\c_one_int	3089, 3108, 3114, 3120, 3124, 3127, 3499, 3518, 3524, 3530, 3534, 3537
\c_zero_int	232, 243, 2049, 2059, 2071, 2072, 2087, 2089, 3299, 3309, 3469, 3830
\item	28, 39, 40, 62, 71, 84, 85, 87, 90, 97
\item	71, 73, 92, 93, 99, 101, 256, 1859, 1865, 1890, 1898, 1985, 2224, 2227, 2431, 2465, 3207, 3209, 3615, 3617, 3727
\item*	6, 12, 2463
item-pos*	2339
item-sym*	2339
\itemindent	23, 76
\itemindent	75
itemindent	666
\itemsep	86, 87
\itemsep	3042, 3048
\itemwidth	3096, 3140, 3144, 3506, 3550, 3554
K	
keyans	12, 2850
keyans*	12, 3604
keyanspic	13, 3003

Keys for environments provide by **enumext**:

above*	24, 49-51
above	24, 49-51, 79, 83, 91, 98
after	38-40, 81, 84, 91, 98
align	24, 31, 74, 94
before*	38-40, 79, 91, 98
before	38-40, 83
below*	24, 49-51
below	24, 49-51, 81, 84, 91, 98
check-ans	24, 25, 27, 57, 58, 63, 69, 71-73, 80, 81, 95, 106
columns-sep*	25, 60, 78, 91
columns-sep	40, 61, 78, 80, 83, 91
columns*	25, 60, 78, 91
columns	24, 40, 41, 43, 49, 61, 78, 80, 83, 91
first	38-40, 94
font	31, 74, 94
item-pos*	63, 64, 70
item-sym*	23, 63, 64, 70, 72
item*-sep	72
itemindent	24, 37, 73, 94
itemsep	36, 77
labelsep	31, 72, 76, 94
labelwidth	30-35, 76
label	23, 30, 31, 35, 87
lisparindent	77
list-indent	23, 37, 86
list-offset	37
listparindent	37, 94
mark-ans	25, 59, 66
mark-pos	59, 60
mark-ref	25, 59, 65
mini-env	24, 40, 43, 48, 49, 71, 80, 83, 89, 91, 97, 98
mini-sep	24, 40, 80, 83
miniright*	24, 41
miniright	24, 41, 48, 89
minirigth*	27
minirigth	27
no-store	25, 57-59
noitemsep	36, 45
nosep	36, 45
parindent	77
parsep	36, 77, 94
partopsep	36
ref	23, 27, 32-34
resume*	51, 52, 56, 57, 81
resume	23, 51-57, 81, 91
rightmargin	37
save-ans	24, 52-57, 61, 63, 67, 68, 73, 78, 82, 85, 91, 99, 100, 102, 103, 106
save-key	25, 52, 78
save-ref	26, 29, 59, 63, 65, 67, 69, 73, 100
save-sep	59
series	51-56, 78, 81
show-ans	25, 59, 60, 62, 64, 66, 73, 100
show-length	27, 38, 106
show-pos	25, 59, 60, 62, 64, 66, 69, 73, 100
start	24, 27, 35, 36, 52
store-brk	63, 64
topsep	36
widest	23, 27, 35, 36
wrap-ans	59, 62, 66
wrap-label*	31, 71, 74, 93, 94, 99
wrap-label	31, 74, 93, 94, 99
wrap-opt	59

keys commands:

\keys_define:nn	367, 389, 418, 431, 478, 535, 604, 624, 668, 687, 744, 753, 832, 849, 1268, 1379, 1626, 1685, 1759, 1795, 1813, 1931, 2341, 3863, 3926
\l_keys_key_str	4017
\keys_set:nn	381, 856, 1273, 1278, 1520, 1525, 1612, 1620, 1977, 2668, 2672, 2889, 3243, 3644, 3928, 3929, 3930, 3931, 3932, 3933, 3934, 3935, 3936, 3937, 3938, 3939, 3977

keyval commands:

\keyval_parse:NNn	1393
-------------------	------

L

label	429, 476, 535
Labels provide by enumext :	
\Alph*	30, 31
\Roman*	30, 31
\alph*	30, 31
\arabic*	27, 30, 31
\roman*	30, 31
\labelsep	87
\labelsep	3043, 3046
labelsep	365
\labelwidth	31, 87
\labelwidth	3043, 3044
labelwidth	365
\leftmargin	23, 76
\leftmargin	75, 3043
legacy commands:	
\legacy_if:nTF	3385, 3388, 3752, 3755
\legacy_if_gset_false:n	946
\legacy_if_set_false:n	3387, 3754
\legacy_if_set_true:n	3347, 3372, 3379, 3392, 3691, 3719, 3759
\linewidth	80, 83
\linewidth	2756, 2899, 3065, 3092, 3153, 3502, 3563
\list	28
\list	254
list-indent	666
list-offset	666
\listparindent	3045
listparindent	666
\lrbox	3402, 3765

M

\makebox	87
\makebox	1921, 1923, 2485, 3416, 3424, 3428, 3778, 3782
\makelabel	71, 74, 87
\makelabel	74, 2491, 2507
\makesavenoteenv	287
mark-ans	1757
mark-pos	1757, 1793
mark-ref	1757
mini-env	830
mini-sep	830
\minipage	28
\minipage	260
\miniright	10, 48, 1214, 2818, 2952
\miniright*	10
mode commands:	
\mode_if_vertical:TF	901, 929, 1052, 1131
\mode_leave_vertical:	699, 713, 725, 737, 1890, 1898, 1919, 2483, 3414

msg commands:

\msg_error:nn	.. 2880, 2884, 2971, 3030, 3224, 3631, 3637, 3940
\msg_error:nnn	. 1218, 1222, 1247, 1264, 1532, 1536, 3853, 3858, 3923, 3993
\msg_error:nnnn	1640, 1645, 1945, 1949, 1953, 2872, 2967, 2975
\msg_fatal:nn 2648
\msg_fatal:nnn 319
\msg_info:nnn 13, 16, 269, 283
\msg_line_context:	.. 4021, 4026, 4031, 4046, 4077, 4081, 4085, 4090, 4095, 4100, 4105, 4109, 4114, 4119, 4123, 4128, 4132, 4137, 4142, 4147, 4151, 4155
\msg_new:nnn	3994, 3998, 4002, 4006, 4011, 4015, 4019, 4024, 4029, 4044, 4059, 4063, 4067, 4074, 4079, 4083, 4088, 4093, 4098, 4103, 4107, 4112, 4117, 4121, 4126, 4130, 4135, 4140, 4145, 4149, 4153
\msg_note:nnnnn 1652, 1659
\msg_term:nnn 1749
\msg_term:nnnn 2575, 2585, 2614, 2619
\msg_warning:nn 2817, 2951
\msg_warning:nnn 1752
\msg_warning:nnnn	2265, 2523, 2528, 3103, 3116, 3513, 3526
\multicolsep 80, 83
\multicolsep 2786, 2924

N

\NeedsTeXFormat 3
\newcounter 322
\NewDocumentCommand	1214, 1941, 2963, 3837, 3891, 3947
\NewDocumentEnvironment	. 2624, 2850, 3003, 3195, 3604
\newlabel 29
\newlabel 305
no-store 1683
\noindent 90, 97
\noindent	. 2763, 2904, 3162, 3208, 3457, 3572, 3616, 3818
\nointerlineskip 2763, 2904, 3162, 3572
noitemsep 622
\nopagebreak 912, 940, 1063, 1142, 1205, 1211
\normalfont 2134, 2310, 2323, 3738
nosep 622

P

Packages:

enumext 22, 56, 75, 85, 105
enumitem 30
expl3 87
footnotehyper 29
hyperref 25, 26, 29, 65, 69, 93, 94, 105
lua-visual-debug 43
multicol 22, 105
shortlst 87
\par	912, 940, 1063, 1142, 1205, 1211, 1240, 1257, 2113, 2807, 2822, 2941, 2956, 3077, 3180, 3187, 3457, 3471, 3590, 3597, 3818, 3832
\parindent 3434, 3788
\parsep 42, 45, 86, 87
\parsep 1891, 1899, 2605, 3042, 3049, 3054
parsep 622
\parskip 3435, 3789
\partopsep 87
\partopsep 2606, 3047
partopsep 622

peek commands:

\peek_meaning:NTF	3324, 3338, 3355, 3366, 3668, 3682, 3699
\peek_meaning_remove:NTF 3331, 3675
\peek_remove_spaces:n 2469
\phantomsection 29
\phantomsection 294
prg commands:	
\prg_do_nothing: 298
\prg_new_protected_conditional:Npnn	... 196
\prg_replicate:nn 213, 4072
\prg_return_false: 200
\prg_return_true: 199
\printkeyans 14, 103, 3891
prop commands:	
\prop_count:N	1841, 2101, 2137, 2205, 2313, 2326, 3741
\prop_gput_if_not_in:Nnn 1836, 1839
\prop_if_exist:NTF 1670, 3857
\prop_item:Nn 3860
\prop_new:N 1672
\ProvidesExplPackage 4

R

\raggedcolumns 2795, 2930
\ref 65, 67
ref 429, 476, 535
\refstepcounter 3394, 3761
regex commands:	
\regex_match:nnTF	198, 579, 581, 593, 595, 2685, 2698, 3255, 3268
\regex_replace_once:nnN 206
\renewcommand 464, 504, 514, 560
\RenewDocumentCommand	... 2363, 2431, 2465, 2491, 2507
\RequirePackage 17
resume 1377
resume* 1377
rightmargin 666
\Roman 31, 35
\Roman 341
\roman 31, 35
\roman 342, 447, 3879

S

save-ans 1624
save-ref 1757
save-sep 1757
scan commands:	
\scan_stop: 87, 3056, 3207, 3615, 3848, 3851
seq commands:	
\seq_clear:N 3954
\seq_const_from_clist:Nn 3942
\seq_count:N 3016, 3958
\seq_gclear:N 2361, 2362
\seq_gput_right:Nn 1848, 2374, 2375
\seq_if_empty:NTF 2380, 3906, 3972
\seq_if_exist:NTF 1674, 3904
\seq_item:Nn 3074
\seq_map_function:NN 3963
\seq_map_inline:Nn	.. 3912, 3917, 3951, 3973, 3974
\seq_map_pairwise_function:NNN 2382
\seq_new:N 111, 112, 129, 157, 158, 1676
\seq_pop_left:NN 3962
\seq_put_right:Nn 2977, 3970, 3987
\seq_set_from_clist:Nn 3955
\seq_set_map_e:NNn 3964

\seq_show:N	3908
series	1377
\setcounter	590, 594, 596, 2566, 2608, 3021
\setenumext	6–9, 104, 3867, 3872, 3877, 3882, 3887, 3947
\setlength	1892, 1900
show-ans	1757, 1793
show-length	742
show-pos	1793
skip commands:	
\skip_add:Nn	881, 887, 893, 903, 907, 931, 935, 1032, 1038, 1044, 1054, 1058, 1080, 1133, 1137, 3042
\skip_eval:n	1891, 1899
\skip_gset:Nn	1153, 1157, 1161
\skip_gzero_new:N	1148, 1149
\skip_horizontal:N	714, 726, 738, 3417, 3431, 3785
\skip_horizontal:n	700, 1920, 1928, 2484, 2486, 3415, 3793
\skip_if_eq:nnTF	879, 885, 891, 955, 989, 1030, 1036, 1042, 1073, 1078, 1099, 1150, 1172, 1285, 1299, 1313, 1324, 1335, 1346, 1357, 1368
\skip_new:N	69, 70, 74, 75, 76, 77, 78, 133, 176
\skip_set:Nn	864, 868, 917, 921, 958, 962, 966, 973, 977, 981, 992, 997, 1001, 1007, 1012, 1017, 1075, 1076, 1077, 1084, 1088, 1092, 1101, 1106, 1110, 1113, 1117, 1121, 1152, 1156, 1174, 1178, 1182, 1188, 1192, 1196, 3036, 3050
\skip_set_eq:NN	2564, 2604, 2605, 3434, 3435, 3788, 3789
\skip_use:N	866, 870, 905, 909, 913, 933, 937, 956, 975, 984, 990, 995, 999, 1010, 1014, 1015, 1020, 1056, 1060, 1086, 1286, 1290, 1293, 1300, 1304, 1307, 2807
\skip_zero:N	2606, 2786, 2924, 3047, 3048
\skip_zero_new:N	1068, 1069, 1070, 1147, 1169, 1170, 1171
\c_zero_skip	879, 885, 891, 956, 990, 1030, 1036, 1042, 1073, 1078, 1099, 1150, 1172, 1286, 1300, 1313, 1324, 1335, 1346, 1357, 1368
\small	3869, 3874, 3879, 3884, 3889
\star	2345
start	602
\stepcounter	2367, 2984
str commands:	
\c_backslash_str	4081, 4090, 4091, 4095, 4096, 4100, 4101, 4132, 4133, 4137, 4142, 4143
\c_colon_str	2100, 2204, 3848
\str_case:nn	226
\str_case:nnTF	1400, 1409
\str_clear:N	2665, 3242
\str_count:n	213, 4072
\str_if_empty:N	1421, 1462, 1489
\str_if_eq:nnTF	2567, 2610
\str_if_in:nnTF	3844
\str_new:N	128, 171
\str_set:Nn	421, 422, 423, 1774, 1775, 1798, 1799
\string	287
\strutbox	960, 964, 968, 979, 983, 994, 1003, 1009, 1019, 1032, 1038, 1044, 1075, 1076, 1077, 1080, 1090, 1094, 1103, 1110, 1115, 1123, 1152, 1153, 1156, 1163, 1176, 1184, 1190, 1198, 3052

T

TeX and \TeX 2_ε commands:

\@auxout	303
\@currentenv	226
\protected@write	303

text commands:

\text_expand:n	3840
\textasteriskcentered	1771, 1785
\thepage	309
tl commands:	
\c_space_tl	2292, 4031, 4046
\tl_clear:N	394, 400, 1974, 2147, 2221, 3705
\tl_clear_new:N	351
\tl_const:Nn	37, 335
\tl_gclear:N	1442, 1447, 2267, 2502, 2847, 3191, 3418, 3486, 3601
\tl_gclear_new:N	1429
\tl_gput_right:Nn	336
\tl_greplace_all:Nnn	357
\tl_gset:Nn	1430, 1443, 1448, 2255, 2830, 3361
\tl_gset_eq:NN	353, 2412, 3411
\tl_if_blank:nTF	3409
\tl_if_empty:N	452, 471, 523, 529, 567, 1455, 1460, 1482, 1487, 1545, 1609, 1617, 1636, 1855, 1886, 2000, 2158, 2231, 2263, 2286, 2481, 3708, 3985
\tl_if_empty:nTF	1510
\tl_if_exist:N	1515
\tl_if_novalue:nTF	1975, 1986, 2155, 2229, 2271, 2365, 2390, 2408, 2413, 2442, 2663, 3014, 3240, 3642, 3706, 3949
\tl_map_inline:Nn	204, 354
\tl_new:N	34, 39, 40, 43, 44, 50, 52, 53, 54, 56, 57, 90, 91, 92, 98, 99, 100, 102, 103, 104, 105, 106, 108, 109, 115, 116, 126, 127, 138, 148, 149, 150, 153, 170, 173
\tl_put_left:Nn	1863, 1896, 1983, 2298, 2332, 3724, 3727
\tl_put_right:Nn	352, 462, 502, 512, 558, 1817, 1824, 1867, 1902, 1985, 1991, 1999, 2002, 2012, 2017, 2020, 2026, 2052, 2062, 2076, 2092, 2098, 2103, 2150, 2153, 2160, 2162, 2189, 2194, 2199, 2202, 2211, 2224, 2227, 2233, 2235, 2245, 2690, 2703, 3260, 3273, 3710, 3711, 3865, 3870, 3875, 3880, 3885
\tl_remove_all:Nn	3984
\tl_remove_once:Nn	2040, 2174
\tl_replace_all:Nnn	356
\tl_reverse:N	2039, 2041, 2173, 2175
\tl_set:Nn	45, 321, 395, 399, 404, 405, 451, 496, 554, 697, 711, 723, 735, 1544, 1635, 2131, 2273, 2307, 2320, 2410, 3713, 3735, 3982
\tl_set_eq:NN	362, 457, 460, 499, 501, 509, 511, 555, 557, 2038, 2172, 2185, 2454, 2458, 2995, 2997
\tl_to_str:n	1515, 1521, 1526, 3840
\tl_trim_spaces:n	352, 3970, 3982, 3988
\tl_use:N	358, 361, 473, 525, 531, 569, 768, 772, 776, 780, 784, 788, 792, 796, 800, 804, 808, 812, 816, 820, 824, 828, 1925, 2045, 2053, 2064, 2078, 2083, 2095, 2397, 2403, 2427, 2445, 2449, 2457, 2493, 2494, 2501, 2509, 2510, 2516, 2631, 2856, 3000, 3185, 3421, 3432, 3436, 3595, 3775, 3786, 3792, 3796, 3894, 3895, 3896, 3897, 3898, 3966

token commands:

\token_to_str:N	305
\topsep	1892, 1900
topsep	622
\typeout	236, 247, 273, 276, 286, 287, 454, 1713, 1732, 2657, 3234

U

\u	207
use commands:	
\use:N	214, 2498, 2633

\use:n	1391, 3846	1899, 3011, 3022, 3472, 3833
\use_none:nn	297	
\usecounter	2565, 2607	
V		
\value	1458, 1464, 1471, 1477, 1485, 1491, 1498, 1504	
\vspace	947, 1290, 1293, 1304, 1307, 1317, 1319, 1328, 1330, 1339, 1341, 1350, 1352, 1361, 1363, 1372, 1374, 1891,	
W		
widest		602
wrap-ans		1757
wrap-label		365
wrap-label*		365
wrap-opt		1757