

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-15^{*}

©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>	10
1.3.3	Support for <code>minipage</code>	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	11
1.3.5	Support for <code>\footnote</code>	4	4.6.1	The command <code>\anspic</code>	12
2	The environment <code>enumext</code>	4	4.7	Printing stored content	12
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	12
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	12
3	The command <code>\setenumext</code>	5	5	Full examples	13
3.1	Keys for <code>label</code> and <code>ref</code>	6	6	The way of non-enumerated lists	16
3.2	Keys for spaces	6	7	References	18
3.2.1	Vertical spaces	7	8	Change history	18
3.2.2	Horizontal spaces	7	9	Index of Documentation	19
3.3	Keys for <code>add code</code>	8	10	Implementation	21
3.4	Keys for <code>start</code> and <code>resume</code>	8	11	Index of Implementation	107
3.5	Keys for <code>multicols</code>	8			
3.6	Keys for <code>minipage</code>	8			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-05-15.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

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

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

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

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

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

3. Third type of questions

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

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


(A) value

(B) value


(C) value

(D) value


(E) value
4. Question with image and label below:




(A)




(B)



(C)



(D)



(E)

5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value



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

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

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

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

1.1 Description and usage

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

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

```
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.

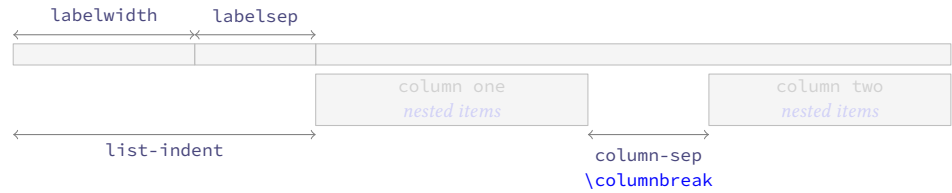


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 `\mini-right` (see §3.6).

1.3.4 The \label and \ref system

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

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

1.3.5 Support for \footnote

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

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

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

2 The environment enumext

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

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

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

Example

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

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

2.1 The `\item*` in `enumext`

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

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

3.1 Keys for label and ref

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

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

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

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

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

- Internally, it renews the command associated with each counter when it is executed, i.e., `\theenumxi` is modified when the key is executed at the first level, `\theenumxii` when it is executed at the second level and `\theenumxiii` together with `\theenumxiv` when it is executed at the third and fourth levels.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

3.4 Keys for start and resume

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

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

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

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

3.5 Keys for multicol

`columns = {⟨integer⟩}` default: `1`

Set the *number of columns* to be used by the `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 the command `\centering`.

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

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

4 The storage system

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

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

4.1 Keys for storage

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

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

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

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

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

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

`save-sep` = {⟨text symbol⟩} default: {, }

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

`mark-ans` = {⟨symbol⟩} default: \textasteriskcentered

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

`mark-pos` = {⟨left | right⟩} default: left

Sets the aligned of the *symbol* defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

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[⟨custom⟩] \item* \item*[⟨content⟩] \end{keyans}`

`keyans*` `\begin{keyans*}[⟨key = val⟩] \item \item[⟨custom⟩] \item* \item*[⟨content⟩] \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[⟨custom⟩]` work in the usual.

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

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

4.5.1 The `\item*` in `keyans`

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

The `\item*` and `\item* [<content>]` command store the current $\langle label \rangle$ set by `label` key next to the $\langle content \rangle$ (if it is present) in $\langle store name \rangle$ set by `save-ans` key in the “first level” of the `enumext` environment. The starred version ‘`*`’ cannot be separated by spaces ‘`␣`’ from the command, i.e. `\item*` and the optional argument does “not support” verbatim content. By design it is assumed that the starred version ‘`*`’ will only appear “once” within the environment.

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

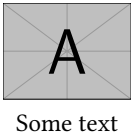
Example

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

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

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

1. Text containing a question.
(A) Choice
* (B) Correct choice
(C) Choice
(D) Choice
2. Text containing a question and image.
(A) Choice
(B) Choice
(C) Choice
(D) Choice
* (E) [note] Correct choice



Some text

4.6 The environment `keyanspic`

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

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

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

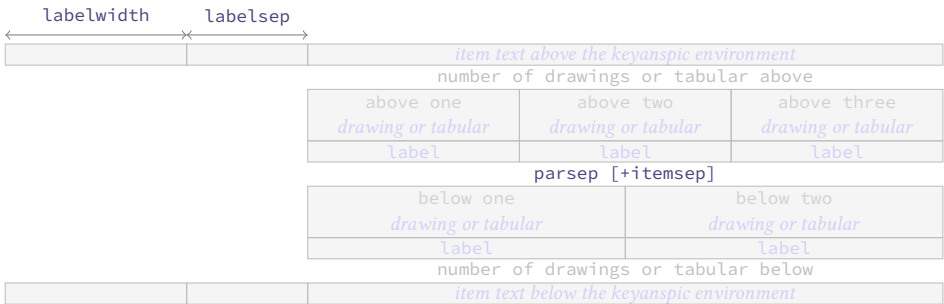


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

4.6.1 The command `\anspic`

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

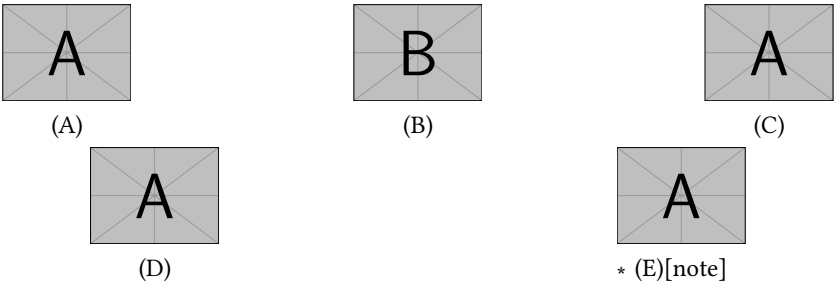
The `\anspic` command take three arguments, the *starred version* ‘`*`’ store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command `\getkeyans`

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

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

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

4.7.2 The command `\printkeyans`

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

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

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

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

Example

```
\begin{enumext}[save-ans=sample,columns=2,show-pos=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 36 km/h.

B 360 km/h.

C 27,8 km/h.

D $3,60 \times 10^8$ km/h.
3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:

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

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

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

B

C

D

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


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

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

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

1. B
2. A
3. B
4. A

Example 2

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

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

B

C

D

36 km/h.

360 km/h.

27,8 km/h.

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

B

C

D

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

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

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

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

B

C

D

36 km/h.

360 km/h.

27,8 km/h.

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

B

C

D

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

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

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

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


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

*

*

*

Example 3

A “simple multiple choice” test .

1. First type of questions
- A

 value

B

 correct

C

 value

D

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

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

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

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

A

 value

B

 value

C

 value
4. Question with image and label below:



A



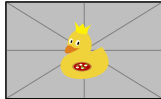
D



B



C



E



5. Question with image on left side:
- A

 value

B

 value

C

 value

D

 correct

E

 value

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 2024-05-15 – 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 9

Commands provide by enumext:

\anskey 4, 10, 11

\anspic* 4, 10–13

\anspic 10, 12, 13

\getkeyans 4, 11, 13

\item* 4–7, 10–12

\item 6, 7, 9–11

\miniright 4, 5, 10

\printkeyans 4, 6, 11, 13

\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, 10

enumext 4–6, 9–14, 17

keyans* 4, 5, 10

keyanspic 4, 7, 10–13, 17

keyans 4–12, 17

Environments:

enumerate 1, 3, 4, 6, 19

list 4, 9, 19

minipage 3–5, 9, 10, 19

multicols 3, 5, 9

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, 9

columns 5, 8, 9

first 9

font 7

item-pos* 6

item-sym* 6

itemindent 8

itemsep 8, 12

labelsep 4, 6–10, 18

labelwidth 4, 6, 7, 9, 10, 18

label 7, 9, 10, 12, 13, 17, 18

list-indent 4, 8, 9

list-offset 4, 8, 18

listparindent 8

mark-ans 10

mark-pos 10

mark-ref 11

mini-env 5, 8, 9

mini-sep 5, 9

miniright* 10

miniright 10

no-store 11

noitemsep 8

nosep 8, 17

parsep 8, 12

partopsep 8

ref 5, 7

resume 9

rightmargin 8

save-ans 5, 9–13

save-ref 5, 7, 11, 13

save-sep 10

show-ans 10, 11

show-length 7

show-pos 10, 11, 13

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 9

\listparindent 8

P

Packages:

enumerate 18

enumext 1–4, 13, 18, 19

enumitem 4, 5, 9, 18, 19

footnotehyper 5

hyperref 5, 11, 19

l3prop 1, 19

l3seq 1, 19

©2024 by Pablo González L

20 / 118

multicol	1, 2, 5, 19	\ref	5
xsim	3	\rightmargin	8
\parsep	8		
\partopsep	8		
R		T	
\raggedcolumns	5	\topsep	8

10 Implementation

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

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

10.1 General conventions

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

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

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

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

10.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

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

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

10.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-15}
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` and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§10.8).

```

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.)

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\l__enumext_resume_name_tl
\l__enumext_resume_name_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` (§10.22). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.27).

```

37 \int_new:N \g__enumext_resume_int
38 \int_new:N \g__enumext_resume_vii_int
39 \tl_new:N \l__enumext_resume_name_tl
40 \bool_new:N \l__enumext_resume_name_bool
41 \tl_new:N \g__enumext_item_symbol_tl
42 \tl_new:N \g__enumext_standar_series_tl
43 \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.

```

44 \dim_new:N \l__enumext_current_widest_dim
45 \tl_new:N \g__enumext_counter_styles_tl
46 \tl_new:N \g__enumext_widest_label_tl
47 \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) which determines the internal values for `\leftmargin` and `\itemindent`.

```

48 \cs_set_protected:Npn \__enumext_tmp:n #1
49 {
50   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
51   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
52   \dim_new:c { l__enumext_leftmargin_#1_dim }
53   \dim_new:c { l__enumext_itemindent_#1_dim }
54 }
55 \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).

```

56 \cs_set_protected:Npn \__enumext_tmp:n #1
57 {
58   \skip_new:c { l__enumext_multicols_above_#1_skip }
59   \skip_new:c { l__enumext_multicols_below_#1_skip }
60 }
61 \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).

```

62 \int_new:N \g__enumext_minipage_stat_int
63 \skip_new:N \l__enumext_minipage_left_skip
64 \skip_new:N \l__enumext_minipage_right_skip
65 \skip_new:N \l__enumext_minipage_after_skip
66 \skip_new:N \g__enumext_minipage_right_skip
67 \skip_new:N \g__enumext_minipage_after_skip
68 \cs_set_protected:Npn \__enumext_tmp:n #1
69 {
70   \dim_new:c { \l__enumext_minipage_left_#1_dim }
71   \bool_new:c { \l__enumext_minipage_active_#1_bool }
72 }
73 \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

```

74 \cs_set_protected:Npn \__enumext_tmp:n #1
75 {
76   \bool_new:c { \l__enumext_wrap_label_#1_bool }
77   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
78   \int_new:c { \l__enumext_start_#1_int }
79   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
80   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
81   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
82   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
83   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
84 }
85 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

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

```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§10.22) activates all the mechanism related to `\anskey`, `keyans`, `keyans*` and `keyanspic`.

The variable `\l__enumext_store_name_tl` sets the name for the storage in `⟨sequence⟩` and `⟨prop list⟩`, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the `check-ans` key (§10.22).

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*` (§10.35.1) for the `keyanspic` environment.

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

```

86 \bool_new:N \l__enumext_store_active_bool
87 \tl_new:N \l__enumext_store_name_tl
88 \tl_new:N \g__enumext_store_name_tl
89 \tl_new:N \l__enumext_store_anskey_arg_tl
90 \int_new:N \l__enumext_store_columns_join_int
91 \tl_new:N \l__enumext_store_keyans_label_tl
92 \tl_new:N \l__enumext_store_keyans_item_opt_tl
93 \tl_new:N \l__enumext_keyans_item_opt_tl
94 \tl_new:N \l__enumext_keyans_tmpa_tl
95 \tl_new:N \l__enumext_keyans_tmpb_tl
96 \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` (§10.40).

```

97 \tl_new:N \l__enumext_setkey_tmpa_tl
98 \tl_new:N \l__enumext_setkey_tmpb_tl
99 \int_new:N \l__enumext_setkey_tmpa_int
100 \seq_new:N \l__enumext_setkey_tmpa_seq
101 \seq_new:N \l__enumext_setkey_tmpb_seq

```

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

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

Internal variables used by [$\langle key = val \rangle$] in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.39) and the keys `columns*` and `columns-sep*`.

```
102 \cs_set_protected:Npn \l__enumext_tmp:n #1
103 {
104   \tl_new:c { \l__enumext_store_opt_#1_tl }
105   \tl_new:c { \l__enumext_print_keyans_#1_tl }
106   \bool_new:c { \l__enumext_store_columns_#1_bool }
107   \int_new:c { \l__enumext_store_columns_#1_int }
108   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
109   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
110   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
111 }
112 \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).

```
113 \bool_new:N \l__enumext_show_answer_bool
114 \bool_new:N \l__enumext_show_position_bool
115 \tl_new:N \l__enumext_mark_ref_sym_tl
116 \tl_new:N \l__enumext_mark_answer_sym_tl
117 \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 (§10.35.2).

```
118 \seq_new:N \l__enumext_keyans_pic_body_seq
119 \dim_new:N \l__enumext_keyans_pic_width_dim
120 \int_new:N \l__enumext_keyans_pic_above_int
121 \int_new:N \l__enumext_keyans_pic_below_int
122 \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.

```
123 \bool_new:N \l__enumext_store_ans_bool
124 \bool_new:N \l__enumext_check_ans_bool
125 \bool_new:N \g__enumext_check_ans_show_bool
126 \bool_new:N \g__enumext_check_ans_show_h_bool
127 \tl_new:N \g__enumext_check_ans_item_tl
128 \int_new:N \g__enumext_count_item_anskey_int
129 \int_new:N \g__enumext_count_item_number_int
130 \int_new:N \g__enumext_standar_star_env_int
131 \int_new:N \g__enumext_starred_star_env_int
132 \int_new:N \g__enumext_starred_keyans_star_env_int
133 \int_new:N \g__enumext_standar_keyans_star_env_int
134 \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`.

```
135 \bool_new:N \l__enumext_hyperref_bool
136 \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.

```
137 \tl_new:N \l__enumext_newlabel_arg_one_tl
138 \tl_new:N \l__enumext_newlabel_arg_two_tl
```

```

139 \tl_new:N \l__enumext_store_write_aux_file_tl
140 \cs_set_protected:Npn \__enumext_tmp:n #1
141 {
142     \tl_new:c { l__enumext_label_copy_#1_tl }
143 }
144 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

`\g__enumext_footnote_int`

Internal variables used for redefinition of `\footnote`.

`\g__enumext_footnote_arg_seq`

```

145 \int_new:N \g__enumext_footnote_int

```

`\g__enumext_footnote_int_seq`

```

146 \seq_new:N \g__enumext_footnote_arg_seq

```

```

147 \seq_new:N \g__enumext_footnote_int_seq

```

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

`\c__enumext_counter_style_tl`

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

`\l__enumext_ref_key_arg_tl`

```

148 \tl_const:Nn \c__enumext_counter_style_tl
149 { { arabic } { roman } { Roman } { alph } { Alph } }

```

`\l__enumext_ref_aux_tl`

```

150 \tl_new:N \l__enumext_ref_key_arg_tl

```

`\l__enumext_the_counter_X_tl`

```

151 \tl_new:N \l__enumext_ref_aux_tl

```

`\l__enumext_counter_style_for_ref_X_tl`

```

152 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

153 {
154     \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
155     \tl_new:c { l__enumext_the_counter_#1_tl }

```

```

156     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
157 }

```

```

158 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

`\l__enumext_item_starred_X_bool`

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

`\l__enumext_item_column_pos_X_int`

```

159 \cs_set_protected:Npn \__enumext_tmp:n #1

```

`\g__enumext_item_count_all_X_int`

```

160 {

```

```

161     \bool_new:c { l__enumext_item_starred_#1_bool }

```

`\l__enumext_joined_item_X_int`

```

162     \int_new:c { l__enumext_item_column_pos_#1_int }

```

`\l__enumext_joined_item_aux_X_int`

```

163     \int_new:c { g__enumext_item_count_all_#1_int }

```

`\l__enumext_tmpa_X_int`

```

164     \int_new:c { l__enumext_joined_item_#1_int }

```

`\l__enumext_item_text_X_box`

```

165     \int_new:c { l__enumext_joined_item_aux_#1_int }

```

`\l__enumext_joined_width_X_dim`

```

166     \int_new:c { l__enumext_tmpa_#1_int }

```

`\l__enumext_item_width_X_dim`

```

167     \box_new:c { l__enumext_item_text_#1_box }

```

`\g__enumext_item_symbol_aux_X_tl`

```

168     \dim_new:c { l__enumext_joined_width_#1_dim }

```

`\l__enumext_align_label_X_str`

```

169     \dim_new:c { l__enumext_item_width_#1_dim }

```

`\g__enumext_minipage_active_X_bool`

```

170     \tl_new:c { g__enumext_item_symbol_aux_#1_tl }

```

`\g__enumext_miniright_code_X_tl`

```

171     \str_new:c { l__enumext_align_label_#1_str }

```

`\g__enumext_minipage_center_X_bool`

```

172     \bool_new:c { g__enumext_minipage_active_#1_bool }

```

`\g__enumext_minipage_right_X_dim`

```

173     \tl_new:c { g__enumext_miniright_code_#1_tl }

```

`\g__enumext_minipage_right_X_skip`

```

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 `miniright` and `miniright*` 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 { \__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_show_length:nnn`

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

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

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
208 \cs_set_protected:Nn \__enumext_zero_count_level:
209 {
210   \cs_set_protected:Npn \__enumext_tmp:n ##1
211   {
212     \int_gzero:c { g__enumext_count_level_##1_int }
213   }
214   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
215 }
```

(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 76) and to the `__enumext_safe_exec_vii:` function in the definition of the `enumext*` environment (pag 88).

```
216 \cs_new_protected:Nn \__enumext_current_env_set_bool:
217 {
```

```

218 \str_case:en { \@currentvir }
219 {
220   {enumext}
221   {
222     \bool_lazy_and:nnT
223     { \bool_not_p:n { \g__enumext_standar_bool } }
224     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
225     {
226       \bool_gset_true:N \g__enumext_standar_bool
227       \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
228       \typeout{working-on-enumext}
229     }
230   }
231   {enumext*}
232   {
233     \bool_lazy_and:nnT
234     { \bool_not_p:n { \g__enumext_starred_bool } }
235     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
236     {
237       \bool_gset_true:N \g__enumext_starred_bool
238       \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
239       \typeout{working-on-enumext*}
240     }
241   }
242 }
243 }

```

(End of definition for `__enumext_current_env_set_bool:.`)

10.6 Copying list and minipage environments

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

```

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

```

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

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w

```

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

```

244 \__enumext_at_begin_document:n
245 {
246   \cs_new_eq:NN \__enumext_start_list:nn \list
247   \cs_new_eq:NN \__enumext_stop_list: \endlist
248   \cs_new_eq:NN \__enumext_item_std:w \item
249 }

```

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

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

```

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

```

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

```

\__enumext_minipage:w
\__enumext_endminipage:

```

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

```

250 \__enumext_at_begin_document:n
251 {
252   \cs_new_eq:NN \__enumext_minipage:w \minipage
253   \cs_new_eq:NN \__enumext_endminipage: \endminipage
254 }

```

(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.

```
255 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
256 \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 `__enumext_footnotes_key_bool` to “true”.

```
257 \cs_new_protected:Nn \__enumext_after_hyperref:
258 {
259   \IfPackageLoadedTF { hyperref }
260   {
261     \msg_info:nnn { enumext } { package-load } { hyperref }
262     \bool_set_true:N \l__enumext_hyperref_bool
263     \IfHyperBoolean{hyperfootnotes}
264     {
265       \typeout{hyperfootnotes=true}
266       \bool_set_true:N \l__enumext_footnotes_key_bool
267     }
268     { \typeout{hyperfootnotes=false} }
269   }
270   { }
```

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`.

```
271 \bool_if:NT \l__enumext_footnotes_key_bool
272 {
273   \IfPackageLoadedTF { footnotehyper }
274   {
275     \msg_info:nnn { enumext } { package-load } { footnotehyper }
276   }
277   {
278     \typeout{No ~ footnotehyper ~ load}
279     \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
280     \bool_set_false:N \l__enumext_footnotes_key_bool
281   }
282 }
```

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.

```
283 \bool_if:NTF \l__enumext_hyperref_bool
284 {
285   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
286   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
287 }
288 {
289   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
290   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
291 }
292 }
```

(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.

```
293 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
294 {
295   \protected@write \@auxout { }
296   {
297     \token_to_str:N \newlabel {#1}
```

```

298         {
299             {#2}
300             \bool_if:NT \l__enumext_hyperref_bool
301             { { \thepage } {#2} {#1} }
302             { }
303         }
304     }
305     \__enumext_hypertarget:nn {#1} { }
306     \__enumext_phantomsection:
307 }

```

(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.

```

308 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
309 {
310     \cs_if_exist:cTF { c@ #2 }
311     { \msg_fatal:nnn { enumext } { counters } { #2 } }
312     {
313         \tl_set:Nn #1 { #2 }
314         \newcounter { #2 }
315     }
316 }

```

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

The counters created here are `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment, `enumXvi` for `keyanspic` environment, `enumXvii` for `enumext*` and `enumXviii` for the `keyans*` environments.

```

enumXi      317 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     318 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    319 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv     320 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXv      321 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXvii    322 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXviii   323 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
            324 \__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.

```

325 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
326 {
327     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
328     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
329 }
330 \__enumext_register_counter_style:Nn \arabic { 0 }
331 \__enumext_register_counter_style:Nn \Alph { M }
332 \__enumext_register_counter_style:Nn \alph { m }
333 \__enumext_register_counter_style:Nn \Roman { VIII }
334 \__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.

```

335 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
336 {
337   \hbox_set:Nn \__enumext_label_width_by_box {#2}
338   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
339 }
340 \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.

```

341 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
342 {
343   \tl_clear_new:N #1
344   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
345   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
346   \tl_map_inline:Nn \g__enumext_counter_styles_tl
347   {
348     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
349     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
350     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
351   }
352   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
353   { \tl_use:N \g__enumext_widest_label_tl }
354   \tl_set_eq:cN { the #2 } #1
355 }
356 \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.

```

357 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
358 {
359   \keys_define:nn { enumext / #1 }
360   {
361     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
362     font      .value_required:n = true,
363     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
364     labelsep  .initial:n   = {0.3333em},
365     labelsep  .value_required:n = true,
366     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
367     labelwidth .value_required:n = true,
368     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
369     wrap-label .initial:n   = {##1},
370     wrap-label .value_required:n = true,
371     wrap-label* .code:n = {
372       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
373       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
374     },
375     wrap-label* .value_required:n = true,
376   }
377 }
378 \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.

```

379 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
380 {

```

```

381 \keys_define:nn { enumext / #1 }
382 {
383   align .choice:,
384   align / left .code:n =
385     {
386       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
387       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
388     },
389   align / right .code:n =
390     {
391       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
392       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
393     },
394   align / center .code:n =
395     {
396       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
397       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
398     },
399   align .initial:n = left,
400   align .value_required:n = true,
401 }
402 }
403 \clist_map_inline:nn
404 {
405   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
406 }
407 { \__enumext_tmp:nn #1 }

```

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

```

408 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
409 {
410   \keys_define:nn { enumext / #1 }
411   {
412     align .choice:,
413     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
414     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
415     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
416     align .initial:n = left,
417     align .value_required:n = true,
418   }
419 }
420 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

10.11 Setting label and ref keys

`__enumext_regex_label_ref_key:`

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

It loops through the defined counter styles in `\c__enumext_counter_style_tl` and replace `*` by real command, for example, looking for `\arabic*` and replacing that by `\arabic{<counter>}` defined on the current level.

```

421 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
422 {
423   \tl_map_inline:Nn \c__enumext_counter_style_tl
424   {
425     \regex_replace_once:nnN { \c{##1}\* }
426     { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
427   }
428 }

```

(End of definition for `__enumext_regex_label_ref_key:`.)

`__enumext_set_label_ref:n`

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

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

```

429 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
430 {
431   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}

```



```

432 \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_counter_ \__enumext_level: _tl }
433 \__enumext_regex_label_ref_key:
434 \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
435 \tl_put_right:ce { \l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
436 {
437   \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
438   { \exp_not:V \l__enumext_ref_key_arg_tl }
439 }
440 }

```

(End of definition for `__enumext_set_label_ref:n`)

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

```

441 \cs_new_protected:Nn \__enumext_use_key_ref:
442 {
443   \tl_if_empty:cF { \l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
444   {
445     \tl_use:c { \l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
446   }
447 }

```

(End of definition for `__enumext_use_key_ref:`)

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

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

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

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

```

448 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
449 {
450   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
451   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
452   {
453     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
454     \__enumext_regex_label_ref_key:
455     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
456     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
457     {
458       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
459       { \exp_not:V \l__enumext_ref_key_arg_tl }
460     }
461   }
462   {
463     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
464     \__enumext_regex_label_ref_key:
465     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
466     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
467     {
468       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
469       { \exp_not:V \l__enumext_ref_key_arg_tl }
470     }
471   }
472 }

```

(End of definition for `__enumext_set_label_ref_h:n`)

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

```

473 \cs_new_protected:Nn \__enumext_use_key_ref_h:
474 {
475   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
476   {
477     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
478     {
479       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl

```

```

480     }
481   }
482   {
483     \tl_if_empty:NF \__enumext_counter_style_for_ref_viii_tl
484     {
485       \tl_use:N \__enumext_counter_style_for_ref_viii_tl
486     }
487   }
488 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

Here we set the default $\langle labels \rangle$ of the four levels of `enumext` environment, along with the default value for `labelwidth` key.

```

\__enumext_label_i_tl 489 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_ii_tl 490 {
\__enumext_label_iii_tl 491   \keys_define:nn { enumext / #1 }
\__enumext_label_iv_tl 492   {
493     label .code:n = {
494       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
495       { \__enumext_counter_#2_tl } {##1}
496       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
497       \__enumext_current_widest_dim
498     },
499     label .initial:n = #3,
500     label .value_required:n = true,
501     ref .code:n = \__enumext_set_label_ref:n {##1},
502     ref .value_required:n = true,
503   }
504 }
505 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
506 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
507 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
508 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

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

```

\__enumext_label_vii_tl 509 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_viii_tl 510 {
511   \keys_define:nn { enumext / #1 }
512   {
513     label .code:n = {
514       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
515       { \__enumext_counter_#2_tl } {##1}
516       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
517       \__enumext_current_widest_dim
518     },
519     label .initial:n = #3,
520     label .value_required:n = true,
521     ref .code:n = \__enumext_set_label_ref_h:n {##1},
522     ref .value_required:n = true,
523   }
524 }
525 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
526 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

10.11.3 Define and set label key for keyans and keyanspic environment

Here we set the default $\langle label \rangle$ for `keyans` and `keyanspic` environment, along with the default value for `labelwidth`. The `keyanspic` environment use the same $\langle label \rangle$ as the `keyans` environment.

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

```

527 \keys_define:nn { enumext / keyans }
528 {
529   label .code:n = {
530     \__enumext_label_style:cvn { \__enumext_label_v_tl }
531     { \__enumext_counter_v_tl } {##1}

```

```

532         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
533         \l__enumext_current_widest_dim
534         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
535         { \l__enumext_counter_vi_tl } {#1}
536         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
537         \l__enumext_current_widest_dim
538     },
539     label .initial:n = (\Alph*),
540     label .value_required:n = true,
541 }

```

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

10.12 Setting start and widest keys

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

```

#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3: <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.

```

542 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
543 {
544     \__enumext_if_is_int:nTF { #3 }
545     {
546         \int_set:Nn #2 {#3}
547     }
548     {
549         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
550         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
551         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
552         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
553     }
554 }
555 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

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.

```

556 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
557 {
558     \__enumext_if_is_int:nTF {#4}
559     {
560         \setcounter{enumX#1} { #4 }
561     }
562     {
563         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
564         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
565         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
566         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
567     }
568     \__enumext_label_width_by_box:cv
569     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
570 }
571 \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.

```

572 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
573 {

```

```

574 \keys_define:nn { enumext / #1 }
575 {
576   start .code:n = {
577     \__enumext_start_from:ccn
578     { l__enumext_label_#2_tl }
579     { l__enumext_start_#2_int } {##1}
580   },
581   start .initial:n = 1,
582   widest .code:n = {
583     \__enumext_widest_from:nccn {#2}
584     { l__enumext_label_#2_tl }
585     { l__enumext_labelwidth_#2_dim } {##1}
586   },
587   widest .value_required:n = true,
588   start .value_required:n = true,
589 }
590 }
591 \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 592 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
partopsep 593 {
parsep 594 \keys_define:nn { enumext / #1 }
noitemsep 595 {
nosep 596   topsep .skip_set:c = { l__enumext_topsep_#2_skip },
597   topsep .initial:n = {#3},
598   topsep .value_required:n = true,
599   partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
600   partopsep .initial:n = {#4},
601   partopsep .value_required:n = true,
602   parsep .skip_set:c = { l__enumext_parsep_#2_skip },
603   parsep .initial:n = {#5},
604   parsep .value_required:n = true,
605   itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
606   itemsep .initial:n = {#6},
607   itemsep .value_required:n = true,
608   noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
609   noitemsep .value_forbidden:n = true,
610   nosep .meta:n = {
611     itemsep = 0pt, parsep = 0pt,
612     topsep = 0pt, partopsep = 0pt,
613   },
614   nosep .value_forbidden:n = true,
615 }
616 }

```

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

```

617 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
618 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
619 { 4.0pt plus 2.0pt minus 1.0pt }
620 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
621 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
622 { 2.0pt plus 1.0pt minus 1.0pt }
623 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
624 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
625 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
626 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
627 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
628 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
629 { 2.0pt plus 1.0pt minus 1.0pt }
630 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
631 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
632 { 4.0pt plus 2.0pt minus 1.0pt }
633 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
634 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
635 { 2.0pt plus 1.0pt minus 1.0pt }

```

(End of definition for topsep and others.)

10.14 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```

636 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
637 {
638   \keys_define:nn { enumext / #1 }
639   {
640     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
641     itemindent .value_required:n = true,
642     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
643     rightmargin .value_required:n = true,
644     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
645     listparindent .value_required:n = true,
646     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
647     list-offset .value_required:n = true,
648     list-indent .code:n =
649       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
650       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
651     list-indent .value_required:n = true,
652   }
653 }
654 \clist_map_inline:Nn \__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.

```

655 \cs_set_protected:Npn \__enumext_tmp:n #1
656 {
657   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
658 }
659 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {##1} }

```

10.14.1 Functions for setting the fake itemindent

__enumext_fake_item:
__enumext_keyans_fake_item:
__enumext_fake_item_vii:
__enumext_fake_item_viii:

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.

```

660 \cs_set_protected:Nn \__enumext_fake_item:
661 {
662   \dim_compare:nNnT
663     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
664     >
665     { \c_zero_dim }
666   {
667     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
668     {
669       \exp_not:N \mode_leave_vertical:
670       \exp_not:n { \skip_horizontal:n }
671       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
672       \ignorespaces
673     }
674   }
675 }
676 \cs_set_protected:Nn \__enumext_keyans_fake_item:
677 {
678   \dim_compare:nNnT
679     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
680   {
681     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
682     {
683       \exp_not:N \mode_leave_vertical:
684       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
685     }
686   }
687 }
688 \cs_set_protected:Nn \__enumext_fake_item_vii:
689 {
690   \dim_compare:nNnT
691     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }

```

```

692     {
693         \tl_set:Nc \__enumext_fake_item_indent_vii_tl
694         {
695             \exp_not:N \mode_leave_vertical:
696             \exp_not:N \skip_horizontal:N \__enumext_fake_item_indent_vii_dim
697         }
698     }
699 }
700 \cs_set_protected:Nn \__enumext_fake_item_viii:
701 {
702     \dim_compare:nNt
703     { \__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
704     {
705         \tl_set:Nc \__enumext_fake_item_indent_viii_tl
706         {
707             \exp_not:N \mode_leave_vertical:
708             \exp_not:N \skip_horizontal:N \__enumext_fake_item_indent_viii_dim
709         }
710     }
711 }

```

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

10.15 Setting show-length key

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

```

712 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
713 {
714     \keys_define:nn { enumext / #1 }
715     {
716         show-length .bool_set:c = { \__enumext_show_length_#2_bool },
717         show-length .initial:n = false,
718     }
719 }
720 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

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

```

before*
after
first
721 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
722 {
723     \keys_define:nn { enumext / #1 }
724     {
725         before .tl_set:c = { \__enumext_before_no_starred_key_#2_tl },
726         before .value_required:n = true,
727         before* .tl_set:c = { \__enumext_before_starred_key_#2_tl },
728         before* .value_required:n = true,
729         after .tl_set:c = { \__enumext_after_stop_list_#2_tl },
730         after .value_required:n = true,
731         first .tl_set:c = { \__enumext_after_list_args_#2_tl },
732         first .value_required:n = true,
733     }
734 }
735 \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.

```

\__enumext_before_keys_exec:
\__enumext_after_stop_list:
\__enumext_after_args_exec:
736 \cs_new_protected:Nn \__enumext_before_args_exec:
737 {
738     \tl_use:c { \__enumext_before_starred_key_ \__enumext_level: _tl }
739 }

```

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⟩`.

```
740 \cs_new_protected:Nn \__enumext_before_keys_exec:
741 {
742   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
743 }
```

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

```
744 \cs_new_protected:Nn \__enumext_after_stop_list:
745 {
746   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
747 }
```

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`.

```
748 \cs_new_protected:Nn \__enumext_after_args_exec:
749 {
750   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
751 }
```

(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.

```
752 \cs_new_protected:Nn \__enumext_before_args_exec_v:
753 {
754   \tl_use:N \l__enumext_before_starred_key_v_tl
755 }
```

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⟩`.

```
756 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
757 {
758   \tl_use:N \l__enumext_before_no_starred_key_v_tl
759 }
```

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

```
760 \cs_new_protected:Nn \__enumext_after_stop_list_v:
761 {
762   \tl_use:N \l__enumext_after_stop_list_v_tl
763 }
```

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`.

```
764 \cs_new_protected:Nn \__enumext_after_args_exec_v:
765 {
766   \tl_use:N \l__enumext_after_list_args_v_tl
767 }
```

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

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

`__enumext_before_args_exec_vii`: The function `__enumext_before_args_exec_v`: executes the `{⟨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.

```
768 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
769 {
770   \tl_use:N \l__enumext_before_starred_key_vii_tl
771 }
772 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
773 {
774   \tl_use:N \l__enumext_before_starred_key_viii_tl
775 }
```


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`.

```

776 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
777 {
778   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
779 }
780 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
781 {
782   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
783 }

```

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

```

784 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
785 {
786   \tl_use:N \l__enumext_after_stop_list_vii_tl
787 }
788 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
789 {
790   \tl_use:N \l__enumext_after_stop_list_viii_tl
791 }

```

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`.

```

792 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
793 {
794   \tl_use:N \l__enumext_after_list_args_vii_tl
795 }
796 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
797 {
798   \tl_use:N \l__enumext_after_list_args_viii_tl
799 }

```

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

10.17 Setting keys for multicols and minipage

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

```

800 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
801 {
802   \keys_define:nn { enumext / #1 }
803   {
804     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
805     mini-env .value_required:n = true,
806     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
807     mini-sep .initial:n = 0.3333em,
808     mini-sep .value_required:n = true,
809     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
810     columns-sep .value_required:n = true,
811     columns .int_set:c = { l__enumext_columns_#2_int },
812     columns .initial:n = 1,
813     columns .value_required:n = true,
814   }
815 }
816 \clist_map_inline:Nn \__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`.

```

817 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
818 {
819   \keys_define:nn { enumext / #1 }
820   {
821     columns .initial:n = 2,
822     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
823     miniright .value_required:n = true,
824     miniright* .code:n = {

```

```

825         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
826         \keys_set:nn { enumext / #1 } { miniright = {##1} }
827     },
828     miniright* .value_required:n = true,
829 }
830 }
831 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for multicol

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

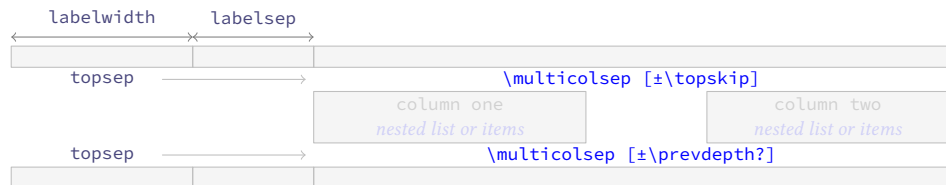


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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

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

10.18.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

832 \cs_new_protected:Nn \__enumext_multi_set_vskip:
833 {
834     \skip_set:cn { l__enumext_multicol_above_ \__enumext_level: _skip }
835     {
836         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
837     }
838     \skip_set:cn { l__enumext_multicol_below_ \__enumext_level: _skip }
839     {
840         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
841     }
842     \__enumext_add_pre_parsep:
843 }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

844 \cs_new_protected:Nn \__enumext_add_pre_parsep:
845 {
846     \int_case:nn { \l__enumext_level_int }
847     {
848         { 2 }{
849             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
850             {
851                 \skip_add:Nn \l__enumext_multicol_above_ii_skip { \l__enumext_parsep_i_skip }
852             }
853         }
854         { 3 }{
855             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
856             {

```

```

857             \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_iii_skip
858         }
859     }
860     { 4 }{
861         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
862         {
863             \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
864         }
865     }
866 }
867 }

```

(End of definition for `__enumext_add_pre_parsep:`)

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

```

868 \cs_new_protected:Nn \__enumext_multi_addvspace:
869 {
870     \__enumext_multi_set_vskip:
871     \mode_if_vertical:T
872     {
873         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
874         {
875             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
876         }
877         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
878         {
879             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
880         }
881     }
882     \par\nopagebreak
883     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
884 }

```

(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:`

```

885 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
886 {
887     \skip_set:Nn \l__enumext_multicols_above_v_skip
888     {
889         \l__enumext_topsep_v_skip
890     }
891     \skip_set:Nn \l__enumext_multicols_below_v_skip
892     {
893         \l__enumext_topsep_v_skip
894     }
895 }
896 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
897 {
898     \__enumext_keyans_multi_set_vskip:
899     \mode_if_vertical:T
900     {
901         \skip_add:Nn \l__enumext_multicols_above_v_skip
902         {
903             \skip_use:N \l__enumext_partopsep_v_skip
904         }
905         \skip_add:Nn \l__enumext_multicols_below_v_skip
906         {
907             \skip_use:N \l__enumext_partopsep_v_skip
908         }
909     }
910     \par\nopagebreak
911     \addvspace{ \l__enumext_multicols_above_v_skip }
912 }

```

(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.

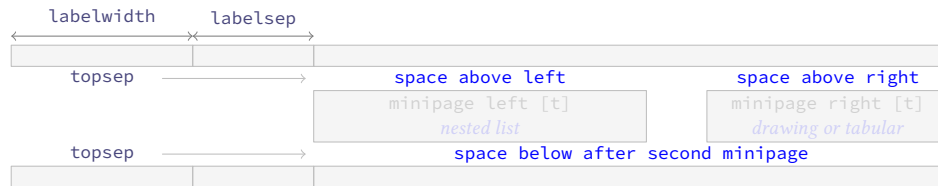


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```

913 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
914 {
915     \__enumext_minipage:w [ t ] { #1 }
916     \legacy_if_gset_false:n { @minipage }
917     \vspace { 0pt }
918 }
919 { \__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`.

```

920 \cs_new_protected:Nn \__enumext_mini_set_vskip:
921 {
922     \int_compare:nNnTF
923     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
924     {
```

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.

```

925     \skip_if_eq:nnTF
926     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
927     {
928         \skip_set:Nn \l__enumext_minipage_left_skip
929         {
930             -0.150\box_dp:N \strutbox
931         }
932         \skip_set:Nn \l__enumext_minipage_right_skip
933         {
934             0.695\box_dp:N \strutbox
935         }
936     }
```

```

936     \skip_set:Nn \l__enumext_minipage_after_skip
937     {
938         \box_dp:N \strutbox
939     }
940     \__enumext_zero_parsep:
941 }
942 {
943     \skip_set:Nn \l__enumext_minipage_left_skip
944     {
945         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
946     }
947     \skip_set:Nn \l__enumext_minipage_right_skip
948     {
949         0.695\box_dp:N \strutbox
950     }
951     \skip_set:Nn \l__enumext_minipage_after_skip
952     {
953         1.85\box_dp:N \strutbox
954         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
955     }
956 }
957 }
958 {

```

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.

```

959     \skip_if_eq:nnTF
960     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
961     {
962         \skip_set:Nn \l__enumext_minipage_left_skip
963         {
964             0.5\box_dp:N \strutbox
965             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
966         }
967         \skip_set:Nn \l__enumext_minipage_right_skip
968         {
969             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
970         }
971         \skip_set:Nn \l__enumext_minipage_after_skip
972         {
973             1.6\box_dp:N \strutbox
974         }
975     }
976     {
977         \skip_set:Nn \l__enumext_minipage_left_skip
978         {
979             0.5875\box_dp:N \strutbox
980             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
981         }
982         \skip_set:Nn \l__enumext_minipage_right_skip
983         {
984             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
985             + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
986         }
987         \skip_set:Nn \l__enumext_minipage_after_skip
988         {
989             0.325\box_dp:N \strutbox
990             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
991         }
992     }
993 }
994 }

```

(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.

```

995 \cs_new_protected:Nn \__enumext_zero_parsep:
996 {

```

```

997 \int_case:nn { \l__enumext_level_int }
998 {
999   { 2 }{
1000     \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1001     {
1002       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1003     }
1004   }
1005   { 3 }{
1006     \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1007     {
1008       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1009     }
1010   }
1011   { 4 }{
1012     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1013     {
1014       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1015     }
1016   }
1017 }
1018 }

```

(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*.

```

1019 \cs_new_protected:Nn \__enumext_mini_addvspace:
1020 {
1021   \__enumext_mini_set_vskip:
1022   \mode_if_vertical:T
1023   {
1024     \skip_add:Nn \l__enumext_minipage_left_skip
1025     {
1026       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1027     }
1028     \skip_add:Nn \l__enumext_minipage_after_skip
1029     {
1030       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1031     }
1032   }
1033   \par\nopagebreak
1034   \addvspace { \l__enumext_minipage_left_skip }
1035 }

```

(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`.

```

1036 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1037 {
1038   \skip_zero_new:N \l__enumext_minipage_after_skip
1039   \skip_zero_new:N \l__enumext_minipage_left_skip
1040   \skip_zero_new:N \l__enumext_minipage_right_skip
1041   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1042   {
1043     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1044     {
1045       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1046       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1047       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1048       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1049       {
1050         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1051       }
1052     }
1053   }

```

```

1053     {
1054         \skip_set:Nn \l__enumext_minipage_left_skip
1055         {
1056             \skip_use:N \l__enumext_topsep_v_skip
1057         }
1058         \skip_set:Nn \l__enumext_minipage_right_skip
1059         {
1060             0.705\box_dp:N \strutbox
1061         }
1062         \skip_set:Nn \l__enumext_minipage_after_skip
1063         {
1064             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1065         }
1066     }
1067 }
1068 {
1069     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1070     {
1071         \skip_set:Nn \l__enumext_minipage_left_skip
1072         {
1073             0.5\box_dp:N \strutbox
1074             + \l__enumext_partopsep_v_skip
1075         }
1076         \skip_set:Nn \l__enumext_minipage_right_skip
1077         {
1078             \l__enumext_partopsep_v_skip
1079         }
1080         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1081     }
1082     {
1083         \skip_set:Nn \l__enumext_minipage_left_skip
1084         {
1085             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1086         }
1087         \skip_set:Nn \l__enumext_minipage_right_skip
1088         {
1089             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1090         }
1091         \skip_set:Nn \l__enumext_minipage_after_skip
1092         {
1093             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1094         }
1095     }
1096 }
1097 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

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

```

1098 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1099 {
1100     \__enumext_keyans_mini_set_vskip:
1101     \mode_if_vertical:T
1102     {
1103         \skip_add:Nn \l__enumext_minipage_left_skip
1104         {
1105             \l__enumext_partopsep_v_skip
1106         }
1107         \skip_add:Nn \l__enumext_minipage_after_skip
1108         {
1109             \l__enumext_partopsep_v_skip
1110         }
1111     }
1112     \par\nopagebreak
1113     \addvspace { \l__enumext_minipage_left_skip }
1114 }

```

(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*`.

```

1115 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1116 {
1117   \skip_zero_new:N \l__enumext_minipage_left_skip
1118   \skip_gzero_new:N \g__enumext_minipage_right_skip
1119   \skip_gzero_new:N \g__enumext_minipage_after_skip
1120   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1121   {
1122     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1123     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1124   }
1125   {
1126     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1127     \skip_gset:Nn \g__enumext_minipage_right_skip
1128     {
1129       \l__enumext_topsep_vii_skip
1130     }
1131     \skip_gset:Nn \g__enumext_minipage_after_skip
1132     {
1133       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1134     }
1135   }
1136 }
1137 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1138 {
1139   \skip_zero_new:N \l__enumext_minipage_after_skip
1140   \skip_zero_new:N \l__enumext_minipage_left_skip
1141   \skip_zero_new:N \l__enumext_minipage_right_skip
1142   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1143   {
1144     \skip_set:Nn \l__enumext_minipage_left_skip
1145     {
1146       0.5\box_dp:N \strutbox
1147     }
1148     \skip_set:Nn \l__enumext_minipage_right_skip
1149     {
1150       \l__enumext_partopsep_viii_skip
1151     }
1152     \skip_set:Nn \l__enumext_minipage_after_skip
1153     {
1154       1.6\box_dp:N \strutbox
1155     }
1156   }
1157   {
1158     \skip_set:Nn \l__enumext_minipage_left_skip
1159     {
1160       0.5875\box_dp:N \strutbox
1161     }
1162     \skip_set:Nn \l__enumext_minipage_right_skip
1163     {
1164       \l__enumext_topsep_viii_skip
1165     }
1166     \skip_set:Nn \l__enumext_minipage_after_skip
1167     {
1168       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1169     }
1170   }
1171 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext-mini_env*` environment on the *left side* when the `miniright` key is active in the `enumext*` and `keyans*` environments.

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

```

1172 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:

```

```

1173 {
1174   \__enumext_mini_set_vskip_vii:
1175   \par\nopagebreak
1176   \addvspace { \__enumext_minipage_left_skip }
1177 }
1178 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1179 {
1180   \__enumext_mini_set_vskip_viii:
1181   \par\nopagebreak
1182   \addvspace { \__enumext_minipage_left_skip }
1183 }

```

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

10.19.4 The command \miniright

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

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

```

1184 \NewDocumentCommand \miniright { s }
1185 {
1186   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1187   {
1188     \msg_error:nnn { enumext } { wrong-miniright-place }
1189   }
1190   \int_compare:nNnT { \__enumext_level_int } = { 0 }
1191   {
1192     \msg_error:nnn { enumext } { wrong-miniright-place }
1193   }
1194   \int_compare:nNnTF { \__enumext_keyans_level_int } = { 1 }
1195   {
1196     \__enumext_keyans_mini_right_cmd:n {#1}
1197   }
1198   { \__enumext_mini_right_cmd:n {#1} }
1199 }

```

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

`__enumext_mini_right_cmd:n` The function `__enumext_mini_right_cmd:n` takes as argument the *starred version* ‘*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `__enumext_minipage_right_X_dim`, if so we close the `\multicols` environment with the `__enumext_mini_env*` environment on the “left side”, then we open the `__enumext_mini_env*` environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the `\centering` command when the starred argument ‘*’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1200 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1201 {
1202   \dim_compare:nNnTF
1203   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1204   {
1205     \__enumext_multicols_stop:
1206     \end{\__enumext_mini_env*}
1207     \hfill
1208     \begin{\__enumext_mini_env*}
1209     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1210     \par\addvspace { \__enumext_minipage_right_skip }
1211     \bool_if:nF {#1}
1212     {
1213       \centering
1214     }
1215     \int_gzero:N \g__enumext_minipage_stat_int
1216   }
1217   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1218 }

```

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

`__enumext_keyans_mini_right_cmd:n`

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

```

1219 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1220 {
1221   \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1222   {
1223     \__enumext_keyans_multicols_stop:
1224     \end{__enumext_mini_env*}
1225     \hfill
1226     \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1227     \par\addvspace { \__enumext_minipage_right_skip }
1228     \bool_if:nF {#1}
1229     {
1230       \centering
1231     }
1232     \int_gzero:N \g__enumext_minipage_stat_int
1233   }
1234   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1235 }

```

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

10.20 Setting above and below keys

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

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

`above`
`above*`
`below`
`below*`

```

1236 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1237 {
1238   \keys_define:nn { enumext / #1 }
1239   {
1240     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1241     above .value_required:n = true,
1242     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1243                       \keys_set:nn { enumext / #1 } { above = {##1} },
1244     above* .value_required:n = true,
1245     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1246     below .value_required:n = true,
1247     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1248                       \keys_set:nn { enumext / #1 } { below = {##1} },
1249     below* .value_required:n = true,
1250   }
1251 }
1252 \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.

```

1253 \cs_new_protected:Nn \__enumext_vspace_above:
1254 {
1255   \skip_if_eq:nnF
1256   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1257   {
1258     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1259     {
1260       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1261     }
1262     {
1263       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1264     }
1265   }
1266 }

```

(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.

```

1267 \cs_new_protected:Nn \__enumext_vspace_below:
1268 {
1269   \skip_if_eq:nnF
1270   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1271   {
1272     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1273     {
1274       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1275     }
1276     {
1277       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1278     }
1279   }
1280 }

```

(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.

```

1281 \cs_new_protected:Nn \__enumext_vspace_above_v:
1282 {
1283   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1284   {
1285     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1286     {
1287       \vspace*{ \__enumext_vspace_above_v_skip }
1288     }
1289     { \vspace { \__enumext_vspace_above_v_skip } }
1290   }
1291 }

```

(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.

```

1292 \cs_new_protected:Nn \__enumext_vspace_below_v:
1293 {
1294   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1295   {
1296     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1297     {
1298       \vspace*{ \__enumext_vspace_below_v_skip }
1299     }
1300     { \vspace { \__enumext_vspace_below_v_skip } }
1301   }
1302 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

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

`__enumext_vspace_above_viii:`

```

1303 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1304 {
1305   \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1306   {
1307     \bool_if:NTF \__enumext_vspace_a_star_vii_bool
1308     {
1309       \vspace*{ \__enumext_vspace_above_vii_skip }
1310     }
1311     { \vspace { \__enumext_vspace_above_vii_skip } }
1312   }
1313 }
1314 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1315 {
1316   \skip_if_eq:nnF { \__enumext_vspace_above_viii_skip } { \c_zero_skip }

```

```

1317     {
1318         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1319         {
1320             \vspace*{ \l__enumext_vspace_above_viii_skip }
1321         }
1322         { \vspace { \l__enumext_vspace_above_viii_skip } }
1323     }
1324 }

```

(End of definition for \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii:.)

The functions \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii: apply the vertical space below the `enumext*` and `keyans*` environments set by the `below*` and `below` keys.

```

1325 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1326 {
1327     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1328     {
1329         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1330         {
1331             \vspace*{ \l__enumext_vspace_below_vii_skip }
1332         }
1333         { \vspace { \l__enumext_vspace_below_vii_skip } }
1334     }
1335 }
1336 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1337 {
1338     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1339     {
1340         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1341         {
1342             \vspace*{ \l__enumext_vspace_below_viii_skip }
1343         }
1344         { \vspace { \l__enumext_vspace_below_viii_skip } }
1345     }
1346 }

```

(End of definition for \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii:.)

10.21 Setting save-ans key

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

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

```

1347 \keys_define:nn { enumext / level-1 }
1348 {
1349     save-ans .code:n = \l__enumext_storing_set:n {#1},
1350     save-ans .value_required:n = true,
1351 }
1352 \keys_define:nn { enumext / enumext* }
1353 {
1354     save-ans .code:n = \l__enumext_storing_set_vii:n {#1},
1355     save-ans .value_required:n = true,
1356 }

```

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

10.21.1 Internal functions for `save-ans` key

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

The boolean var `\l__enumext_store_active_bool` will be set to true activating the entire internal storage mechanism, then the integer variable for the `resume` key will be created (if not exist).

```

1357 \cs_new_protected:Npn \l__enumext_storing_set:n #1
1358 {
1359     \tl_set:Ne \l__enumext_store_name_tl {#1}
1360     \tl_if_empty:NTF \l__enumext_store_name_tl
1361     {
1362         \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1363     }

```

```

1364     {
1365         \__enumext_storing_standar:
1366     }
1367 }
1368 \cs_new_protected:Npn \__enumext_storing_set_vii:n #1
1369 {
1370     \tl_set:Nx \l__enumext_store_name_tl {#1}
1371     \tl_if_empty:NTF \l__enumext_store_name_tl
1372     {
1373         \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1374     }
1375     {
1376         \__enumext_storing_starred:
1377     }
1378 }
1379 \cs_new_protected:Nn \__enumext_storing_standar:
1380 {
1381     \bool_if:NTF \l__enumext_standar_level_one_bool
1382     {
1383         \__enumext_storing_exec:
1384     }
1385     {
1386         \msg_warning:nnn { enumext } { save-ans-nested } { enumext }
1387     }
1388 }
1389 \cs_new_protected:Nn \__enumext_storing_starred:
1390 {
1391     \bool_if:NTF \l__enumext_starred_level_one_bool
1392     {
1393         \__enumext_storing_exec:
1394     }
1395     {
1396         \msg_warning:nnn { enumext } { save-ans-nested } { enumext* }
1397     }
1398 }
1399 \cs_new_protected:Nn \__enumext_storing_exec:
1400 {
1401     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1402     {
1403         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1404     }
1405     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1406     {
1407         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1408     }
1409     \bool_set_true:N \l__enumext_store_active_bool
1410     \bool_set_true:N \l__enumext_store_ans_bool
1411     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1412     {
1413         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1414     }
1415 }

```

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

10.22 Setting series 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, but, discarding some specific $\langle keys \rangle$.

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

```

resume
resume*
1416 \cs_set_protected:Npn \__enumext_tmp:n #1
1417 {
1418     \keys_define:nn { enumext / #1 }
1419     {
1420         series .str_set:N = \l__enumext_series_str,
1421         series .value_required:n = true,
1422         resume .code:n = \__enumext_resume_series:n {##1},
1423         resume* .code:n = \__enumext_resume_starred:,
1424         resume* .value_forbidden:n = true,
1425     }

```

```

1426     }
1427     \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.22.1 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.

```

1428 \cs_new:Npn \__enumext_filter_series:n #1
1429 {
1430   \use:e
1431   {
1432     \keyval_parse:NNn
1433     \__enumext_filter_series_key:n
1434     \__enumext_filter_series_pair:nn {#1}
1435   }
1436 }

```

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.

```

1437 \cs_new:Npn \__enumext_filter_series_key:n #1
1438 {
1439   \str_case:nnF {#1}
1440   {
1441     { resume } {}
1442     { resume* } {}
1443   }
1444   { , { \exp_not:n {#1} } }
1445 }

```

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

```

1446 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1447 {
1448   \str_case:nnF {#1}
1449   {
1450     { series } {}
1451     { resume } {}
1452     { save-key } {}
1453     { save-ans } {}
1454   }
1455   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1456 }

```

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

```

  \__enumext_parse_series_name:n
  \__enumext_resume_last:n

```

The function `__enumext_parse_series_name:n` will be in charge of saving the filtered *⟨keys⟩* in a global variable `\g__enumext_series_⟨series name⟩_tl` created globally when using the key `series`, otherwise it will call the function `__enumext_resume_last:n`. This function is passed to the function `__enumext_parse_keys_parse_keys:n` in the `enumext` environment definition (§10.33) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§10.36).

```

1457 \cs_new_protected:Npn \__enumext_parse_series_name:n #1
1458 {
1459   \str_if_empty:NTF \l__enumext_series_str
1460   {
1461     \bool_if:NF \l__enumext_resume_name_bool
1462     {
1463       \__enumext_resume_last:n {#1}
1464     }
1465   }
1466   {
1467     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1468     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1469     { \__enumext_filter_series:n {#1} }
1470     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1471     {
1472       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1473     }
1474   }
1475 }

```


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.

```

1476 \cs_new_protected:Npn \__enumext_resume_last:n #1
1477 {
1478   \bool_if:NT \l__enumext_standar_level_one_bool
1479   {
1480     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT]]}
1481     \tl_gclear:N \g__enumext_standar_series_tl
1482     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1483   }
1484   \bool_if:NT \l__enumext_starred_level_one_bool
1485   {
1486     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT*]]}
1487     \tl_gclear:N \g__enumext_starred_series_tl
1488     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1489   }
1490 }

```

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

10.22.2 Internal function for resume and resume* keys

The keys `resume` without assigned value and `resume*` reset the *counter* of the list according to the last value of the counter of the previous list, the first one only the *counter* and the second one with the optional values filtered from the last non-nested list in which the key `series` is not present. When assigning value to `resume={⟨series name⟩}` it will use the previous values of the list in which the `series={⟨series name⟩}` key was executed.

`__enumext_resume_series:n`

The function `__enumext_resume_series:n` will handle the argument passed to the `resume` key in the `enumext` environment. 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` environment in which the `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.

```

1491 \cs_new_protected:Npn \__enumext_resume_series:n #1
1492 {
1493   \tl_if_empty:nTF {#1}
1494   {
1495     \__enumext_resume_counter:n { }
1496   }
1497   {
1498     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1499     {
1500       \__enumext_resume_counter:n {#1}
1501       \bool_if:NT \g__enumext_standar_bool
1502       {
1503         \keys_set:nv { enumext / level-1 }
1504         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1505       }
1506       \bool_if:NT \g__enumext_starred_bool
1507       {
1508         \keys_set:nv { enumext / enumext* }
1509         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1510       }
1511     }
1512     { \msg_error:nnn { enumext } { unknown-series } {#1} }
1513   }
1514 }

```

(End of definition for `__enumext_resume_series:n`)

`__enumext_resume_counter:n`

```

\__enumext_resume_counter:
  \__enumext_resume_counter_name:
  \__enumext_resume_counter_store:
1515 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1516 {
1517   \bool_set_true:N \l__enumext_resume_name_bool
1518   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1519   \tl_if_empty:NTF \l__enumext_resume_name_tl

```

```

1520     {
1521         \__enumext_resume_counter:
1522     }
1523     {
1524         \__enumext_resume_counter_name:
1525     }
1526     \__enumext_resume_counter_store:
1527 }

1528 \cs_new_protected:Nn \__enumext_resume_counter:
1529 {
1530     \bool_if:NT \g__enumext_standar_bool
1531     {
1532         \int_gincr:N \g__enumext_resume_int
1533         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1534     }
1535     \bool_if:NT \g__enumext_starred_bool
1536     {
1537         \int_gincr:N \g__enumext_resume_vii_int
1538         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1539     }
1540 }

1541 \cs_new_protected:Nn \__enumext_resume_counter_name:
1542 {
1543     \bool_if:NT \g__enumext_standar_bool
1544     {
1545         \int_set:Nn \l__enumext_start_i_int
1546         {
1547             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1548         }
1549     }
1550     \bool_if:NT \g__enumext_starred_bool
1551     {
1552         \int_set:Nn \l__enumext_start_vii_int
1553         {
1554             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1555         }
1556     }
1557 }

1558 \cs_new_protected:Nn \__enumext_resume_counter_store:
1559 {
1560     \bool_lazy_and:nnT
1561     { \bool_if_p:N \l__enumext_standar_level_one_bool }
1562     { \bool_if_p:N \l__enumext_store_active_bool }
1563     {
1564         \int_set:Nn \l__enumext_start_i_int
1565         {
1566             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1567         }
1568     }
1569     \bool_lazy_and:nnT
1570     { \bool_if_p:N \l__enumext_starred_level_one_bool }
1571     { \bool_if_p:N \l__enumext_store_active_bool }
1572     {
1573         \int_set:Nn \l__enumext_start_vii_int
1574         {
1575             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1576         }
1577     }
1578 }

```

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

__enumext_resume_starred:

```

1579 \cs_new_protected:Nn \__enumext_resume_starred:
1580 {
1581     \bool_if:NT \g__enumext_standar_bool
1582     {
1583         \tl_if_empty:NF \g__enumext_standar_series_tl
1584         {

```

```

1585         \__enumext_resume_counter:n { }
1586         \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1587     }
1588 }
1589 \bool_if:NT \g__enumext_starred_bool
1590 {
1591     \tl_if_empty:NF \g__enumext_starred_series_tl
1592     {
1593         \__enumext_resume_counter:n { }
1594         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1595     }
1596 }
1597 }

```

(End of definition for __enumext_resume_starred:.)

__enumext_resume_counter_set:

```

1598 \cs_new_protected:Nn \__enumext_resume_counter_set:
1599 {
1600     \bool_if:NT \g__enumext_standar_bool
1601     {
1602         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1603         {
1604             \int_gset_eq:cN { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1605         }
1606         \tl_if_empty:NF \l__enumext_series_str
1607         {
1608             \int_gset_eq:cN { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1609         }
1610         \tl_if_empty:NTF \l__enumext_resume_name_tl
1611         {
1612             \str_if_empty:NT \l__enumext_series_str
1613             {
1614                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1615             }
1616         }
1617         {
1618             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1619             {
1620                 \int_gset_eq:cN { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1621             }
1622         }
1623     }
1624     \bool_if:NT \g__enumext_starred_bool
1625     {
1626         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1627         {
1628             \int_gset_eq:cN { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1629         }
1630         \tl_if_empty:NF \l__enumext_series_str
1631         {
1632             \int_gset_eq:cN { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1633         }
1634         \tl_if_empty:NTF \l__enumext_resume_name_tl
1635         {
1636             \str_if_empty:NT \l__enumext_series_str
1637             {
1638                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1639             }
1640         }
1641         {
1642             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1643             {
1644                 \int_gset_eq:cN { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1645             }
1646         }
1647     }
1648 }

```

(End of definition for __enumext_resume_counter_set:.)

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.

```

1649 \cs_set_protected:Npn \__enumext_tmp:n #1
1650 {
1651   \keys_define:nn { enumext / #1 }
1652   {
1653     check-ans .bool_set:N = \__enumext_check_ans_bool,
1654     check-ans .initial:n = false,
1655     no-store .code:n = {
1656       \bool_set_false:N \__enumext_store_ans_bool
1657       \bool_set_false:N \__enumext_check_ans_bool
1658     },
1659     no-store .value_forbidden:n = true,
1660   }
1661 }
1662 \clist_map_inline:nn
1663 {
1664   level-1, level-2, level-3, level-4, enumext*
1665 }
1666 { \__enumext_tmp:n {#1} }
```

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

10.23.2 Set-up check answer mechanism

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.

```

1667 \cs_new_protected:Nn \__enumext_check_ans_set:
1668 {
1669   \int_case:nn { \__enumext_level_int }
1670   {
1671     { 1 }{
1672       \bool_lazy_all:nT
1673       {
1674         { \bool_if_p:N \g__enumext_starred_bool }
1675         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1676       }

```

```

1677         {
1678             \int_gdecr:N \g__enumext_count_item_number_int
1679             \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1680         }
1681     }
1682     { 2 }{
1683         \int_gdecr:N \g__enumext_count_item_number_int
1684     }
1685     { 3 }{
1686         \int_gdecr:N \g__enumext_count_item_number_int
1687     }
1688     { 4 }{
1689         \int_gdecr:N \g__enumext_count_item_number_int
1690     }
1691 }
1692 \int_case:nn { \l__enumext_level_h_int }
1693 {
1694     { 1 }{
1695         \bool_if:NT \g__enumext_standar_bool
1696         {
1697             \int_gdecr:N \g__enumext_count_item_number_int
1698             \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1699         }
1700     }
1701 }
1702 }

```

(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.

```

1703 \cs_new_protected:Nn \__enumext_check_ans_exec:
1704 {
1705     \bool_if:NT \l__enumext_check_ans_bool
1706     {
1707         \__enumext_check_ans_set:
1708     }
1709 }

```

(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.

```

1710 \cs_new_protected:Nn \__enumext_check_ans_show:
1711 {
1712     \int_compare:nNnTF
1713     { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1714     {
1715         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1716     }
1717     {
1718         \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1719     }
1720     \int_gzero:N \g__enumext_count_item_number_int
1721     \int_gzero:N \g__enumext_count_item_anskey_int
1722 }

```

(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*.

```

1723 \cs_set_protected:Npn \__enumext_tmp:n #1
1724 {
1725     \keys_define:nn { enumext / #1 }
1726     {
1727         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,

```

```

1728     wrap-ans     .initial:n = \fbox{##1},
1729     wrap-ans     .value_required:n = true,
1730     wrap-opt     .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1731     wrap-opt     .initial:n = [{##1}],
1732     wrap-opt     .value_required:n = true,
1733     save-sep     .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1734     save-sep     .initial:n = {, ~ },
1735     save-sep     .value_required:n = true,
1736     mark-ans     .tl_set:N = \__enumext_mark_answer_sym_tl,
1737     mark-ans     .initial:n = \textasteriskcentered,
1738     mark-ans     .value_required:n = true,
1739     mark-pos     .choice:,
1740     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1741     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1742     mark-pos     .initial:n = right,
1743     mark-pos     .value_required:n = true,
1744     show-ans     .bool_set:N = \__enumext_show_answer_bool,
1745     show-ans     .initial:n = false,
1746     show-ans     .value_required:n = true,
1747     show-pos     .bool_set:N = \__enumext_show_position_bool,
1748     show-pos     .initial:n = false,
1749     show-pos     .value_required:n = true,
1750     mark-ref     .tl_set:N = \__enumext_mark_ref_sym_tl,
1751     mark-ref     .initial:n = \textasteriskcentered,
1752     mark-ref     .value_required:n = true,
1753     save-ref     .bool_set:N = \__enumext_store_ref_key_bool,
1754     save-ref     .initial:n = false,
1755     save-ref     .value_required:n = true,
1756   }
1757 }
1758 \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`.

```

1759 \cs_set_protected:Npn \__enumext_tmp:n #1
1760 {
1761   \keys_define:nn { enumext / #1 }
1762   {
1763     mark-pos .choice:,
1764     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1765     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1766     mark-pos .initial:n = right,
1767     mark-pos .value_required:n = true,
1768     show-ans .bool_set:N = \__enumext_show_answer_bool,
1769     show-ans .initial:n = false,
1770     show-ans .value_required:n = true,
1771     show-pos .bool_set:N = \__enumext_show_position_bool,
1772     show-pos .initial:n = false,
1773     show-pos .value_required:n = true,
1774   }
1775 }
1776 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1777 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1778 {
1779   \keys_define:nn { enumext / #1 }
1780   {
1781     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1782               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1783               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1784               {
1785                 columns = \exp_not:v { l__enumext_store_columns_#2_int },
1786               },

```

```

1787         columns*      .value_required:n = true,
1788         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1789                     \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1790                     \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1791                     {
1792                         columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim }
1793                     },
1794         columns-sep* .value_required:n = true,
1795     }
1796 }
1797 \clist_map_inline:nn
1798 {
1799     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1800 }
1801 { \__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.

```

1802 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1803 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1804 {
1805     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1806     {
1807         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1808     }
1809     { #1 }
1810 }
1811 \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.

```

1812 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1813 {
1814     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1815 }
1816 \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 `opt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```

1817 \cs_new_protected:Nn \__enumext_store_level_open:
1818 {
1819     \bool_if:NT \l__enumext_store_ans_bool
1820     {
1821         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1822         {
1823             \__enumext_store_addto_seq:n
1824             {
1825                 \item \begin{enumext}
1826             }

```



```

1827     }
1828     {
1829         \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1830         {
1831             \item \begin{enumext} [
1832             ]
1833             \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1834             {
1835             ]
1836             }
1837         \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1838     }
1839 }
1840 }
1841 \cs_new_protected:Nn \__enumext_store_level_close:
1842 {
1843     \bool_if:NT \l__enumext_store_ans_bool
1844     {
1845         \__enumext_store_addto_seq:n { \end{enumext} }
1846     }
1847 }

```

(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.

```

1848 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1849 {
1850     \bool_if:NT \l__enumext_store_ans_bool
1851     {
1852         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1853         {
1854             \__enumext_store_addto_seq:n
1855             {
1856                 \item \mode_leave_vertical:
1857                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1858                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1859             }
1860         }
1861         {
1862             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1863             {
1864                 \item \mode_leave_vertical:
1865                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1866                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1867                 ]
1868             }
1869             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1870             {
1871             ]
1872             }
1873             \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1874         }
1875     }
1876 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1877 {
1878     \bool_if:NT \l__enumext_store_ans_bool
1879     {
1880         \__enumext_store_addto_seq:n { \end{enumext*} }
1881     }
1882 }

```

(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

1883 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1884 {
1885   \mode_leave_vertical:
1886   \skip_horizontal:n { -\dim_use:N #2 }
1887   \makebox[0pt][ r ]
1888   {
1889     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1890     {
1891       \tl_use:N \l__enumext_mark_answer_sym_tl
1892     }
1893   }
1894   \skip_horizontal:n { \dim_use:N #2 }
1895 }
1896 \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.

```

1897 \keys_define:nn { enumext / anskey }
1898 {
1899   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1900   item-sym* .value_required:n = true,
1901   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1902   item-pos* .value_required:n = true,
1903   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1904   store-brk .default:n = true,
1905   store-brk .value_forbidden:n = true,
1906 }

```

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⟩*.

```

1907 \NewDocumentCommand \anskey { d() s o +m }
1908 {
1909   \bool_if:NF \l__enumext_store_active_bool
1910   {
1911     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1912   }
1913   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
1914   {
1915     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1916   }
1917   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1918   {
1919     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1920   }
1921   \group_begin:
1922     \bool_if:NT \l__enumext_store_ans_bool
1923     {
1924       \bool_if:NT \l__enumext_check_ans_bool
1925       {
1926         \int_gincr:N \g__enumext_count_item_anskey_int
1927       }
1928       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1929     }
1930   \group_end:
1931 }

```

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

__enumext_store_anskey_code:nnnn

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

```

1932 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1933 {
1934   \__enumext_store_addto_prop:n {#4}
1935   \bool_if:NT \l__enumext_store_ref_key_bool
1936   {
1937     \__enumext_store_internal_ref:
1938   }
1939   \__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*.

```

1940   \tl_clear:N \l__enumext_store_anskey_arg_tl
1941   \tl_if_novalue:nF {#3}
1942   {
1943     \keys_set:nn { enumext / anskey } {#3}
1944   }
1945   \bool_lazy_and:nnT
1946   { \bool_if_p:N \l__enumext_store_columns_break_bool }
1947   { \bool_not_p:n { \l__enumext_starred_bool } }
1948   {
1949     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1950   }
1951   \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).

```

1952   \tl_if_novalue:nF {#1}
1953   {
1954     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1955     \bool_if:NT \l__enumext_starred_bool
1956     {
1957       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1958       {
1959         ( \exp_not:V \l__enumext_store_columns_join_int )
1960       }
1961     }
1962   }

```

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.

```

1963   \bool_if:nTF {#2}
1964   {
1965     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1966     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1967     {
1968       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1969       {
1970         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1971       }
1972     }
1973     \dim_compare:nT
1974     {
1975       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1976     }
1977     {
1978       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1979       {
1980         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1981       }
1982     }
1983     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1984   }
1985   {

```

```

1986     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1987   }

```

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`.

```

1988   \bool_lazy_and:nnT
1989   { \bool_if_p:N \l__enumext_store_ref_key_bool }
1990   { \bool_if_p:N \l__enumext_hyperref_bool }
1991   {
1992     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1993     {
1994       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1995       { \exp_not:V \l__enumext_mark_ref_sym_tl }
1996     }
1997   }
1998   \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1999 }

```

(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`.

```

2000 \cs_new_protected:Nn \__enumext_store_internal_ref:
2001 {
2002   \cs_set_protected:Npn \__enumext_tmp:n ##1
2003   {
2004     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2005     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2006     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2007     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2008   }
2009   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2010   \cs_set:Npn \__enumext_tmp:n ##1
2011   { . \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.

```

2012   \bool_lazy_all:nT
2013   {
2014     { \bool_if_p:N \g__enumext_starred_bool }
2015     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2016   }
2017   {
2018     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2019     { \tl_use:N \l__enumext_label_copy_vii_tl }
2020   }
2021   \bool_lazy_all:nT
2022   {
2023     { \bool_if_p:N \l__enumext_standar_bool }
2024     { \bool_if_p:N \g__enumext_starred_bool }
2025     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2026   }
2027   {
2028     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2029     {
2030       \tl_use:N \l__enumext_label_copy_vii_tl
2031       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2032     }
2033   }

```

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.

```

2034   \bool_lazy_all:nT
2035   {
2036     { \bool_if_p:N \l__enumext_standar_bool }
2037     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2038     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2039     { \bool_not_p:n { \l__enumext_starred_bool } }
2040   }

```

```

2041     {
2042         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2043         {
2044             \tl_use:N \l__enumext_label_copy_i_tl
2045             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2046         }
2047     }
2048     \cs_set:Npn \l__enumext_tmp:n ##1
2049     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2050     \bool_lazy_all:nT
2051     {
2052         { \bool_if_p:N \l__enumext_standar_bool }
2053         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2054         { \bool_not_p:n { \g__enumext_starred_bool } }
2055         { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2056     }
2057     {
2058         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2059         {
2060             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2061             . \tl_use:N \l__enumext_label_copy_vii_tl
2062         }
2063     }

```

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

```

2064     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2065     {
2066         \l__enumext_store_name_tl \c_colon_str
2067         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2068     }

```

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

```

2069     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2070     {
2071         \l__enumext_newlabel:nn
2072         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2073         { \l__enumext_newlabel_arg_two_tl }
2074     }
2075     \l__enumext_store_write_aux_file_tl
2076 }

```

(End of definition for `\l__enumext_store_internal_ref:`)

`\l__enumext_store_anskey_show_wrap:n`

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

```

2077 \cs_new_protected:Npn \l__enumext_store_anskey_show_wrap:n #1
2078 {
2079     \par
2080     \bool_if:NT \l__enumext_starred_bool
2081     {
2082         \cs_set:Nn \l__enumext_level: { vii }
2083     }
2084     \l__enumext_print_keyans_box:cc
2085     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2086     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2087     \l__enumext_anskey_wrapper:n { #1 }
2088 }

```

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

`\l__enumext_store_anskey_show_left:n`

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

```

2089 \cs_new_protected:Npn \l__enumext_store_anskey_show_left:n #1
2090 {
2091     \bool_if:NT \l__enumext_show_answer_bool
2092     {
2093         \l__enumext_store_anskey_show_wrap:n { #1 }

```

```

2094     }
2095     \bool_if:NT \l__enumext_show_position_bool
2096     {
2097         \tl_set:Nx \l__enumext_mark_answer_sym_tl
2098         {
2099             \group_begin:
2100             \exp_not:N \normalfont
2101             \exp_not:N \footnotesize [ \int_eval:n
2102             {
2103                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2104             }
2105             ]
2106             \group_end:
2107         }
2108         \__enumext_store_anskey_show_wrap:n { #1 }
2109     }
2110 }

```

(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`.

```

2111 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2112 {
2113     \tl_clear:N \l__enumext_store_keyans_label_tl
2114     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2115     {
2116         \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2117     }
2118     {
2119         \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2120     }
2121     \tl_if_novalue:nF { #1 }
2122     {
2123         % Set save-sep
2124         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2125         {
2126             \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2127         }
2128         \tl_put_right:Nx \l__enumext_store_keyans_label_tl { #1 }
2129     }
2130     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2131 }

```

(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.

```

2132 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2133 {
2134     \bool_if:NT \l__enumext_store_ref_key_bool
2135     {
2136         \cs_set_protected:Npn \__enumext_tmp:n #1
2137         {
2138             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2139             \tl_reverse:c { \l__enumext_label_copy_##1_tl }

```

```

2140         \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2141         \tl_reverse:c { \__enumext_label_copy_##1_tl }
2142     }
2143     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2144     \__enumext_keyans_store_ref_aux_i:
2145 }
2146 }

```

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

```

2147 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2148 {
2149     \bool_if:NT \g__enumext_starred_bool
2150     {
2151         \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2152     }
2153     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2154     {
2155         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2156         { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2157     }
2158     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2159     {
2160         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2161         { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2162     }
2163     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2164     {
2165         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2166         { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
2167     }
2168     \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
2169     {
2170         \__enumext_store_name_tl \c_colon_str
2171         \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
2172     }
2173     \__enumext_keyans_store_ref_aux_ii:
2174 }

```

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

```

2175 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2176 {
2177     \tl_put_right:Ne \__enumext_store_write_aux_file_tl
2178     {
2179         \__enumext_newlabel:nn
2180         { \exp_not:V \__enumext_newlabel_arg_one_tl }
2181         { \__enumext_newlabel_arg_two_tl }
2182     }
2183     \__enumext_store_write_aux_file_tl
2184 }

```

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

10.26.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

2185 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2186 {
2187     \tl_clear:N \__enumext_store_keyans_label_tl
2188     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2189     {
2190         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_vi_tl }
2191     }
2192     {
2193         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_v_tl }

```



```

2194     }
2195     \tl_if_novalue:nF { #1 }
2196     {
2197         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2198         {
2199             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2200         }
2201         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2202     }
2203     \__enumext_keyans_addto_seq_link:
2204 }

```

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.

```

2205 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2206 {
2207     \bool_lazy_and:nnT
2208     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2209     { \bool_if_p:N \l__enumext_hyperref_bool }
2210     {
2211         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2212         {
2213             \hfill \exp_not:N \hyperlink
2214             {
2215                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2216             }
2217             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2218         }
2219     }
2220     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2221     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2222     \bool_if:NT \l__enumext_check_ans_bool
2223     {
2224         \int_gincr:N \g__enumext_count_item_anskey_int
2225     }
2226 }

```

(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.

```

2227 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2228 {
2229     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2230     {
2231         \msg_warning:nnnn { enumext } { missing-starred } { #1 } { #2 }
2232     }
2233     { \tl_gclear:N \g__enumext_check_ans_item_tl }
2234 }

```

(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_` `\l__enumext_store_name_tl` `_prop` for `show-pos` key.

```

2235 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2236 {
2237     \tl_if_novalue:nF { #1 }
2238     {
2239         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2240     }

```

```

2241 \bool_if:NT \l__enumext_show_answer_bool
2242 {
2243   \__enumext_keyans_show_ans:
2244 }
2245 \bool_if:NT \l__enumext_show_position_bool
2246 {
2247   \__enumext_keyans_show_pos:
2248 }
2249 }
2250 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2251 {
2252   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2253   {
2254     \bool_lazy_or:nnT
2255     { \bool_if_p:N \l__enumext_show_answer_bool }
2256     { \bool_if_p:N \l__enumext_show_position_bool }
2257     {
2258       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2259     }
2260   }
2261 }
2262 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2263 {
2264   \tl_put_left:Nn \l__enumext_label_v_tl
2265   {
2266     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2267   }
2268 }
2269 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2270 {
2271   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2272   {
2273     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2274     {
2275       \group_begin:
2276       \exp_not:N \normalfont
2277       \exp_not:N \footnotesize [ \int_eval:n
2278         {
2279           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2280         }
2281       ]
2282       \group_end:
2283     }
2284   }
2285   {
2286     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2287     {
2288       \group_begin:
2289       \exp_not:N \normalfont
2290       \exp_not:N \footnotesize [ \int_eval:n
2291         {
2292           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2293         }
2294       ]
2295       \group_end:
2296     }
2297   }
2298   \tl_put_left:Nn \l__enumext_label_v_tl
2299   {
2300     \__enumext_print_keyans_box:NN
2301     \l__enumext_labelwidth_i_dim
2302     \l__enumext_labelsep_i_dim
2303   }
2304 }

```

(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*
2305 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2306 {
2307   \keys_define:nn { enumext / #1 }
2308   {
2309     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2310     item-sym* .value_required:n = true,
2311     item-sym* .initial:n = {$\star$},
2312     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2313     item-pos* .value_required:n = true,
2314   }
2315 }
2316 \clist_map_inline:nn
2317 {
2318   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2319 }
2320 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining \footnote command

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.

```

2321 \cs_new_protected:Nn \__enumext_footnotetext:nn
2322 {
2323   \footnotetext[#1]{#2}
2324 }
2325 \cs_new_protected:Nn \__enumext_renew_footnote:
2326 {
2327   \seq_gclear:N \g__enumext_footnote_arg_seq
2328   \seq_gclear:N \g__enumext_footnote_int_seq
2329   \RenewDocumentCommand \footnote { o +m }
2330   {
2331     \tl_if_novalue:nTF {##1}
2332     {
2333       \stepcounter{footnote}
2334       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2335     }
2336     {
2337       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2338     }
2339     \footnotemark [ \g__enumext_footnote_int ]
2340     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2341     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2342   }
2343 }
2344 \cs_new_protected:Nn \__enumext_print_footnote:
2345 {
2346   \seq_if_empty:NF \g__enumext_footnote_int_seq
2347   {
2348     \seq_map_pairwise_function:NNN
2349     \g__enumext_footnote_int_seq
2350     \g__enumext_footnote_arg_seq
2351     \__enumext_footnotetext:nn
2352   }
2353 }

```

(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

The \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)`.

```

2354 \cs_new_protected:Npn \__enumext_default_item:n #1
2355 {
2356   \tl_if_novalue:nTF {#1}
2357   {
2358     \bool_if:NT \l__enumext_check_ans_bool
2359     {
2360       \int_gincr:N \g__enumext_count_item_number_int
2361     }
2362     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2363     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2364   }
2365   {
2366     \bool_set_eq:cc
2367     { l__enumext_wrap_label_ \__enumext_level: _bool }
2368     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2369     \__enumext_item_std:w {#1} \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2370   }
2371 }

```

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

`__enumext_starred_item:nn`

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

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

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

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

```

2372 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2373 {
2374   \tl_if_novalue:nF {#1}
2375   {
2376     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2377   }
2378   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2379   \tl_if_novalue:nTF {#2}
2380   {
2381     \dim_set_eq:cc
2382     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2383     { l__enumext_labelsep_ \__enumext_level: _dim }
2384   }
2385   {
2386     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2387   }
2388   \bool_if:NT \l__enumext_check_ans_bool
2389   {
2390     \int_gincr:N \g__enumext_count_item_number_int
2391   }
2392   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2393   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2394 }

```

(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 (§10.32).

```

2395 \cs_new_protected:Nn \__enumext_redefine_item:
2396 {
2397   \RenewDocumentCommand \item { s o o }
2398   {
2399     \bool_if:nTF {##1}
2400     {
2401       \__enumext_starred_item:nn {##2} {##3}
2402     }
2403     { \__enumext_default_item:n {##2} }
2404   }
2405 }

```

(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`.

```

2406 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2407 {
2408   \tl_if_novalue:nTF { #1 }
2409   {
2410     \bool_set_true:N \__enumext_wrap_label_v_bool
2411     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2412   }
2413   {
2414     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2415     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2416   }
2417 }

```

(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 $\langle label \rangle$.

```

2418 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2419 {
2420   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2421   \__enumext_keyans_show_left:n { #1 }
2422   \bool_set_true:N \__enumext_wrap_label_v_bool
2423   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

2424   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2425   \__enumext_keyans_addto_prop:n { #1 }
2426   \__enumext_keyans_store_ref:
2427   \__enumext_keyans_addto_seq:n { #1 }
2428 }

```

(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 (§10.32).

```

2429 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2430 {

```

```

2431 \RenewDocumentCommand \item { s o }
2432 {
2433   \bool_if:nTF {##1}
2434   {
2435     \peek_remove_spaces:n
2436     {
2437       \__enumext_keyans_starred_item:n {##2}
2438     }
2439   }
2440   {
2441     \__enumext_keyans_default_item:n {##2}
2442   }
2443 }
2444 }

```

(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.

```

2445 \cs_new_protected:Nn \__enumext_item_starred:
2446 {
2447   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2448   {
2449     \mode_leave_vertical:
2450     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2451     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2452     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2453   }
2454 }

```

(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 (§10.32).

```

2455 \cs_new_protected:Nn \__enumext_make_label:
2456 {
2457   \RenewDocumentCommand \makelabel { m }
2458   {
2459     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2460     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2461     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2462     {
2463       \__enumext_item_starred:
2464       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2465     }
2466     { ##1 }
2467     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2468     \tl_gclear:N \g__enumext_item_symbol_tl
2469   }
2470 }

```

(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 (§10.32).

```

2471 \cs_new_protected:Nn \__enumext_keyans_make_label:
2472 {
2473   \RenewDocumentCommand \makelabel { m }
2474   {
2475     \tl_use:N \l__enumext_label_fill_left_v_tl
2476     \tl_use:N \l__enumext_label_font_style_v_tl
2477     \bool_if:NTF \l__enumext_wrap_label_v_bool

```

```

2478     {
2479         \__enumext_wrapper_label_v:n { ##1 }
2480     }
2481     { ##1 }
2482     \tl_use:N \l__enumext_label_fill_right_v_tl
2483 }
2484 }

```

(End of definition for `__enumext_keyans_make_label:`)

10.31 Calculation of `\leftmargin` and `\itemindent`

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

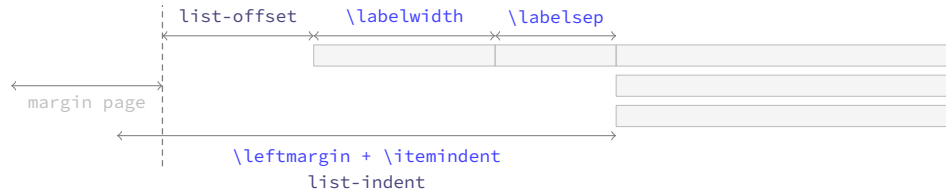


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

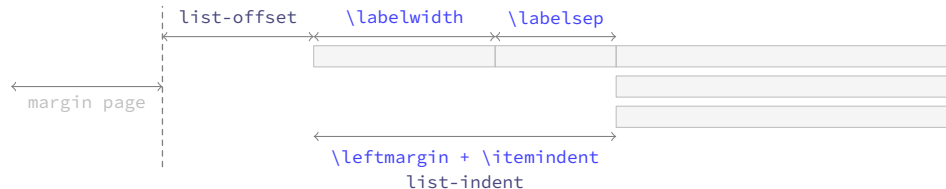


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

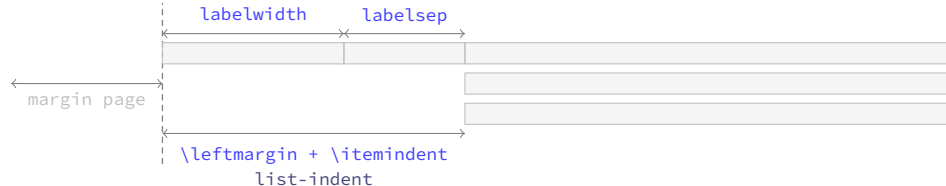


Figure 11: Default horizontal lengths in `enumext`.

```

\__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 (§10.32).

```

2485 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2486 {
2487     \dim_compare:nNt { #1 } < { \c_zero_dim }
2488     {
2489         \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2490         \dim_set:Nn #1 { \dim_abs:n { #1 } }
2491     }
2492     \dim_compare:nNt { #2 } < { \c_zero_dim }
2493     {
2494         \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2495         \dim_set:Nn #2 { \dim_abs:n { #2 } }
2496     }
2497 }

```

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

```

2497     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```


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

```

2498   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2499   {
2500     \dim_set:Nn #6 { #1 + #2 - #4 }
2501     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2502   }
2503   {
2504     \dim_compare:nNnT { #4 } = { #1 + #2 }
2505     { \dim_set:Nn #6 { \c_zero_dim } }
2506     \dim_compare:nNnT { #4 } < { #1 + #2 }
2507     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2508     \dim_compare:nNnT { #4 } > { #1 + #2 }
2509     {
2510       \dim_set:Nn #6 { -#1 - #2 + #4 }
2511       \dim_set:Nn #6 { #6*-1 }
2512     }
2513     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2514   }
2515 }
2516 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

10.32 Setting second argument of the lists

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

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

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

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
2517 \cs_set_protected:Npn \__enumext_tmp:n #1
2518 {
2519   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2520   {
2521     \__enumext_calc_hspace:ccccc
2522     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2523     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2524     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2525     { \__enumext_leftmargin_tmp_#1_bool }
2526     \clist_map_inline:nn
2527     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2528     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2529     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2530     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2531     \usecounter { enumX#1 }
2532     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2533     \str_if_eq:nnTF {#1} { v }
2534     {
2535       \__enumext_keyans_redefine_item:
2536       \__enumext_keyans_make_label:
2537       \__enumext_keyans_fake_item:
2538       \bool_if:cT { \__enumext_show_length_#1_bool }
2539       {
2540         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2541       }
2542     }
2543     {
2544       \__enumext_redefine_item:
2545       \__enumext_make_label:
2546       \__enumext_use_key_ref:
2547       \__enumext_fake_item:
2548       \bool_if:cT { \__enumext_show_length_#1_bool }
2549       {
2550         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2551       }
2552     }
2553   }

```

```

2554 }
2555 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

```

\__enumext_list_arg_two_vii:
  \__enumext_list_arg_two_viii:

```

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

```

2556 \cs_set_protected:Npn \__enumext_tmp:n #1
2557 {
2558   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2559   {
2560     \__enumext_calc_hspace:cccccc
2561     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2562     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2563     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2564     { \__enumext_leftmargin_tmp_#1_bool }
2565     \clist_map_inline:nn
2566     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2567     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2568     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2569     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2570     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2571     \skip_zero:N \partopsep
2572     \usecounter { enumX#1 }
2573     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2574     \__enumext_use_key_ref_h:
2575     \str_if_eq:nnTF {#1} { vii }
2576     {
2577       \__enumext_fake_item_vii:
2578       \bool_if:cT { \__enumext_show_length_vii_bool }
2579       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2580     }
2581     {
2582       \__enumext_fake_item_viii:
2583       \bool_if:cT { \__enumext_show_length_#1_bool }
2584       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2585     }
2586   }
2587 }
2588 \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:.)

10.33 The environment enumext

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

```

2589 \NewDocumentEnvironment{enumext}{0}{}
2590 {
2591   \__enumext_safe_exec:
2592   \__enumext_parse_keys:n {#1}
2593   \__enumext_before_list:
2594   \__enumext_start_store_level:
2595   \__enumext_start_list:nn
2596   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2597   {
2598     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2599     \__enumext_before_keys_exec:
2600   }
2601   \__enumext_after_args_exec:
2602 }
2603 {
2604   \__enumext_stop_list:
2605   \__enumext_stop_store_level:
2606   \__enumext_after_list:
2607 }

```

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

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

```

2608 \cs_new_protected:Nn \__enumext_safe_exec:
2609 {
2610   \__enumext_current_env_set_bool:
2611   \int_incr:N \l__enumext_level_int
2612   \int_compare:nNt { \l__enumext_level_int } > { 4 }
2613   { \msg_fatal:nn { enumext } { list-too-deep } }
2614   \bool_set_true:N \l__enumext_standar_bool
2615   \bool_lazy_all:nT
2616   {
2617     { \bool_if_p:N \g__enumext_standar_bool }
2618     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2619     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2620   }
2621   {
2622     \typeout{[[ON-FIRST-LEVEL-ENUMEXT-NOT-NESTED]]}
2623     \bool_set_true:N \l__enumext_standar_level_one_bool
2624   }
2625 }

```

(End of definition for `__enumext_safe_exec:`)

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

```

2626 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2627 {
2628   \tl_if_novalue:nF {#1}
2629   {
2630     \str_clear:N \l__enumext_series_str
2631     \int_compare:nNtF { \l__enumext_level_int } = { 1 }
2632     {
2633       \keys_set:nn { enumext / level-1 } {#1}
2634       \__enumext_parse_series_name:n {#1}
2635     }
2636     {
2637       \exp_args:Ne \keys_set:nn
2638       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2639     }
2640     \bool_if:NT \l__enumext_store_active_bool
2641     {
2642       \__enumext_parse_store_keys:n {#1}
2643     }
2644   }
2645 }

```

(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:`.

```

2646 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2647 {
2648   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2649   {
2650     \regex_match:nnT { \b columns\b } {#1}
2651     {
2652       \int_set_eq:cc
2653       { \l__enumext_store_columns_ \__enumext_level: _int }
2654       { \l__enumext_columns_ \__enumext_level: _int }
2655       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2656       {
2657         columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2658       }
2659     }
2660   }
2661   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }

```

```

2662     {
2663         \regex_match:nnT { \b columns-sep \b} {#1}
2664         {
2665             \dim_set_eq:cc
2666             { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2667             { l__enumext_columns_sep_ \__enumext_level: _dim }
2668             \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2669             {
2670                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2671             }
2672         }
2673     }
2674 }

```

(End of definition for __enumext_parse_store_keys:n.)

__enumext_start_store_level: The __enumext_start_store_level: and __enumext_stop_store_level: functions activate the level saving mechanism for storage in *sequence* of the \anskey command. __enumext_stop_store_level: If enumext are nested in enumext* add __enumext_store_level_open: to preserve the stored structure.

```

2675 \cs_new_protected:Nn \__enumext_start_store_level:
2676 {
2677     \bool_lazy_all:nT
2678     {
2679         { \bool_if_p:N \l__enumext_store_active_bool }
2680         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2681         { \bool_not_p:n { \g__enumext_starred_bool } }
2682     }
2683     {
2684         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2685         {
2686             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2687             \__enumext_store_level_open:
2688         }
2689     }
2690     \bool_lazy_all:nT
2691     {
2692         { \bool_if_p:N \l__enumext_store_active_bool }
2693         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2694         { \bool_if_p:N \g__enumext_starred_bool }
2695     }
2696     {
2697         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2698         {
2699             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2700             \__enumext_store_level_open:
2701         }
2702     }
2703 }
2704 \cs_new_protected:Nn \__enumext_stop_store_level:
2705 {
2706     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2707     {
2708         \__enumext_store_level_close:
2709     }
2710 }

```

(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.

```

2711 \cs_new_protected:Nn \__enumext_before_list:
2712 {
2713     \__enumext_vspace_above:
2714     \__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.

```

2715     \__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.

```

2716 \dim_compare:nNt
2717 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2718 {
2719   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2720   {
2721     \linewidth
2722     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2723     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2724   }

```

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

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

```

2725 \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2726 \int_gincr:N \g__enumext_minipage_stat_int
2727 \__enumext_mini_addvspace:
2728 \nointerlineskip\noindent
2729 \begin{__enumext_mini_env*}
2730 { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2731 }
2732 \__enumext_multicols_start:
2733 }

```

(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.

```

2734 \cs_new_protected:Nn \__enumext_multicols_start:
2735 {
2736   \int_compare:nNt
2737   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2738   {
2739     \dim_compare:nNt
2740     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2741     {
2742       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2743       {
2744         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2745         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2746         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2747         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2748       }
2749     }
2750     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2751     \skip_zero:N \multicolsep
2752     \int_compare:nNt { \l__enumext_level_int } > { 1 }
2753     {
2754       \dim_zero:N \columnseprule
2755     }

```

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.

```

2756 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2757 {
2758   \__enumext_multi_addvspace:
2759 }
2760 \raggedcolumns

```

```

2761         \begin{multicols}{\int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2762     }
2763 }

```

(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.

```

2764 \cs_new_protected:Nn \__enumext_multicols_stop:
2765 {
2766     \int_compare:nNtT
2767     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2768     {
2769         \end{multicols}
2770         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2771         {
2772             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2773         }
2774     }

```

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

```

2775     \bool_lazy_and:nnT
2776     { \bool_if_p:N \l__enumext_check_ans_bool }
2777     { \bool_not_p:n { \g__enumext_starred_bool } }
2778     {
2779         \bool_gset_true:N \g__enumext_check_ans_show_bool
2780         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2781     }
2782 }

```

(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.

```

2783 \cs_new_protected:Nn \__enumext_after_list:
2784 {
2785     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2786     {
2787         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2788         {
2789             \msg_warning:nn { enumext } { missing-miniright }
2790             \miniright
2791         }
2792         \int_gzero:N \g__enumext_minipage_stat_int
2793         \end{__enumext_mini_env*}
2794         \par\addvspace { \l__enumext_minipage_after_skip }
2795     }
2796     { \__enumext_multicols_stop: }

```

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

```

2797     \__enumext_after_stop_list:
2798     \__enumext_vspace_below:

```

Finally save the *current value* of the counter in \g__enumext_resume_int for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

2799     \bool_set_false:N \l__enumext_standar_bool
2800     \__enumext_resume_counter_set:
2801 }

```

(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`.

```

2802 \__enumext_after_env:nn {enumext}
2803 {
2804   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
2805   {
2806     \bool_if:NT \g__enumext_check_ans_show_bool
2807     {
2808       \__enumext_check_ans_show:
2809     }
2810     \bool_gset_false:N \g__enumext_standar_bool
2811     \bool_gset_false:N \g__enumext_check_ans_show_bool
2812     \tl_gclear:N \g__enumext_store_name_tl
2813   }
2814 }

```

10.34 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.

```

2815 \NewDocumentEnvironment{keyans}{ 0{} }
2816 {
2817   \__enumext_keyans_safe_exec:
2818   \__enumext_keyans_parse_keys:n {#1}
2819   \__enumext_before_list_v:
2820   \__enumext_start_list:nn
2821   { \tl_use:N \l__enumext_label_v_tl }
2822   {
2823     \__enumext_list_arg_two_v:
2824     \__enumext_before_keys_exec_v:
2825   }
2826   \__enumext_after_args_exec_v:
2827 }
2828 {
2829   \__enumext_keyans_check_ans:nn { item }{ keyans }
2830   \__enumext_stop_list:
2831   \__enumext_after_list_v:
2832 }

```

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

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

```

2833 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2834 {
2835   \bool_if:NF \l__enumext_store_active_bool
2836   {
2837     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2838   }
2839   \int_incr:N \l__enumext_keyans_level_int
2840   \bool_set_true:N \l__enumext_keyans_env_bool
2841   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2842   \bool_set_false:N \l__enumext_store_active_bool
2843   \int_compare:nNtT { \l__enumext_keyans_level_int } > { 1 }
2844   {
2845     \msg_error:nn { enumext } { keyans-nested }
2846   }
2847   \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2848   {
2849     \msg_error:nn { enumext } { keyans-wrong-level }
2850   }
2851 }

```

(End of definition for `__enumext_keyans_safe_exec:`.)

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

```
2852 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2853 {
2854   \keys_set:nn { enumext / keyans } {#1}
2855 }
```

(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.

```
2856 \cs_new_protected:Nn \__enumext_before_list_v:
2857 {
2858   \__enumext_vspace_above_v:
2859   \__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.

```
2860   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2861   {
2862     \dim_set:Nn \l__enumext_minipage_left_v_dim
2863     {
2864       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2865     }
2866   }
```

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*.

```
2866   \bool_set_true:N \l__enumext_minipage_active_v_bool
2867   \int_gincr:N \g__enumext_minipage_stat_int
2868   \__enumext_keyans_mini_addvspace:
2869   \nointerlineskip\noindent
2870   \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2871 }
```

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

```
2872   \__enumext_keyans_multicols_start:
2873 }
```

(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.

```
2874 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2875 {
2876   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2877   {
```

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

```
2878   \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2879   {
2880     \dim_set:Nn \l__enumext_columns_sep_v_dim
2881     {
2882       (
2883         \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2884       ) / \l__enumext_columns_v_int
2885       - \l__enumext_listoffset_v_dim
2886     }
2887   }
2888   \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).

```
2889   \skip_zero:N \multicolsep
2890   \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.

```

2891         \bool_if:NF \l__enumext_minipage_active_v_bool
2892         {
2893             \__enumext_keyans_multi_addvspace:
2894         }
2895     \raggedcolumns
2896     \begin{multicols}{\l__enumext_columns_v_int }
2897 }
2898 }

```

(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.

```

2899 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2900 {
2901     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2902     {
2903         \end{multicols}
2904         \bool_if:NF \l__enumext_minipage_active_v_bool
2905         {
2906             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2907         }
2908     }
2909 }

```

(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.

```

2910 \cs_new_protected:Nn \__enumext_after_list_v:
2911 {
2912     \bool_if:NTF \l__enumext_minipage_active_v_bool
2913     {
2914         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2915         {
2916             \msg_warning:nn { enumext } { missing-miniright }
2917             \miniright
2918         }
2919         \int_gzero:N \g__enumext_minipage_stat_int
2920         \end{__enumext_mini-env*}
2921         \par\addvspace{ \l__enumext_minipage_after_skip }
2922     }
2923     { \__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.

```

2924     \bool_set_false:N \l__enumext_keyans_env_bool
2925     \__enumext_after_stop_list_v:
2926     \__enumext_vspace_below_v:
2927 }

```

(End of definition for `__enumext_after_list_v:`)

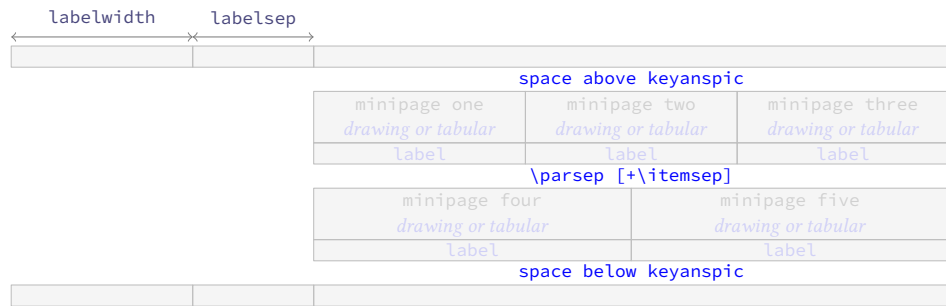
10.35 The environment `keyanspic` and `\anspic`

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`.

10.35.1 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.

```
2928 \NewDocumentCommand \anspic { s o +m }
2929 {
```

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

```
2930   \bool_if:NF \l__enumext_store_active_bool
2931   {
2932     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2933   }
2934   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2935   {
2936     \msg_error:nn { enumext } { keyanspic-wrong-level }
2937   }
2938   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2939   {
2940     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2941   }
```

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.

```
2942   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2943   {
2944     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2945   }
2946 }
```

(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.

```
2947 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2948 {
2949   \stepcounter { enumXvi }
2950   #3 \\\
2951   \bool_if:nT { #1 }
2952   {
2953     \__enumext_keyans_addto_prop:n { #2 }
2954     \__enumext_keyans_store_ref:
2955     \__enumext_keyans_addto_seq:n { #2 }
2956     \bool_lazy_or:nnT
2957     { \bool_if_p:N \l__enumext_show_answer_bool }
2958     { \bool_if_p:N \l__enumext_show_position_bool }
2959     {
2960       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2961       \__enumext_keyans_show_left:n { #2 }
2962       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2963     }
2964   }
2965   \tl_use:N \l__enumext_label_font_style_v_tl
2966   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2967 }
```

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

10.35.2 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.

```
2968 \NewDocumentEnvironment{keyanspic}{ o }
2969 {
2970   \__enumext_keyans_pic_safe_exec:
2971   \__enumext_start_list:nn
2972   { }
2973   {
2974     \__enumext_keyans_pic_arg_two:
2975   }

```

We apply the “adjusted” vertical spacing above the environment

```
2976   \vspace { \__enumext_keyans_pic_above_skip }
2977 }
```

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.

```
2978 {
2979   \tl_if_novalue:nTF { #1 }
2980   {
2981     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2982   }
2983   { \__enumext_keyans_pic_do:n { #1 } }
2984   \__enumext_stop_list:
2985   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2986   \setcounter { enumXvi } { 0 }
2987   \vspace { \__enumext_topsep_v_skip }
2988   %\bool_set_false:N \l__enumext_store_active_bool
2989 }
```

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

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

```
2990 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2991 {
2992   \int_incr:N \l__enumext_keyans_pic_level_int
2993   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2994   {
2995     \msg_error:nn { enumext } { keyanspic-nested }
2996   }
2997 }
```

(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`.

```
2998 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2999 {
3000   \dim_compare:nNnT { #1 } < { 0pt }
3001   { \skip_set:Nn #1 { -#1 } }
3002 }
```

(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.

```
3003 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3004 {
```

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.

```
3005   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3006   \__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.

```

3007 \skip_add:Nn \parsep { \itemsep }
3008 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3009 \dim_zero:N \labelwidth
3010 \dim_zero:N \listparindent
3011 \dim_zero:N \labelsep
3012 \skip_zero:N \partopsep
3013 \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.

```

3014 \__enumxt_keyans_pic_skip_abs:N \parsep
3015 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
3016 {
3017   \box_dp:N \strutbox
3018   + \l__enumxt_topsep_v_skip
3019   - \parsep
3020 }
3021 \__enumxt_item_std:w \scan_stop:
3022 }

```

(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`.

```

3023 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
3024 {
3025   \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
3026 }
3027 \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.

```

3028 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
3029 {
3030   \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
3031   \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
3032   \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
3033   \int_step_inline:nnn
3034   { \l__enumxt_keyans_pic_above_int + 1 }
3035   { \l__enumxt_keyans_pic_below_int }
3036   {
3037     \__enumxt_minipage:w [ b ]{ \l__enumxt_keyans_pic_width_dim }
3038     \centering
3039     \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
3040     \__enumxt_endminipage:
3041   }
3042   \par
3043 }

```

(End of definition for `__enumxt_keyans_pic_row:n`)

10.36 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 ($\langle number \rangle$).

10.36.1 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.

```

3044 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3045 {
3046   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3047   {
3048     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3049     {
3050       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3051       / \l__enumext_columns_vii_int
3052     }
3053   }
3054   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3055   \dim_set:Nn \l__enumext_item_width_vii_dim
3056   {
3057     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3058     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3059     - \l__enumext_labelsep_vii_dim
3060   }
3061   \dim_zero_new:N \itemwidth
3062 }

```

(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`.

```

3063 \cs_new_protected:Npn \_enumext_starred_joined_item_vii:n #1
3064 {
3065   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3066   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3067   {
3068     \msg_warning:nnee { enumext } { item-joined }
3069     { \int_use:N \l__enumext_joined_item_vii_int }
3070     { \int_use:N \l__enumext_columns_vii_int }
3071     \int_set:Nn \l__enumext_joined_item_vii_int
3072     {
3073       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3074     }
3075   }
3076   \int_compare:nNnT
3077   { \l__enumext_joined_item_vii_int }
3078   >
3079   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3080   {
3081     \msg_warning:nnee { enumext } { item-joined-columns }
3082     { \int_use:N \l__enumext_joined_item_vii_int }
3083     {
3084       \int_eval:n
3085       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3086     }
3087     \int_set:Nn \l__enumext_joined_item_vii_int
3088     {
3089       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3090     }
3091   }

```

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

```

3092   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3093   {
3094     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3095     \int_decr:N \l__enumext_joined_item_aux_vii_int
3096     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3097     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3098     \dim_set:Nn \l__enumext_joined_width_vii_dim
3099     {
3100       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3101       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3102         + \l__enumext_columns_sep_vii_dim
3103         ) * \l__enumext_joined_item_aux_vii_int

```

```

3104     }
3105     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3106   }
3107   {
3108     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3109     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3110   }
3111 }

```

(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`.

```

3112 \cs_new_protected:Nn \__enumext_start_mini_vii:
3113 {
3114   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3115   {
3116     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3117     {
3118       \linewidth
3119       - \l__enumext_minipage_right_vii_dim
3120       - \l__enumext_minipage_hsep_vii_dim
3121     }
3122     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3123     \dim_gset_eq:NN
3124       \g__enumext_minipage_right_vii_dim
3125       \l__enumext_minipage_right_vii_dim
3126     \__enumext_mini_addvspace_vii:
3127     \nointerlineskip\noindent
3128     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3129   }
3130 }

```

(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*”.

```

3131 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3132 {
3133   \bool_if:NT \l__enumext_minipage_active_vii_bool
3134   {
3135     \end{__enumext_mini_env*}
3136     \hfill
3137     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3138   }
3139 }

```

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*”.

```

3140 \__enumext_after_env:nn {enumext*}
3141 {
3142   \bool_if:NT \g__enumext_minipage_active_vii_bool
3143   {
3144     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3145     \par\addvspace { \g__enumext_minipage_right_skip }
3146     \bool_if:NF \g__enumext_minipage_center_vii_bool
3147     {
3148       \centering
3149     }
3150     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3151     \end{__enumext_mini_env*}
3152     \par\addvspace{ \g__enumext_minipage_after_skip }
3153   }
3154   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3155   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3156   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3157   \dim_gzero:N \g__enumext_minipage_right_vii_dim

```



```

3158     \bool_gset_false:N \g__enumext_starred_bool
3159 }

```

(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.

```

3160 \NewDocumentEnvironment{enumext*}{o}{
3161 {
3162     \__enumext_safe_exec_vii:
3163     \__enumext_parse_keys_vii:n {#1}
3164     \__enumext_before_list_vii:
3165     \__enumext_start_store_level_vii:
3166     \__enumext_start_list:nn { }
3167     {
3168         \__enumext_list_arg_two_vii:
3169         \__enumext_before_keys_exec_vii:
3170     }
3171     \__enumext_starred_columns_set_vii:
3172     \item[] \scan_stop:
3173     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3174     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3175 }
3176 {
3177     \__enumext_stop_item_tmp_vii:
3178     \__enumext_remove_extra_parsep_vii:
3179     \__enumext_stop_list:
3180     \__enumext_stop_store_level_vii:
3181     \__enumext_after_list_vii:
3182 }

```

(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`.

```

3183 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3184 {
3185     \__enumext_current_env_set_bool:
3186     \int_incr:N \l__enumext_level_h_int
3187     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3188     {
3189         \msg_error:nn { enumext } { nested }
3190     }
3191     \bool_set_true:N \l__enumext_starred_bool
3192     \bool_lazy_all:nT
3193     {
3194         { \bool_if_p:N \g__enumext_starred_bool }
3195         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3196         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
3197     }
3198     {
3199         \typeout{[[ON-FIRST-LEVEL-ENUMEXT*-NOT-NESTED]]}
3200         \bool_set_true:N \l__enumext_starred_level_one_bool
3201     }
3202 }

```

(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.

```

3203 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3204 {
3205     \tl_if_novalue:nF {#1}
3206     {
3207         \str_clear:N \l__enumext_series_str
3208         \keys_set:nn { enumext / enumext* } {#1}
3209         \__enumext_parse_series_name:n {#1}
3210         \bool_if:NT \l__enumext_store_active_bool
3211         {

```

```

3212         \__enumext_parse_store_keys_vii:n {#1}
3213     }
3214 }
3215 }

```

(End of definition for __enumext_parse_keys_vii:n.)

__enumext_parse_store_keys_vii:n

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

```

3216 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3217 {
3218     \bool_if:NF \l__enumext_store_columns_vii_bool
3219     {
3220         \regex_match:nnT { \b columns\b } {#1}
3221         {
3222             \int_set_eq:NN
3223             \l__enumext_store_columns_vii_int
3224             \l__enumext_columns_vii_int
3225             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3226             {
3227                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3228             }
3229         }
3230     }
3231     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3232     {
3233         \regex_match:nnT { \b columns-sep\b } {#1}
3234         {
3235             \dim_set_eq:NN
3236             \l__enumext_store_columns_sep_vii_dim
3237             \l__enumext_columns_sep_vii_dim
3238             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3239             {
3240                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3241             }
3242         }
3243     }
3244 }

```

(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`.

```

3245 \cs_new_protected:Nn \__enumext_before_list_vii:
3246 {
3247     \__enumext_vspace_above_vii:
3248     \__enumext_check_ans_exec: % need by chek-ans
3249     \__enumext_before_args_exec_vii:
3250     \__enumext_start_mini_vii:
3251 }

```

(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.

```

3252 \cs_new_protected:Nn \__enumext_after_list_vii:
3253 {
3254     \__enumext_stop_mini_vii:
3255     \__enumext_after_stop_list_vii:
3256     \__enumext_vspace_below_vii:
3257     %\bool_gset_false:N \g__enumext_starred_bool
3258     \bool_set_false:N \l__enumext_starred_bool
3259     \__enumext_resume_counter_set:
3260 }

```

(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`.

```

3261 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3262 {
3263   \bool_if:NT \l__enumext_store_active_bool
3264   {
3265     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3266     {
3267       \__enumext_store_level_open_vii:
3268     }
3269   }
3270 }
3271 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3272 {
3273   \bool_if:NT \l__enumext_store_active_bool
3274   {
3275     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3276     {
3277       \__enumext_store_level_close_vii:
3278     }
3279   }
3280 }
```

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

10.36.2 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`.

```

3281 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3282 {
3283   \__enumext_stop_item_tmp_vii:
3284   \int_incr:N \l__enumext_item_column_pos_vii_int
3285   \int_gincr:N \g__enumext_item_count_all_vii_int
3286   \__enumext_item_peek_args_vii:
3287 }
```

(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).

```

3288 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3289 {
3290   \peek_meaning:NTF (
3291     { \__enumext_joined_item_vii:w }
3292     { \__enumext_joined_item_vii:w (1) }
3293   }
```

(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`.

```

3294 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3295 {
3296   \__enumext_starred_joined_item_vii:n {#1}
3297   \peek_meaning_remove:NTF *
3298     { \__enumext_starred_item_vii:w }
3299     { \__enumext_standard_item_vii:w }
3300 }
```

(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].

```

3301 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3302 {
3303   \bool_set_false:N \l__enumext_item_starred_vii_bool
3304   \peek_meaning:NTF [
3305     {
3306       \bool_set_eq:NN
3307         \l__enumext_wrap_label_vii_bool
3308         \l__enumext_wrap_label_opt_vii_bool
3309       \__enumext_start_item_vii:w
3310     }
3311     {
3312       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3313       \legacy_if_set_true:n { @noitemarg }
3314       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3315     }
3316   }

```

(End of definition for __enumext_standard_item_vii:w.)

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

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

```

3317 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3318 {
3319   \bool_set_true:N \l__enumext_item_starred_vii_bool
3320   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3321   \peek_meaning:NTF [
3322     { \__enumext_starred_item_vii_aux_i:w }
3323     { \__enumext_starred_item_vii_aux_ii:w }
3324   }
3325   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3326   {
3327     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3328     \__enumext_starred_item_vii_aux_ii:w
3329   }
3330   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3331   {
3332     \peek_meaning:NTF [
3333       { \__enumext_starred_item_vii_aux_iii:w }
3334       {
3335         \dim_set_eq:NN
3336           \l__enumext_item_symbol_sep_vii_dim
3337           \l__enumext_labelsep_vii_dim
3338         \legacy_if_set_true:n { @noitemarg }
3339         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3340       }
3341     }
3342   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3343   {
3344     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3345     \legacy_if_set_true:n { @noitemarg }
3346     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3347   }

```

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

10.36.3 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.

```

3348 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3349 {
3350   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3351   \legacy_if:nT { @noitemarg }
3352   {
3353     \legacy_if_set_false:n { @noitemarg }
3354     \legacy_if:nT { @nmbrlist }
3355     {
3356       \bool_if:NT \l__enumext_hyperref_bool
3357       {
3358         \legacy_if_set_true:n { @hyper@item }
3359       }
3360       \refstepcounter{enumXvii}
3361       \bool_if:NT \l__enumext_check_ans_bool
3362       {
3363         \int_gincr:N \g__enumext_count_item_number_int
3364       }
3365     }
3366   }

```

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

```

3367   \group_begin:
3368   \lrbox{ \l__enumext_item_text_vii_box }
3369   \bool_if:NF \l__enumext_footnotes_key_bool
3370   {
3371     \__enumext_renew_footnote:
3372   }
3373   \bool_if:NT \l__enumext_item_starred_vii_bool
3374   {
3375     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3376     {
3377       \tl_gset_eq:NN
3378       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3379     }
3380     \mode_leave_vertical:
3381     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3382     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3383     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3384     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3385   }
3386   \group_begin:
3387   \tl_use:N \l__enumext_label_font_style_vii_tl
3388   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3389   {
3390     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3391     { \__enumext_wrapper_label_vii:n {#1} }
3392   }
3393   {
3394     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3395   }
3396   \group_end:
3397   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3398   \tl_use:N \l__enumext_after_list_args_vii_tl
3399   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3400   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3401   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3402   \tl_use:N \l__enumext_fake_item_indent_vii_tl
3403 }

```

(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.

```

3404 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3405 {
3406     \__enumext_endminipage:
3407     \endlrbox
3408     \group_end:
3409     \box_set_wd:Nn \l__enumext_item_text_vii_box
3410     {
3411         \l__enumext_joined_width_vii_dim
3412         + \l__enumext_labelwidth_vii_dim
3413         + \l__enumext_labelsep_vii_dim
3414     }
3415     \int_set:Nn \hbadness { 10000 }
3416     \box_use:N \l__enumext_item_text_vii_box
3417     \bool_if:NF \l__enumext_footnotes_key_bool
3418     {
3419         \__enumext_print_footnote:
3420     }
3421     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3422     {
3423         \par\noindent
3424         \int_zero:N \l__enumext_item_column_pos_vii_int
3425     }
3426     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3427 }

```

(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.

```

3428 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3429 {
3430     \int_compare:nNnT
3431     {
3432         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3433     }
3434     =
3435     { \c_zero_int }
3436     {
3437         \par
3438         \vspace{ -\l__enumext_itemsep_vii_skip }
3439         \int_gzero:N \g__enumext_item_count_all_vii_int
3440     }
3441 }

```

(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`.

```

3442 \__enumext_after_env:nn {enumext*}
3443 {
3444     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3445     {
3446         \bool_if:NT \g__enumext_check_ans_show_h_bool
3447         {
3448             \__enumext_check_ans_show:
3449         }
3450         \bool_gset_false:N \g__enumext_starred_bool
3451         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3452         \tl_gclear:N \g__enumext_store_name_tl
3453     }
3454 }

```

10.37 The environment `keyans*`

10.37.1 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.

```

3455 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3456 {
3457     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }

```

```

3458     {
3459         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3460         {
3461             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3462             / \l__enumext_columns_viii_int
3463         }
3464     }
3465     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3466     \dim_set:Nn \l__enumext_item_width_viii_dim
3467     {
3468         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3469         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3470         - \l__enumext_labelsep_viii_dim
3471     }
3472     \dim_zero_new:N \itemwidth
3473 }

```

(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.

```

3474 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3475 {
3476     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3477     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3478     {
3479         \msg_warning:nnee { enumext } { item-joined }
3480         { \int_use:N \l__enumext_joined_item_viii_int }
3481         { \int_use:N \l__enumext_columns_viii_int }
3482         \int_set:Nn \l__enumext_joined_item_viii_int
3483         {
3484             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3485         }
3486     }
3487     \int_compare:nNnT
3488     { \l__enumext_joined_item_viii_int }
3489     >
3490     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3491     {
3492         \msg_warning:nnee { enumext } { item-joined-columns }
3493         { \int_use:N \l__enumext_joined_item_viii_int }
3494         {
3495             \int_eval:n
3496             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3497         }
3498         \int_set:Nn \l__enumext_joined_item_viii_int
3499         {
3500             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3501         }
3502     }
3503     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3504     {
3505         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3506         \int_decr:N \l__enumext_joined_item_aux_viii_int
3507         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3508         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3509         \dim_set:Nn \l__enumext_joined_width_viii_dim
3510         {
3511             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3512             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3513                 + \l__enumext_columns_sep_viii_dim
3514                 ) * \l__enumext_joined_item_aux_viii_int
3515         }
3516         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3517     }
3518     {
3519         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3520         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3521     }
3522 }

```

(End of definition for `__enumext_starred_joined_item_viii:n`)

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

```

3523 \cs_new_protected:Nn \__enumext_start_mini_viii:
3524 {
3525   \dim_compare:nNt { \__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3526   {
3527     \dim_set:Nn \__enumext_minipage_left_viii_dim
3528     {
3529       \linewidth
3530       - \__enumext_minipage_right_viii_dim
3531       - \__enumext_minipage_hsep_viii_dim
3532     }
3533     \bool_set_true:N \__enumext_minipage_active_viii_bool
3534     \dim_gset_eq:NN
3535       \g__enumext_minipage_right_viii_dim
3536       \l__enumext_minipage_right_viii_dim
3537     \__enumext_mini_addvspace_viii:
3538     \nointerlineskip\noindent
3539     \begin{__enumext_mini_env*}{ \__enumext_minipage_left_viii_dim }
3540   }
3541 }
3542 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3543 {
3544   \bool_if:NT \__enumext_minipage_active_viii_bool
3545   {
3546     \end{__enumext_mini_env*}
3547     \hfill
3548     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3549   }
3550 }
3551 \__enumext_after_env:nn {keyans*}
3552 {
3553   \bool_if:NT \g__enumext_minipage_active_viii_bool
3554   {
3555     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3556     \par\addvspace { \g__enumext_minipage_right_skip }
3557     \bool_if:NF \g__enumext_minipage_center_viii_bool
3558     {
3559       \centering
3560     }
3561     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3562     \end{__enumext_mini_env*}
3563     \par\addvspace{ \g__enumext_minipage_after_skip }
3564   }
3565   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3566   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3567   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3568   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3569 }

```

(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.

```

3570 \NewDocumentEnvironment{keyans*}{ o }
3571 {
3572   \__enumext_safe_exec_viii:
3573   \__enumext_parse_keys_viii:n {#1}
3574   \__enumext_before_list_viii:
3575   \__enumext_start_list:nn { }
3576   {
3577     \__enumext_list_arg_two_viii:
3578     \__enumext_before_keys_exec_viii:
3579   }
3580   \__enumext_starred_columns_set_viii:
3581   \item[] \scan_stop:
3582   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3583   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3584 }

```



```

3585 {
3586   \__enumext_stop_item_tmp_viii:
3587   \__enumext_remove_extra_parsep_viii:
3588   \__enumext_keyans_check_ans:nn { item }{ keyans* }
3589   \__enumext_stop_list:
3590   \__enumext_after_list_viii:
3591 }

```

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

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

```

3592 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3593 {
3594   \int_incr:N \l__enumext_keyans_level_h_int
3595   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3596   {
3597     \msg_error:nn { enumext } { nested }
3598   }
3599   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3600   \bool_set_false:N \l__enumext_store_active_bool
3601   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3602   {
3603     \msg_error:nn { enumext } { keyans-wrong-level }
3604   }
3605 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

3606 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3607 {
3608   \tl_if_novalue:nF {#1}
3609   {
3610     \keys_set:nn { enumext / keyans* } {#1}
3611   }
3612 }

```

(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`.

```

3613 \cs_new_protected:Nn \__enumext_before_list_viii:
3614 {
3615   \__enumext_vspace_above_viii:
3616   \__enumext_before_args_exec_viii:
3617   \__enumext_start_mini_viii:
3618 }

```

(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.

```

3619 \cs_new_protected:Nn \__enumext_after_list_viii:
3620 {
3621   \__enumext_stop_mini_viii:
3622   \__enumext_after_stop_list_viii:
3623   \__enumext_vspace_below_viii:
3624 }

```

(End of definition for `__enumext_after_list_viii:`.)

10.37.2 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`.

```
3625 \cs_new_protected_nopar:Nn \_enumext_start_item_tmp_viii:
3626 {
3627   \_enumext_stop_item_tmp_viii:
3628   \int_incr:N \l\_enumext_item_column_pos_viii_int
3629   \int_gincr:N \g\_enumext_item_count_all_viii_int
3630   \_enumext_item_peek_args_viii:
3631 }
```

(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).

```
3632 \cs_new_protected:Nn \_enumext_item_peek_args_viii:
3633 {
3634   \peek_meaning:NTF (
3635     { \_enumext_joined_item_viii:w }
3636     { \_enumext_joined_item_viii:w (1) }
3637 }
```

(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`.

```
3638 \cs_new_protected:Npn \_enumext_joined_item_viii:w (#1)
3639 {
3640   \_enumext_starred_joined_item_viii:n {#1}
3641   \peek_meaning_remove:NTF *
3642     { \_enumext_starred_item_viii:w }
3643     { \_enumext_standard_item_viii:w }
3644 }
```

(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]`.

```
3645 \cs_new_protected:Npn \_enumext_standard_item_viii:w
3646 {
3647   \bool_set_false:N \l\_enumext_item_starred_viii_bool
3648   \peek_meaning:NTF [
3649     {
3650       \bool_set_eq:NN
3651       \l\_enumext_wrap_label_viii_bool
3652       \l\_enumext_wrap_label_opt_viii_bool
3653       \_enumext_start_item_viii:w
3654     }
3655     {
3656       \bool_set_true:N \l\_enumext_wrap_label_viii_bool
```

```

3657         \legacy_if_set_true:n { @noitemarg }
3658         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3659     }
3660 }

```

(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*[\content].

```

3661 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3662 {
3663     \bool_set_true:N \__enumext_item_starred_viii_bool
3664     \bool_set_true:N \__enumext_wrap_label_viii_bool
3665     \peek_meaning:NTF [
3666         { \__enumext_starred_item_viii_aux_i:w }
3667         { \__enumext_starred_item_viii_aux_ii:w }
3668     }

```

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 \sequence and \prop list.

```

3669 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3670 {
3671     \tl_clear:N \__enumext_store_keyans_label_tl
3672     \tl_if_no_value:nF { #1 }
3673     {
3674         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3675         {
3676             \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
3677             \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
3678         }
3679         \tl_set:Ne \__enumext_keyans_item_opt_tl { #1 }
3680     }
3681     \__enumext_starred_item_viii_aux_ii:w
3682 }
3683 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3684 {
3685     \legacy_if_set_true:n { @noitemarg }
3686     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3687 }

```

(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 \label for \item* followed by the [\content] for \item*[\content] if present in the \sequence and \prop list set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```

3688 \cs_new_protected:Nn \__enumext_starred_item_exec:
3689 {
3690     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3691     \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3692     \__enumext_keyans_store_ref:
3693     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3694     \__enumext_keyans_addto_seq_link:
3695     \bool_if:NT \__enumext_show_answer_bool
3696     {
3697         \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3698     }
3699     \bool_if:NT \__enumext_show_position_bool
3700     {
3701         \tl_set:Ne \__enumext_mark_answer_sym_tl
3702         {
3703             \group_begin:
3704             \exp_not:N \normalfont
3705             \exp_not:N \footnotesize [ \int_eval:n
3706                 {
3707                     \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3708                 }
3709             ]
3710             \group_end:

```

```

3711     }
3712     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3713   }
3714 }

```

(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.

```

3715 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3716 {
3717   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3718   \legacy_if:nT { @noitemarg }
3719   {
3720     \legacy_if_set_false:n { @noitemarg }
3721     \legacy_if:nT { @nmbrrlist }
3722     {
3723       \bool_if:NT \l__enumext_hyperref_bool
3724       {
3725         \legacy_if_set_true:n { @hyper@item }
3726       }
3727       \refstepcounter{enumXviii}
3728     }
3729   }

```

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

```

3730   \group_begin:
3731   \lrbox{ \l__enumext_item_text_viii_box }
3732   \bool_if:NF \l__enumext_footnotes_key_bool
3733   {
3734     \__enumext_renew_footnote:
3735   }
3736   \bool_if:NT \l__enumext_item_starred_viii_bool
3737   {
3738     \__enumext_starred_item_exec:
3739   }
3740   \group_begin:
3741   \tl_use:N \l__enumext_label_font_style_viii_tl
3742   \bool_if:NTF \l__enumext_wrap_label_viii_bool
3743   {
3744     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3745     { \__enumext_wrapper_label_viii:n {#1} }
3746   }
3747   {
3748     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
3749   }
3750   \group_end:
3751   \skip_horizontal:N \l__enumext_labelsep_viii_dim
3752   \tl_use:N \l__enumext_after_list_args_viii_tl
3753   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3754   \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3755   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3756   \bool_if:NT \l__enumext_item_starred_viii_bool
3757   {
3758     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3759     \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3760   }
3761   {
3762     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3763   }
3764 }

```

(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.

```

3765 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3766 {
3767   \__enumext_endminipage:
3768   \endlrbox

```

```

3769 \group_end:
3770 \box_set_wd:Nn \l__enumext_item_text_viii_box
3771 {
3772   \l__enumext_joined_width_viii_dim
3773   + \l__enumext_labelwidth_viii_dim
3774   + \l__enumext_labelsep_viii_dim
3775 }
3776 \int_set:Nn \hbadness { 10000 }
3777 \box_use:N \l__enumext_item_text_viii_box
3778 \bool_if:NF \l__enumext_footnotes_key_bool
3779 {
3780   \__enumext_print_footnote:
3781 }
3782 \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3783 {
3784   \par\noindent
3785   \int_zero:N \l__enumext_item_column_pos_viii_int
3786 }
3787 { \hspace{ \l__enumext_columns_sep_viii_dim } }
3788 }

```

(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.

```

3789 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3790 {
3791   \int_compare:nNnT
3792   {
3793     \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3794   }
3795   =
3796   { \c_zero_int }
3797   {
3798     \par
3799     \vspace{ -\l__enumext_itemsep_viii_skip }
3800     \int_gzero:N \g__enumext_item_count_all_viii_int
3801   }
3802 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

10.38 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.

```

3803 \NewDocumentCommand \getkeyans { m }
3804 {
3805   \exp_args:Ne \__enumext_getkeyans_aux:n
3806   { \tl_to_str:e { \text_expand:n {#1} } }
3807 }

```

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

__enumext_getkeyans_aux:n

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

```

3808 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3809 {
3810   \str_if_in:nnTF {#1} { : }
3811   {
3812     \use:e
3813     {
3814       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3815       { {##1} {##2} }
3816     }
3817     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3818   }
3819   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3820 }

```

(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⟩*.

```

3821 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3822 {
3823   \prop_if_exist:cF { g__enumext_#1_prop }
3824   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3825   \group_begin:
3826     \prop_item:cn { g__enumext_#1_prop }{#2}
3827   \group_end:
3828 }

```

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

10.39 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`.

```

3829 \keys_define:nn { keyanskey / print }
3830 {
3831   level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3832                       {
3833                         \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3834                       },
3835   level-1 .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3836   level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3837                       {
3838                         \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3839                       },
3840   level-2 .initial:n   = { nosep, label=(\alph*. , first=\small, font=\small },
3841   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3842                       {
3843                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3844                       },
3845   level-3 .initial:n   = { nosep, label=\roman*. , first=\small, font=\small },
3846   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3847                       {
3848                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3849                       },
3850   level-4 .initial:n   = { nosep, label=\Alph*. , first=\small, font=\small },
3851   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3852                       {
3853                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3854                       },
3855   level-* .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3856 }

```

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

```

3857 \NewDocumentCommand \printkeyans { s O{ } m }
3858 {
3859   \group_begin:
3860     \tl_use:N \l__enumext_print_keyans_i_tl
3861     \tl_use:N \l__enumext_print_keyans_ii_tl
3862     \tl_use:N \l__enumext_print_keyans_iii_tl
3863     \tl_use:N \l__enumext_print_keyans_iv_tl
3864     \tl_use:N \l__enumext_print_keyans_vii_tl
3865     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3866   \group_end:
3867 }

```

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

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

```

#1:  starred
#2:  key-val
#3:  seq-name

```

```

3868 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3869 {
3870   \seq_if_exist:cTF { g__enumext_#3_seq }
3871   {
3872     \seq_if_empty:cF { g__enumext_#3_seq }
3873     {
3874       %%\seq_show:c { g__enumext_#3_seq }
3875       \bool_if:nTF {#1}
3876       {
3877         \begin{enumext*}[#2]
3878         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3879         \end{enumext*}
3880       }
3881       {
3882         \begin{enumext}[#2]
3883         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3884         \end{enumext}
3885       }
3886     }
3887   }
3888   {
3889     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3890   }
3891 }

```

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

10.40 The command `\setenumext`

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

```

3892 \keys_define:nn { enumext / meta-families }
3893 {
3894   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3895   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
3896   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
3897   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
3898   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },
3899   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} },
3900   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
3901   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } },
3902   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } },
3903   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } },
3904   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } },
3905   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } },
3906   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
3907 }

```

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

```

3908 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3909 {
3910   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3911   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3912 }

```

`\setenumext` Now we define the user command `\setenumext`.

```

3913 \NewDocumentCommand \setenumext { o +m }
3914 {
3915   \tl_if_novalue:nTF {#1}
3916   {
3917     \seq_map_inline:Nn \c__enumext_all_families_seq
3918   }
3919   {
3920     \seq_clear:N \l__enumext_setkey_tmpa_seq
3921     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3922     \int_set:Nn \l__enumext_setkey_tmpa_int
3923     {
3924       \seq_count:N \l__enumext_setkey_tmpb_seq
3925     }
3926     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3927     {
3928       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3929       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n

```

```

3930         \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3931         {
3932             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3933         }
3934     }
3935     {
3936         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3937     }
3938     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3939     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3940     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3941 }
3942 {
3943     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3944 }
3945 }

```

(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.

```

3946 \cs_new_protected:Npn \__enumext_set_parse:n #1
3947 {
3948     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3949     \int_step_inline:nnn { 0 } { 4 } % <- max level
3950     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3951     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3952     {
3953         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3954         { \tl_trim_spaces:n {#1} }
3955     }
3956     { \__enumext_set_error:nn {#1} { } }
3957 }
3958 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3959 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

10.41 Messages

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

```

3960 \msg_new:nnn { enumext } { package-load }
3961 {
3962     The ~ '#1' ~ package ~ is ~ already ~ loaded.
3963 }
3964 \msg_new:nnn { enumext } { package-not-load }
3965 {
3966     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3967 }
3968 \msg_new:nnn { enumext } { package-load-foot }
3969 {
3970     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3971 }

```

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

```

3972 \msg_new:nnn { enumext } { counters }
3973 {
3974     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3975     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3976 }

```

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

```

3977 \msg_new:nnn { enumext } { invalid-key }
3978 {
3979     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3980 }
3981 \msg_new:nnn { enumext } { unknown-key-family }
3982 {
3983     Unknown~key~family~`\l_keys_key_str'~for~enumext.
3984 }

```


Messages used in length calculation.

```

3985 \msg_new:nnn { enumext } { width-negative }
3986 {
3987   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3988   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
3989 }
3990 \msg_new:nnn { enumext } { width-zero }
3991 {
3992   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3993   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
3994 }

```

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

```

3995 \msg_new:nnn { enumext } { list-lengths }
3996 {
3997   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3998   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3999   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4000   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4001   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4002   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4003   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4004   \__enumext_show_length:nnn { skip } { topsep } {#1}
4005   \__enumext_show_length:nnn { skip } { parsep } {#1}
4006   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4007   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4008   ****
4009 }

```

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

```

4010 \msg_new:nnn { enumext } { list-lengths-not-nested }
4011 {
4012   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4013   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4014   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4015   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4016   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4017   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4018   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4019   \__enumext_show_length:nnn { skip } { topsep } {#1}
4020   \__enumext_show_length:nnn { skip } { parsep } {#1}
4021   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4022   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4023   ****
4024 }

```

Messages used by `save-ans` key.

```

4025 \msg_new:nnn { enumext } { save-ans-empty }
4026 {
4027   The ~ 'save-ans' ~ key ~ cannot ~ be ~ empty~ in ~ '#1'. ~ \msg_line_context:.
4028 }
4029 \msg_new:nnn { enumext } { save-ans-nested }
4030 {
4031   The ~ 'save-ans' ~ key ~ cannot ~ be ~ used ~ in ~ nested ~ '#1'. ~ \msg_line_context:.
4032 }

```

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

```

4033 \msg_new:nnn { enumext } { items-same-answer }
4034 {
4035   *****~Checking~answers~on~'#1'~OK~*****\
4036   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \
4037   *****
4038   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4039 }
4040 \msg_new:nnn { enumext } { item-different-answer }
4041 {
4042   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4043   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4044 }

```

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

```

4045 \msg_new:nnn { enumext } { missing-starred }
4046 {

```

```

4047     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4048 }

```

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

```

4049 \msg_new:nnn { enumext } { list-too-deep }
4050 {
4051     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4052     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4053 }

```

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

```

4054 \msg_new:nnn { enumext } { anskey-wrong-place }
4055 {
4056     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4057     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4058 }
4059 \msg_new:nnn { enumext } { anspic-wrong-place }
4060 {
4061     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4062     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4063 }
4064 \msg_new:nnn { enumext } { command-wrong-place }
4065 {
4066     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4067     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4068 }

```

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

```

4069 \msg_new:nnn { enumext } { keyans-nested }
4070 {
4071     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4072 }
4073 \msg_new:nnn { enumext } { keyans-wrong-level }
4074 {
4075     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4076     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4077 }
4078 \msg_new:nnn { enumext } { wrong-place }
4079 {
4080     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4081     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4082 }
4083 \msg_new:nnn { enumext } { keyanspic-nested }
4084 {
4085     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4086 }
4087 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4088 {
4089     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4090     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4091 }

```

Messages used by `\getkeyans` command.

```

4092 \msg_new:nnn { enumext } { undefined-storage-anskey }
4093 {
4094     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4095 }

```

Messages used by `\miniright` command.

```

4096 \msg_new:nnn { enumext } { missing-miniright }
4097 {
4098     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4099     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4100 }
4101 \msg_new:nnn { enumext } { wrong-miniright-place }
4102 {
4103     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4104     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4105 }
4106 \msg_new:nnn { enumext } { wrong-miniright-use }
4107 {
4108     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4109     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4110 }

```

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

```
4111 \msg_new:nnn { enumext } { nested }
4112 {
4113   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4114 }
4115 \msg_new:nnn { enumext } { item-joined }
4116 {
4117   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4118 }
4119 \msg_new:nnn { enumext } { item-joined-columns }
4120 {
4121   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4122 }
```

10.42 Finish package

Finish package implementation.

```
4123 \file_input_stop:
4124 \</package>
```

11 Index of Implementation

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

Symbols	
<code>*</code>	425
<code>\+</code>	198
<code>\-</code>	198
<code>\\</code> 206, 2950, 3974, 3987, 3992, 3997, 4012, 4035, 4036, 4051, 4056, 4061, 4066, 4075, 4080, 4089, 4098, 4103, 4108	
A	
<code>above</code>	<u>1236</u>
<code>above*</code>	<u>1236</u>
<code>\addvspace</code> .. 883, 911, <u>1034</u> , 1113, 1176, 1182, 1210, 1227, 2772, 2794, 2906, 2921, 3145, 3152, 3556, 3563	
<code>after</code>	<u>721</u>
<code>align</code>	<u>379</u>
<code>\Alph</code>	31, 35
<code>\Alph</code>	331, 508, 526, 539, 3850
<code>\alph</code>	31, 35
<code>\alph</code>	332, 506, 3840
<code>\anskey</code>	11, 62, <u>1897</u>
<code>\anspic</code>	13, 83, 84, <u>2928</u>
<code>\arabic</code>	31, 32
<code>\arabic</code>	330, 505, 525, 3835, 3855
B	
<code>\b</code>	2650, 2663, 3220, 3233
<code>\baselineskip</code>	43
<code>\baselineskip</code>	1857, 1865
<code>before</code>	<u>721</u>
<code>before*</code>	<u>721</u>
<code>below</code>	<u>1236</u>
<code>below*</code>	<u>1236</u>
bool commands:	
<code>\bool_gset_false:N</code> .. 2810, 2811, 3154, 3158, 3257, 3450, 3451, 3565	
<code>\bool_gset_true:N</code> 226, 237, 825, 2779, 3137, 3155, 3548, 3566	
<code>\bool_if:NTF</code> . 271, 283, 300, 1258, 1272, 1285, 1296, 1307, 1318, 1329, 1340, 1381, 1391, 1461, 1478, 1484, 1501, 1506, 1530, 1535, 1543, 1550, 1581, 1589, 1600, 1624, 1695, 1705, 1819, 1843, 1850, 1878, 1909, 1922, 1924, 1935, 1955, 2080, 2091, 2095, 2134, 2149, 2222, 2241, 2245, 2358, 2388, 2461, 2477, 2538, 2548, 2578, 2583, 2640, 2648, 2661, 2706, 2756, 2770, 2785, 2806, 2835, 2891, 2904, 2912, 2930, 3133, 3142, 3146, 3210, 3218, 3231, 3263, 3273, 3356, 3361, 3369, 3373, 3388, 3417, 3446, 3544, 3553, 3557, 3695, 3699, 3723, 3732, 3736, 3742, 3756, 3778	
<code>\bool_if:nTF</code> 1211, 1228, 1963, 2399, 2433, 2497, 2951, 3875	
<code>\bool_if_p:N</code> 1561, 1562, 1570, 1571, 1674, 1946, 1989, 1990, 2014, 2023, 2024, 2036, 2052, 2208, 2209, 2255, 2256, 2617, 2679, 2692, 2694, 2776, 2957, 2958, 3194	
<code>\bool_lazy_all:nTF</code> .. 1672, 2012, 2021, 2034, 2050, 2615, 2677, 2690, 3192	
<code>\bool_lazy_and:nnTF</code> .. 222, 233, 1560, 1569, 1945, 1988, 2207, 2775	
<code>\bool_lazy_or:nnTF</code>	2254, 2956
<code>\bool_new:N</code> 25, 26, 27, 28, 29, 30, 31, 40, 50, 71, 76, 77, 82, 83, 86, 106, 108, 110, 113, 114, 123, 124, 125, 126, 135, 136, 161, 172, 174	
<code>\bool_not_p:n</code> 223, 234, 1947, 2039, 2054, 2680, 2681, 2693, 2777	
<code>\bool_set_eq:NN</code>	2366, 2414, 3306, 3650
<code>\bool_set_false:N</code> 280, 1656, 1657, 2799, 2842, 2924, 2988, 3005, 3258, 3303, 3600, 3647	
<code>\bool_set_true:N</code> 262, 266, 372, 649, 1242, 1247, 1409, 1410, 1517, 1781, 1788, 2362, 2392, 2410, 2422, 2614, 2623, 2686, 2699, 2725, 2840, 2866, 3122, 3191, 3200, 3312, 3319, 3320, 3533, 3656, 3663, 3664	
box commands:	
<code>\box_dp:N</code> . 930, 934, 938, 949, 953, 964, 973, 979, 989, 1002, 1008, 1014, 1045, 1046, 1047, 1050, 1060, 1064, 1073, 1080, 1085, 1093, 1122, 1123, 1126, 1133, 1146, 1154, 1160, 1168, 3017	
<code>\box_new:N</code>	47, 167
<code>\box_set_wd:Nn</code>	3409, 3770
<code>\box_use:N</code>	3416, 3777
<code>\box_wd:N</code>	338
C	
<code>\c</code>	425, 426, 549, 551, 563, 565
<code>\cB</code>	426
<code>\cE</code>	426
<code>\centering</code>	1213, 1230, 3038, 3148, 3559
<code>check-ans</code>	<u>1649</u>
Document class:	
<code>article</code>	36
clist commands:	
<code>\clist_const:Nn</code>	179
<code>\clist_map_function:nN</code>	3025
<code>\clist_map_inline:Nn</code> . 378, 591, 654, 720, 735, 816, 1252	
<code>\clist_map_inline:nn</code> . 36, 55, 61, 73, 85, 112, 144, 158, 178, 214, 403, 420, 659, 831, 1427, 1662, 1758, 1776, 1797, 2009, 2143, 2316, 2526, 2529, 2555, 2565, 2568, 2588	
<code>\columnbreak</code>	63
<code>\columnbreak</code>	1949
<code>columns</code>	<u>800</u>
<code>columns*</code>	<u>1777</u>
<code>columns-sep</code>	<u>800</u>
<code>columns-sep*</code>	<u>1777</u>
<code>\columnsep</code>	79, 82
<code>\columnsep</code>	2750, 2888
<code>\columnseprule</code>	79, 82
<code>\columnseprule</code>	2754, 2890
Commands provide by enumext :	
<code>\anskey</code> 24, 25, 51, 57, 60, 62, 64–66, 68, 78, 91, 101, 102, 106	
<code>\anspic*</code>	24, 66–68, 84–86, 101, 102
<code>\anspic</code>	60, 83–86, 106
<code>\getkeyans</code>	60, 101, 106
<code>\item*</code>	24, 60, 66–68, 71, 72, 92, 99, 101, 102
<code>\itemwidth</code>	87, 94, 95
<code>\item</code>	70, 72, 87, 91, 92, 95, 98
<code>\miniright</code>	24, 40, 48, 49, 79, 80, 82, 83, 106
<code>\printkeyans</code>	25, 60, 102
<code>\setenumext</code>	24, 103, 104
Counters defined by enumext :	
<code>enumXiii</code>	23, 30

enumXii 23, 30
enumXiv 23, 30
enumXi 23, 30
enumXviii 23, 30
enumXvii 23, 30, 92
enumXvi 23, 30
enumXv 23, 30

cs commands:

\cs_generate_variant:Nn 340, 356, 555, 571, 1802, 1811, 1816, 1896, 2516, 3027
\cs_if_exist:NTF 310
\cs_new:Nn 192
\cs_new:Npn 202, 1428, 1437, 1446
\cs_new_eq:NN 246, 247, 248, 252, 253, 285, 286, 289, 290
\cs_new_protected:Nn . 216, 257, 421, 441, 473, 736, 740, 744, 748, 752, 756, 760, 764, 768, 772, 776, 780, 784, 788, 792, 796, 832, 844, 868, 885, 896, 920, 995, 1019, 1036, 1098, 1115, 1137, 1172, 1178, 1253, 1267, 1281, 1292, 1303, 1314, 1325, 1336, 1379, 1389, 1399, 1528, 1541, 1558, 1579, 1598, 1667, 1703, 1710, 1817, 1841, 1848, 1876, 1883, 2000, 2132, 2147, 2175, 2205, 2250, 2262, 2269, 2321, 2325, 2344, 2395, 2429, 2445, 2455, 2471, 2608, 2675, 2704, 2711, 2734, 2764, 2783, 2833, 2856, 2874, 2899, 2910, 2947, 2990, 3003, 3023, 3028, 3044, 3112, 3131, 3183, 3245, 3252, 3261, 3271, 3288, 3428, 3455, 3523, 3542, 3592, 3613, 3619, 3632, 3688, 3789
\cs_new_protected:Npn 184, 188, 293, 308, 325, 335, 341, 429, 448, 542, 556, 1200, 1219, 1357, 1368, 1457, 1476, 1491, 1515, 1803, 1812, 1932, 2077, 2089, 2111, 2185, 2227, 2235, 2354, 2372, 2406, 2418, 2485, 2519, 2558, 2626, 2646, 2852, 2998, 3063, 3203, 3216, 3294, 3301, 3317, 3325, 3330, 3342, 3474, 3606, 3638, 3645, 3661, 3669, 3683, 3808, 3821, 3868, 3946, 3958
\cs_new_protected_nopar:Nn ... 3281, 3404, 3625, 3765
\cs_new_protected_nopar:Npn 3348, 3715
\cs_set:Nn 2082
\cs_set:Npn 2010, 2048, 3814
\cs_set_eq:NN .. 3173, 3174, 3350, 3582, 3583, 3717
\cs_set_protected:Nn 208, 660, 676, 688, 700
\cs_set_protected:Npn 32, 48, 56, 68, 74, 102, 140, 152, 159, 210, 357, 379, 408, 489, 509, 572, 592, 636, 655, 712, 721, 800, 817, 1236, 1416, 1649, 1723, 1759, 1777, 2002, 2136, 2305, 2517, 2556
\cs_to_str:N 327, 350

D

\d 198
\DeclareDocumentEnvironment 913

dim commands:

\dim_abs:n 2490, 2495
\dim_add:Nn 3008
\dim_compare:nNnTF . 662, 678, 690, 702, 1202, 1221, 2487, 2492, 2498, 2504, 2506, 2508, 2716, 2739, 2860, 2878, 3000, 3046, 3114, 3457, 3525
\dim_compare:nTF 1973
\dim_gset_eq:NN 3123, 3534
\dim_gzero:N 3157, 3568
\dim_new:N 44, 51, 52, 53, 70, 96, 109, 119, 168, 169, 175
\dim_set:Nn .. 338, 650, 1789, 2386, 2490, 2495, 2497, 2500, 2501, 2505, 2507, 2510, 2511, 2513, 2719, 2742, 2862, 2880, 3030, 3048, 3055, 3098, 3116, 3344, 3459, 3466, 3509, 3527

\dim_set_eq:NN 496, 516, 532, 536, 2381, 2528, 2567, 2665, 2750, 2888, 3105, 3108, 3109, 3235, 3335, 3516, 3519, 3520
\dim_use:N 663, 671, 1203, 1209, 1886, 1889, 1894, 2450, 2452, 2717, 2722, 2723, 2730, 2740, 2744, 2745, 2747
\dim_zero:N 2754, 2890, 3009, 3010, 3011
\dim_zero_new:N 3061, 3472
\c_zero_dim 665, 679, 691, 703, 1203, 1221, 1975, 2487, 2492, 2498, 2505, 2717, 2740, 2860, 2878, 3046, 3114, 3457, 3525

E

\end .. 1206, 1224, 1845, 1880, 2769, 2793, 2903, 2920, 3135, 3151, 3546, 3562, 3879, 3884
\endlist 28
\endlist 247
\endlrbox 3407, 3768
\endminipage 28
\endminipage 253
enumext 5, 2589
enumext internal commands:

\g__enumext_ __enumext_store_name_tl
 _prop 68
__enumext_add_pre_parsep: ... 41, 842, 844, 844
__enumext_after_args_exec: . 39, 736, 748, 2601
__enumext_after_args_exec_v: . 39, 40, 752, 764, 2826
__enumext_after_args_exec_vii: ... 768, 792
__enumext_after_args_exec_viii: 796
__enumext_after_env:n 80
__enumext_after_env:nn .. 81, 94, 188, 188, 2802, 3140, 3442, 3551
__enumext_after_hyperref: ... 29, 255, 257, 257
__enumext_after_list: 80, 90, 97, 2606, 2783, 2783
\l__enumext_after_list_args_v_tl 766
\l__enumext_after_list_args_vii_tl 794, 3398
\l__enumext_after_list_args_viii_tl 798, 3752
__enumext_after_list_v: .. 83, 2831, 2910, 2910
__enumext_after_list_vii: ... 3181, 3252, 3252
__enumext_after_list_viii: .. 3590, 3619, 3619
__enumext_after_star_env:nn 88
__enumext_after_stop_list: ... 39, 40, 736, 744, 2797
__enumext_after_stop_list_v: 39, 752, 760, 2925
\l__enumext_after_stop_list_v_tl 762
__enumext_after_stop_list_vii: 768, 784, 3255
\l__enumext_after_stop_list_vii_tl ... 786
__enumext_after_stop_list_viii: . 788, 3622
\l__enumext_after_stop_list_viii_tl ... 790
\l__enumext_align_label_vii_str .. 3390, 3394
\l__enumext_align_label_viii_str . 3744, 3748
\l__enumext_align_label_X_str 159
\c__enumext_all_envs_clist .. 179, 378, 591, 654, 720, 735, 816, 1252
\c__enumext_all_families_seq .. 103, 3908, 3917, 3939
__enumext_anskey_wrapper:n 1727, 2087
__enumext_at_begin_document:n .. 28, 184, 184, 244, 250
__enumext_before_args_exec: 38, 736, 736, 2714
__enumext_before_args_exec_v: .. 39, 752, 752, 2859
__enumext_before_args_exec_vii: .. 768, 768, 3249
__enumext_before_args_exec_viii: 772, 3616

```

\__enumext_before_keys_exec: 39, 736, 740, 2599
\__enumext_before_keys_exec_v: .. 39, 752, 756,
    2824
\__enumext_before_keys_exec_vii ..... 768
\__enumext_before_keys_exec_vii: 40, 776, 3169
\__enumext_before_keys_exec_viii: .. 40, 780,
    3578
\__enumext_before_list: ... 78, 2593, 2711, 2711
\__enumext_before_list_v: . 82, 2819, 2856, 2856
\__enumext_before_list_vii: 90, 3164, 3245, 3245
\__enumext_before_list_viii: .. 97, 3574, 3613,
    3613
\l__enumext_before_no_starred_key_v_tl 758
\l__enumext_before_no_starred_key_vii_-
    tl ..... 778
\l__enumext_before_no_starred_key_viii_-
    tl ..... 782
\l__enumext_before_starred_key_v_tl ... 754
\l__enumext_before_starred_key_vii_tl . 770
\l__enumext_before_starred_key_viii_tl 774
\__enumext_calc_hspace:NNNNNNN 74, 2485, 2485,
    2516, 2521, 2560
\l__enumext_check_ans_bool ... 70, 71, 123, 1653,
    1657, 1705, 1924, 2222, 2358, 2388, 2776, 3361
\__enumext_check_ans_exec: .. 58, 78, 1703, 1703,
    2715, 3248
\g__enumext_check_ans_item_tl .. 68, 123, 2221,
    2229, 2233
\__enumext_check_ans_set: . 57, 1667, 1667, 1707
\__enumext_check_ans_show: 58, 1710, 1710, 2808,
    3448
\g__enumext_check_ans_show_bool 80, 123, 2779,
    2806, 2811
\g__enumext_check_ans_show_h_bool 123, 3446,
    3451
\l__enumext_columns_sep_v_dim 2878, 2880, 2888
\l__enumext_columns_sep_vii_dim .. 3046, 3048,
    3057, 3102, 3237, 3426
\l__enumext_columns_sep_viii_dim . 3457, 3459,
    3468, 3513, 3787
\l__enumext_columns_v_int 1041, 2876, 2884, 2896,
    2901
\l__enumext_columns_vii_int .. 3051, 3054, 3058,
    3066, 3070, 3073, 3079, 3085, 3089, 3224, 3421, 3432
\l__enumext_columns_viii_int . 3462, 3465, 3469,
    3477, 3481, 3484, 3490, 3496, 3500, 3782, 3793
\g__enumext_count_item_anskey_int .. 68, 123,
    1713, 1721, 1926, 2224
\g__enumext_count_item_number_int 123, 1678,
    1683, 1686, 1689, 1697, 1713, 1720, 2360, 2390, 3363
\g__enumext_count_item_with_ans_int .... 62
\l__enumext_counter_i_tl ..... 32, 317
\l__enumext_counter_ii_tl ..... 32, 318
\l__enumext_counter_iii_tl ..... 32, 319
\l__enumext_counter_iv_tl ..... 32, 320
\l__enumext_counter_style_for_ref_vii_-
    tl ..... 456, 466, 477, 479
\l__enumext_counter_style_for_ref_viii_-
    tl ..... 483, 485
\l__enumext_counter_style_for_ref_X_tl 148
\c__enumext_counter_style_tl .... 32, 148, 423
\g__enumext_counter_styles_tl . 23, 31, 44, 328,
    346
\l__enumext_counter_v_tl ..... 32, 321
\l__enumext_counter_vi_tl ..... 32, 322
\l__enumext_counter_vii_tl ..... 32, 323, 453
\l__enumext_counter_viii_tl ..... 32, 324, 463
\__enumext_current_env_set_bool: 27, 216, 216,
    2610, 3185
\l__enumext_current_widest_dim 23, 44, 352, 497,
    517, 533, 537
\__enumext_default_item:n ... 2354, 2354, 2403
\__enumext_define_counters:Nn 23, 308, 308, 317,
    318, 319, 320, 321, 322, 323, 324
\__enumext_endminipage: . 28, 250, 253, 919, 3040,
    3406, 3767
\__enumext_fake_item: ..... 660, 660, 2547
\l__enumext_fake_item_indent_v_dim 679, 684
\l__enumext_fake_item_indent_v_tl 681, 2411,
    2415, 2423
\l__enumext_fake_item_indent_vii_dim 691, 696
\l__enumext_fake_item_indent_vii_tl 693, 3402
\l__enumext_fake_item_indent_viii_dim . 703,
    708, 3759
\l__enumext_fake_item_indent_viii_tl .. 705,
    3758, 3762
\l__enumext_fake_item_indent_X_tl ..... 74
\__enumext_fake_item_vii: .... 660, 688, 2577
\__enumext_fake_item_viii: .... 660, 700, 2582
\__enumext_filter_series:n 53, 1428, 1428, 1469,
    1482, 1488
\__enumext_filter_series_key:n 53, 1428, 1433,
    1437
\__enumext_filter_series_pair:nn .. 53, 1428,
    1434, 1446
\g__enumext_footnote_arg_seq . 145, 2327, 2340,
    2350
\g__enumext_footnote_int . 145, 2334, 2337, 2339,
    2341
\g__enumext_footnote_int_seq . 145, 2328, 2341,
    2346, 2349
\__enumext_footnotes_key_bool ..... 29
\l__enumext_footnotes_key_bool 25, 29, 93, 135,
    266, 271, 280, 3369, 3417, 3732, 3778
\__enumext_footnotetext:nn ... 2321, 2321, 2351
\__enumext_getkeyans:nn .. 102, 3817, 3821, 3821
\__enumext_getkeyans_aux:n 101, 3805, 3808, 3808
\l__enumext_hyperref_bool 25, 29, 135, 262, 283,
    300, 1990, 2209, 3356, 3723
\__enumext_hypertarget:nn 29, 257, 285, 289, 305
\__enumext_if_is_int:n ..... 196
\__enumext_if_is_int:nTF ..... 196, 544, 558
\l__enumext_item_column_pos_vii_int 91, 3073,
    3079, 3085, 3089, 3096, 3284, 3421, 3424
\l__enumext_item_column_pos_viii_int ... 98,
    3484, 3490, 3496, 3500, 3507, 3628, 3782, 3785
\l__enumext_item_column_pos_X_int ..... 159
\g__enumext_item_count_all_vii_int 91, 3097,
    3285, 3432, 3439
\g__enumext_item_count_all_viii_int 98, 3508,
    3629, 3793, 3800
\g__enumext_item_count_all_X_int ..... 159
\__enumext_item_peek_args_vii: 91, 3286, 3288,
    3288
\__enumext_item_peek_args_viii: 98, 3630, 3632,
    3632
\__enumext_item_starred: .. 73, 2445, 2445, 2463
\l__enumext_item_starred_vii_bool 3303, 3319,

```


3373
 \l__enumext_item_starred_viii_bool 3647, 3663, 3736, 3756
 \l__enumext_item_starred_X_bool 159
 __enumext_item_std:w 28, 70–72, 86, 244, 248, 2363, 2369, 2393, 2411, 2415, 2423, 3021
 \g__enumext_item_symbol_aux_vii_tl 3327, 3375, 3378, 3382, 3384
 \g__enumext_item_symbol_aux_X_tl 159
 \l__enumext_item_symbol_sep_vii_dim . . 3336, 3344, 3381, 3383
 \g__enumext_item_symbol_tl 23, 71, 37, 2378, 2451, 2468
 \l__enumext_item_symbol_vii_tl 3378
 \l__enumext_item_text_vii_box 3368, 3409, 3416
 \l__enumext_item_text_viii_box 3731, 3770, 3777
 \l__enumext_item_text_X_box 159
 \l__enumext_item_width_vii_dim . . 3055, 3100, 3108, 3109
 \l__enumext_item_width_viii_dim . . 3466, 3511, 3519, 3520
 \l__enumext_item_width_X_dim 159
 \l__enumext_itemindent_X_dim 48
 \l__enumext_itemsep_vii_skip 3438
 \l__enumext_itemsep_viii_skip 3799
 \l__enumext_joined_item_aux_vii_int . . 3094, 3095, 3096, 3097, 3103
 \l__enumext_joined_item_aux_viii_int . 3505, 3506, 3507, 3508, 3514
 \l__enumext_joined_item_aux_X_int . . . 159
 __enumext_joined_item_vii:w . . 91, 3291, 3292, 3294, 3294
 \l__enumext_joined_item_vii_int . . 3065, 3066, 3069, 3071, 3077, 3082, 3087, 3092, 3094, 3100
 __enumext_joined_item_viii:w . 98, 3635, 3636, 3638, 3638
 \l__enumext_joined_item_viii_int . 3476, 3477, 3480, 3482, 3488, 3493, 3498, 3503, 3505, 3511
 \l__enumext_joined_item_X_int 159
 \l__enumext_joined_width_vii_dim . 3098, 3105, 3108, 3399, 3411
 \l__enumext_joined_width_viii_dim 3509, 3516, 3519, 3753, 3772
 \l__enumext_joined_width_X_dim 159
 __enumext_keyans_addto_prop:n 66, 2111, 2111, 2425, 2953
 __enumext_keyans_addto_seq:n . 67, 2185, 2185, 2427, 2955
 __enumext_keyans_addto_seq_link: 2185, 2203, 2205, 3694
 __enumext_keyans_anspic_code:nnn . 84, 2944, 2947, 2947
 __enumext_keyans_check_ans:nn 68, 2227, 2227, 2829, 2985, 3588
 __enumext_keyans_default_item:n . . 72, 2406, 2406, 2441
 \l__enumext_keyans_env_bool 20, 2680, 2693, 2840, 2924
 __enumext_keyans_fake_item: . . 660, 676, 2537
 \l__enumext_keyans_item_opt_tl 86, 2239, 2252, 2258, 3679
 \l__enumext_keyans_level_h_int 20, 2163, 3594, 3595
 \l__enumext_keyans_level_int . . 20, 1194, 1913, 2158, 2839, 2843, 2938
 __enumext_keyans_make_label: 31, 73, 2471, 2471, 2536
 __enumext_keyans_mini_addvspace: 46, 82, 1098, 1098, 2868
 __enumext_keyans_mini_right_cmd:n 49, 1196, 1219, 1219
 __enumext_keyans_mini_set_vskip: . 45, 1036, 1036, 1100
 __enumext_keyans_multi_addvspace: . 83, 885, 896, 2893
 __enumext_keyans_multi_set_vskip: . 42, 885, 885, 898
 __enumext_keyans_multicols_start: 82, 2872, 2874, 2874
 __enumext_keyans_multicols_stop: . 83, 1223, 2899, 2899, 2923
 __enumext_keyans_parse_keys:n 2818, 2852, 2852
 \l__enumext_keyans_pic_above_int . 118, 3031, 3032, 3034
 \l__enumext_keyans_pic_above_skip . . 86, 118, 2976, 3015
 __enumext_keyans_pic_arg_two: 85, 2974, 3003, 3003
 \l__enumext_keyans_pic_below_int . 118, 3031, 3032, 3035
 \l__enumext_keyans_pic_body_seq . . 84–86, 118, 2942, 2981, 3039
 __enumext_keyans_pic_do:n 86, 2981, 2983, 3023, 3023, 3027
 \l__enumext_keyans_pic_level_int . . 20, 1186, 1917, 2114, 2153, 2188, 2271, 2992, 2993
 __enumext_keyans_pic_row:n 86, 3025, 3028, 3028
 __enumext_keyans_pic_safe_exec: . . 85, 2970, 2990, 2990
 __enumext_keyans_pic_skip_abs:N . . 85, 2998, 2998, 3014
 \l__enumext_keyans_pic_width_dim . 118, 3030, 3037
 __enumext_keyans_redefine_item: . . 72, 2429, 2429, 2535
 __enumext_keyans_safe_exec: . 2817, 2833, 2833
 __enumext_keyans_show_ans: . . 2235, 2243, 2262
 __enumext_keyans_show_item_opt: . 2235, 2250, 2423, 2966, 3759
 __enumext_keyans_show_left:n . 72, 2235, 2235, 2421, 2961
 __enumext_keyans_show_pos: . . 2235, 2247, 2269
 __enumext_keyans_starred_item:n . . 72, 2418, 2418, 2437
 __enumext_keyans_store_ref: . . 66, 2132, 2132, 2426, 2954, 3692
 __enumext_keyans_store_ref_aux_i: 67, 2132, 2144, 2147
 __enumext_keyans_store_ref_aux_ii: 67, 2132, 2173, 2175
 \l__enumext_keyans_tmpa_dim 86
 \l__enumext_keyans_tmpa_tl 24, 99, 86, 2420, 2424
 \l__enumext_keyans_tmpb_tl 99, 86
 __enumext_keyans_wrapper_opt:n . . 1730, 2258
 \l__enumext_label_copy_i_tl . . 2044, 2151, 2156, 2161, 2166
 \l__enumext_label_copy_v_tl 2161
 \l__enumext_label_copy_vi_tl 2156

`\l__enumext_label_copy_vii_tl` 2019, 2030, 2061, 2151
`\l__enumext_label_copy_viii_tl` 2166
`\l__enumext_label_copy_X_tl` 137
`\l__enumext_label_fill_left_v_tl` 2475
`\l__enumext_label_fill_left_X_tl` 74
`\l__enumext_label_fill_right_v_tl` 2482
`\l__enumext_label_fill_right_X_tl` 74
`\l__enumext_label_font_style_v_tl` 2476, 2965
`\l__enumext_label_font_style_vii_tl` . . . 3387
`\l__enumext_label_font_style_viii_tl` . . 3741
`\l__enumext_label_i_tl` 489
`\l__enumext_label_ii_tl` 489
`\l__enumext_label_iii_tl` 489
`\l__enumext_label_iv_tl` 489
`__enumext_label_style:Nnn` 23, 31, 341, 341, 356, 494, 514, 530, 534
`\l__enumext_label_v_tl` . . 66, 67, 527, 2119, 2193, 2264, 2298, 2420, 2424, 2821, 2960, 2962
`\l__enumext_label_vi_tl` . 66, 67, 527, 2116, 2190, 2960, 2962, 2966
`\l__enumext_label_vii_tl` . 509, 3314, 3339, 3346
`\l__enumext_label_viii_tl` 509, 3658, 3686, 3690
`\l__enumext_label_width_by_box` . . 44, 337, 338
`__enumext_label_width_by_box:Nn` 31, 335, 335, 340, 352, 568
`\l__enumext_labelsep_i_dim` . . . 2266, 2302, 3697, 3712
`\l__enumext_labelsep_v_dim` 2883
`\l__enumext_labelsep_vii_dim` . 3050, 3059, 3101, 3337, 3397, 3413
`\l__enumext_labelsep_viii_dim` 3461, 3470, 3512, 3751, 3774
`\l__enumext_labelwidth_i_dim` . 2266, 2301, 3697, 3712
`\l__enumext_labelwidth_v_dim` 2883
`\l__enumext_labelwidth_vii_dim` . . . 3050, 3058, 3101, 3390, 3394, 3412
`\l__enumext_labelwidth_viii_dim` . . 3461, 3469, 3512, 3744, 3748, 3773
`\l__enumext_leftmargin_tmp_v_bool` . 85, 3005
`\l__enumext_leftmargin_tmp_X_bool` 48
`\l__enumext_leftmargin_tmp_X_dim` 48
`\l__enumext_leftmargin_X_dim` 48
`__enumext_level:` 192, 192, 432, 434, 435, 443, 445, 663, 667, 671, 738, 742, 746, 750, 834, 836, 838, 840, 873, 875, 877, 879, 883, 923, 926, 945, 954, 960, 965, 969, 980, 984, 985, 990, 1026, 1030, 1203, 1209, 1256, 1258, 1260, 1263, 1270, 1272, 1274, 1277, 1821, 1829, 1833, 1837, 2082, 2085, 2086, 2362, 2363, 2367, 2368, 2369, 2376, 2378, 2382, 2383, 2386, 2392, 2393, 2447, 2450, 2452, 2459, 2460, 2461, 2464, 2467, 2596, 2598, 2648, 2653, 2654, 2655, 2657, 2661, 2666, 2667, 2668, 2670, 2686, 2699, 2706, 2717, 2719, 2722, 2723, 2725, 2730, 2737, 2740, 2742, 2744, 2745, 2746, 2747, 2750, 2756, 2761, 2767, 2770, 2772, 2785
`\l__enumext_level_h_int` . 20, 224, 451, 475, 1675, 1692, 2038, 2055, 2619, 3186, 3187, 3195
`\l__enumext_level_int` 20, 194, 235, 846, 997, 1190, 1669, 2015, 2025, 2031, 2037, 2045, 2053, 2060, 2550, 2611, 2612, 2618, 2631, 2638, 2684, 2697, 2752, 2804, 2847, 2934, 3196, 3265, 3275, 3444, 3601
`__enumext_list_arg_two_i:` 2517
`__enumext_list_arg_two_ii:` 2517
`__enumext_list_arg_two_iii:` 2517
`__enumext_list_arg_two_iv:` 2517
`__enumext_list_arg_two_v:` . 72, 2517, 2823, 3006
`__enumext_list_arg_two_vii:` 2556, 3168
`__enumext_list_arg_two_viii:` 2556, 3577
`\l__enumext_listoffset_v_dim` 2885
`\l__enumext_listparindent_vii_dim` 3400
`\l__enumext_listparindent_viii_dim` . . . 3754
`__enumext_make_label:` 31, 71, 73, 2455, 2455, 2545
`\l__enumext_mark_answer_sym_tl` . 61, 113, 1736, 1891, 2097, 2273, 2286, 3701
`\l__enumext_mark_position_str` 113, 1740, 1741, 1764, 1765, 1889
`\l__enumext_mark_ref_sym_tl` . . 113, 1750, 1995, 2217
`__enumext_mini_addvspace:` . . 45, 79, 1019, 1019, 2727
`__enumext_mini_addvspace_vii:` 47, 1172, 1172, 3126
`__enumext_mini_addvspace_viii:` 47, 1172, 1178, 3537
`__enumext_mini_env*` 913
`__enumext_mini_right_cmd:n` . 48, 49, 1198, 1200, 1200
`__enumext_mini_set_vskip:` . . 43, 920, 920, 1021
`__enumext_mini_set_vskip_vii:` 47, 1115, 1115, 1174
`__enumext_mini_set_vskip_viii:` 47, 1115, 1137, 1180
`__enumext_minipage:w` 28, 250, 252, 915, 3037, 3399, 3753
`\l__enumext_minipage_active_v_bool` . . 82, 83, 2866, 2891, 2904, 2912
`\g__enumext_minipage_active_vii_bool` . . . 88, 3137, 3142, 3154
`\l__enumext_minipage_active_vii_bool` . 3122, 3133
`\g__enumext_minipage_active_viii_bool` 3548, 3553, 3565
`\l__enumext_minipage_active_viii_bool` 3533, 3544
`\g__enumext_minipage_active_X_bool` . . . 159
`\l__enumext_minipage_active_X_bool` 62
`\g__enumext_minipage_after_skip` 62, 1119, 1131, 3152, 3563
`\l__enumext_minipage_after_skip` 43, 44, 80, 83, 62, 936, 951, 971, 987, 1002, 1008, 1014, 1028, 1038, 1047, 1050, 1062, 1080, 1091, 1107, 1139, 1152, 1166, 2794, 2921
`\g__enumext_minipage_center_vii_bool` . 3146, 3155
`\g__enumext_minipage_center_viii_bool` 3557, 3566
`\g__enumext_minipage_center_X_bool` . . . 159
`\l__enumext_minipage_hsep_v_dim` . . . 82, 2864
`\l__enumext_minipage_hsep_vii_dim` . . . 3120
`\l__enumext_minipage_hsep_viii_dim` . . . 3531
`\l__enumext_minipage_left_skip` 43, 82, 62, 928, 943, 962, 977, 1024, 1034, 1039, 1045, 1054, 1071, 1083, 1103, 1113, 1117, 1122, 1126, 1140, 1144, 1158, 1176, 1182
`\l__enumext_minipage_left_v_dim` 82, 2862, 2870
`\l__enumext_minipage_left_vii_dim` 3116, 3128
`\l__enumext_minipage_left_viii_dim` 3527, 3539

`\l__enumext_minipage_left_X_dim` 62
`\g__enumext_minipage_right_skip` 62, 1118, 1123, 1127, 3145, 3556
`\l__enumext_minipage_right_skip` . . 43, 62, 932, 947, 967, 982, 1040, 1046, 1058, 1076, 1087, 1141, 1148, 1162, 1210, 1227
`\l__enumext_minipage_right_v_dim` . . 82, 1221, 1226, 2860, 2864
`\g__enumext_minipage_right_vii_dim` 88, 3124, 3144, 3157
`\l__enumext_minipage_right_vii_dim` 88, 3114, 3119, 3125
`\g__enumext_minipage_right_viii_dim` . . 3535, 3555, 3568
`\l__enumext_minipage_right_viii_dim` . . 3525, 3530, 3536
`\g__enumext_minipage_right_X_dim` 159
`\g__enumext_minipage_right_X_skip` 159
`\g__enumext_minipage_stat_int` . 79, 82, 62, 1215, 1232, 2726, 2787, 2792, 2867, 2914, 2919
`\g__enumext_miniright_code_vii_tl` . 88, 3150, 3156
`\g__enumext_miniright_code_viii_tl` 3561, 3567
`\g__enumext_miniright_code_X_tl` 159
`__enumext_multi_addvspace`: . . . 42, 79, 868, 868, 2758
`__enumext_multi_set_vskip`: . . 41, 832, 832, 870
`\l__enumext_multicols_above_ii_skip` . . . 851
`\l__enumext_multicols_above_iii_skip` . . . 857
`\l__enumext_multicols_above_iv_skip` . . . 863
`\l__enumext_multicols_above_v_skip` 887, 901, 911
`\l__enumext_multicols_above_X_skip` 56
`\l__enumext_multicols_below_v_skip` 891, 905, 2906
`\l__enumext_multicols_below_X_skip` 56
`__enumext_multicols_start`: 79, 2732, 2734, 2734
`__enumext_multicols_stop`: 80, 1205, 2764, 2764, 2796
`__enumext_newlabel:nn` 25, 29, 65, 293, 293, 2071, 2179
`\l__enumext_newlabel_arg_one_tl` 25, 29, 65, 67, 137, 1994, 2064, 2072, 2168, 2180, 2215
`\l__enumext_newlabel_arg_two_tl` 25, 29, 64, 137, 2018, 2028, 2042, 2058, 2073, 2155, 2160, 2165, 2181
`__enumext_parse_keys:n` 2592, 2626, 2626
`__enumext_parse_keys_parse_keys:n` 53
`__enumext_parse_keys_vii:n` 53, 3163, 3203, 3203
`__enumext_parse_keys_viii:n` . 3573, 3606, 3606
`__enumext_parse_series_name:n` 53, 1457, 1457, 2634, 3209
`__enumext_parse_store_keys:n` . 77, 2642, 2646, 2646
`__enumext_parse_store_keys_vii:n` 89, 90, 3212, 3216, 3216
`\l__enumext_parsep_i_skip` 849, 851, 1000, 1048
`\l__enumext_parsep_ii_skip` 855, 857, 1006
`\l__enumext_parsep_iii_skip` 861, 863, 1012
`\l__enumext_parsep_vii_skip` 3401
`\l__enumext_parsep_viii_skip` 3755
`\l__enumext_partopsep_v_skip` . . 903, 907, 1074, 1078, 1085, 1089, 1105, 1109
`\l__enumext_partopsep_viii_skip` 1150
`__enumext_phantomsection`: 29, 257, 286, 290, 306
`__enumext_print_footnote`: . . . 2321, 2344, 3419, 3780
`__enumext_print_keyans_box:NN` 61, 1883, 1883, 1896, 2084, 2266, 2300, 3697, 3712
`\l__enumext_print_keyans_i_tl` 3831, 3860
`\l__enumext_print_keyans_ii_tl` 3836, 3861
`\l__enumext_print_keyans_iii_tl` 3841, 3862
`\l__enumext_print_keyans_iv_tl` 3846, 3863
`\l__enumext_print_keyans_vii_tl` 3851, 3864
`\l__enumext_print_keyans_X_tl` 102
`__enumext_printkeyans:nnn` 102, 3865, 3868, 3868
`__enumext_redefine_item`: . 72, 2395, 2395, 2544
`\l__enumext_ref_aux_tl` 148, 432, 434, 437, 453, 455, 458, 463, 465, 468
`\l__enumext_ref_key_arg_tl` . . 148, 426, 431, 438, 450, 459, 469
`__enumext_regex_label_ref_key`: . . 32, 33, 421, 421, 433, 454, 464
`__enumext_register_counter_style:Nn` . . . 325, 325, 330, 331, 332, 333, 334
`__enumext_remove_extra_parsep_vii`: . . 3178, 3428, 3428
`__enumext_remove_extra_parsep_viii`: . . 3587, 3789, 3789
`__enumext_renew_footnote`: . . . 2321, 2325, 3371, 3734
`\l__enumext_resume_bool` 23
`__enumext_resume_counter`: . 54, 1515, 1521, 1528
`__enumext_resume_counter:n` . . 1495, 1500, 1515, 1515, 1585, 1593
`__enumext_resume_counter_name`: . . 1515, 1524, 1541
`__enumext_resume_counter_set`: . . . 1598, 1598, 2800, 3259
`__enumext_resume_counter_store`: . 1515, 1526, 1558
`\g__enumext_resume_int` 23, 80, 37, 1532, 1533, 1614
`__enumext_resume_last:n` 53, 54, 1457, 1463, 1476
`\l__enumext_resume_name_bool` . . . 37, 1461, 1517
`\l__enumext_resume_name_tl` 37, 1518, 1519, 1547, 1554, 1610, 1618, 1620, 1634, 1642, 1644
`__enumext_resume_series:n` . 54, 1422, 1491, 1491
`__enumext_resume_starred`: . . . 1423, 1579, 1579
`\g__enumext_resume_vii_int` . . 90, 37, 1537, 1538, 1638
`__enumext_safe_exec`: 27, 2591, 2608, 2608
`__enumext_safe_exec_vii`: . 27, 3162, 3183, 3183
`__enumext_safe_exec_viii`: . . . 3572, 3592, 3592
`\l__enumext_series_str` . . 1420, 1459, 1467, 1468, 1470, 1472, 1606, 1608, 1612, 1630, 1632, 1636, 2630, 3207
`__enumext_set_error:nn` 3946, 3956, 3958
`__enumext_set_label_ref:n` . . . 32, 429, 429, 501
`__enumext_set_label_ref_h:n` . 33, 448, 448, 521
`__enumext_set_parse:n` 3929, 3946, 3946
`\l__enumext_setkey_tmpa_int` 97, 3922, 3926, 3938, 3940, 3953
`\l__enumext_setkey_tmpa_tl` 97, 3928, 3932
`\l__enumext_setkey_tmpb_seq` 97, 3921, 3924, 3928, 3929
`\l__enumext_setkey_tmpb_tl` 97, 3948, 3950, 3951
`\l__enumext_show_answer_bool` . 113, 1744, 1768, 2091, 2241, 2255, 2957, 3695

__enumext_show_length:nnn . . 38, 202, 202, 3998,
 3999, 4000, 4001, 4002, 4003, 4004, 4005, 4006, 4007,
 4013, 4014, 4015, 4016, 4017, 4018, 4019, 4020, 4021,
 4022
 \l__enumext_show_position_bool 113, 1747, 1771,
 2095, 2245, 2256, 2958, 3699
 \g__enumext_standar_bool . 27, 20, 223, 226, 1501,
 1530, 1543, 1581, 1600, 1695, 2617, 2810
 \l__enumext_standar_bool . 20, 2023, 2036, 2052,
 2614, 2799
 \g__enumext_standar_keyans_pic_star_env_-
 int 134
 \g__enumext_standar_keyans_star_env_int 133
 \l__enumext_standar_level_one_bool 20, 1381,
 1478, 1561, 2623
 \g__enumext_standar_series_tl . 37, 1481, 1482,
 1583, 1586
 \g__enumext_standar_star_env_int . . 130, 227
 __enumext_standard_item_vii:w . . 91, 92, 3299,
 3301, 3301
 __enumext_standard_item_viii:w 98, 3643, 3645,
 3645
 \g__enumext_starred_bool 27, 89, 90, 20, 234, 237,
 1506, 1535, 1550, 1589, 1624, 1674, 2014, 2024, 2054,
 2149, 2681, 2694, 2777, 3158, 3194, 3257, 3450
 \l__enumext_starred_bool . 89, 90, 20, 1947, 1955,
 2039, 2080, 3191, 3258
 __enumext_starred_columns_set_vii: . . 3044,
 3044, 3171
 __enumext_starred_columns_set_viii: . 3455,
 3455, 3580
 __enumext_starred_item:nn . . 2372, 2372, 2401
 __enumext_starred_item_exec: . 99, 3688, 3688,
 3738
 __enumext_starred_item_vii:w 91, 92, 3298, 3317,
 3317
 __enumext_starred_item_vii_aux_i:w . . 3317,
 3322, 3325
 __enumext_starred_item_vii_aux_ii:w . 3317,
 3323, 3328, 3330
 __enumext_starred_item_vii_aux_iii:w 3317,
 3333, 3342
 __enumext_starred_item_viii:w . . 98, 99, 3642,
 3661, 3661
 __enumext_starred_item_viii_aux_i:w . 3661,
 3666, 3669
 __enumext_starred_item_viii_aux_ii:w 3661,
 3667, 3681, 3683
 __enumext_starred_joined_item_vii:n . 87, 91,
 3063, 3063, 3296
 __enumext_starred_joined_item_viii:n 95, 98,
 3474, 3474, 3640
 \g__enumext_starred_keyans_star_env_int 132
 \l__enumext_starred_level_one_bool 20, 1391,
 1484, 1570, 3200
 \g__enumext_starred_series_tl . 37, 1487, 1488,
 1591, 1594
 \g__enumext_starred_star_env_int . . 131, 238
 __enumext_start_from:NNn 35, 542, 542, 555, 577
 \l__enumext_start_i_int 1533, 1545, 1564
 __enumext_start_item_tmp_vii: 89, 3174, 3281,
 3281
 __enumext_start_item_tmp_viii: 96, 3583, 3625,
 3625
 __enumext_start_item_vii:w 92, 3309, 3314, 3339,
 3346, 3348, 3348
 __enumext_start_item_viii:w . . 98, 3653, 3658,
 3686, 3715, 3715
 __enumext_start_list:nn 28, 75, 85, 244, 246, 2595,
 2820, 2971, 3166, 3575
 __enumext_start_mini_vii: . 90, 3112, 3112, 3250
 __enumext_start_mini_viii: 97, 3523, 3523, 3617
 __enumext_start_store_level: . 78, 2594, 2675,
 2675
 __enumext_start_store_level_vii: . 91, 3165,
 3261, 3261
 \l__enumext_start_vii_int . . . 1538, 1552, 1573
 \l__enumext_start_X_int 74, 572
 __enumext_stop_item_tmp_vii: . 89, 91, 92, 3173,
 3177, 3283, 3350
 __enumext_stop_item_tmp_viii: . . 96, 98, 3582,
 3586, 3627, 3717
 __enumext_stop_item_vii: 92, 93, 3350, 3404, 3404
 __enumext_stop_item_viii: 100, 3717, 3765, 3765
 __enumext_stop_list: . . 28, 244, 247, 2604, 2830,
 2984, 3179, 3589
 __enumext_stop_mini_vii: 88, 90, 3131, 3131, 3254
 __enumext_stop_mini_viii: . 97, 3523, 3542, 3621
 __enumext_stop_store_level: . . 78, 2605, 2675,
 2704
 __enumext_stop_store_level_vii: . . 91, 3180,
 3261, 3271
 \l__enumext_store_active_bool 24, 51, 77, 89, 86,
 1409, 1562, 1571, 1909, 2640, 2679, 2692, 2835, 2842,
 2930, 2988, 3210, 3263, 3273, 3600
 __enumext_store_addto_prop:n 60, 66, 1802, 1803,
 1811, 1934, 2130, 3691
 __enumext_store_addto_seq:n 60, 68, 1812, 1812,
 1816, 1823, 1837, 1845, 1854, 1872, 1880, 1998, 2220
 \l__enumext_store_ans_bool 123, 1410, 1656, 1819,
 1843, 1850, 1878, 1922
 \l__enumext_store_anskey_arg_tl . . 24, 63, 86,
 1940, 1949, 1951, 1957, 1965, 1968, 1978, 1983, 1986,
 1992, 1998
 __enumext_store_anskey_code:nnnn 62, 63, 1928,
 1932, 1932
 __enumext_store_anskey_show_left:n 65, 1939,
 2089, 2089
 __enumext_store_anskey_show_wrap:n 65, 2077,
 2077, 2093, 2108
 \l__enumext_store_columns_break_bool . 1903,
 1946
 \l__enumext_store_columns_join_int 86, 1954,
 1959
 \l__enumext_store_columns_sep_vii_bool 3231
 \l__enumext_store_columns_sep_vii_dim 3236,
 3240
 \l__enumext_store_columns_sep_X_bool . . 102
 \l__enumext_store_columns_sep_X_dim . . . 102
 \l__enumext_store_columns_vii_bool . . . 3218
 \l__enumext_store_columns_vii_int 3223, 3227
 \l__enumext_store_columns_X_bool 102
 \l__enumext_store_columns_X_int 102
 __enumext_store_internal_ref: . . 63, 64, 1937,
 2000, 2000
 \l__enumext_store_item_symbol_sep_dim 1901,
 1975, 1980
 \l__enumext_store_item_symbol_tl . 1899, 1966,
 1970

```

\l__enumext_store_keyans_item_opt_sep_-
    tl . . . . 1733, 2124, 2126, 2197, 2199, 3674, 3676
\l__enumext_store_keyans_item_opt_tl . . 86
\l__enumext_store_keyans_label_tl 24, 66–68,
    86, 2113, 2116, 2119, 2126, 2128, 2130, 2187, 2190,
    2193, 2199, 2201, 2211, 2220, 2221, 3671, 3676, 3677,
    3690, 3691, 3693
\__enumext_store_level_close: . 60, 1817, 1841,
    2708
\__enumext_store_level_close_vii: 1848, 1876,
    3277
\__enumext_store_level_open: . . 59, 60, 77, 1817,
    1817, 2687, 2700
\__enumext_store_level_open_vii: . . 90, 1848,
    1848, 3267
\g__enumext_store_name_tl 24, 80, 86, 1715, 1718,
    2780, 2812, 3452
\l__enumext_store_name_tl 24, 51, 86, 1359, 1360,
    1370, 1371, 1401, 1403, 1405, 1407, 1411, 1413, 1566,
    1575, 1602, 1604, 1626, 1628, 1805, 1807, 1814, 2066,
    2067, 2103, 2170, 2171, 2279, 2292, 2780, 3707
\l__enumext_store_opt_vii_tl . 1852, 1862, 1868,
    1872, 3225, 3238
\l__enumext_store_opt_X_tl . . . . . 102
\l__enumext_store_ref_key_bool 63, 1753, 1935,
    1989, 2134, 2208
\l__enumext_store_upper_level_X_bool . . 102
\l__enumext_store_write_aux_file_tl 25, 65, 67,
    137, 2069, 2075, 2177, 2183
\__enumext_storing_exec: 1357, 1383, 1393, 1399
\__enumext_storing_set:n . . 51, 1349, 1357, 1357
\__enumext_storing_set_vii:n . . . . . 1354, 1368
\__enumext_storing_standar: . . . . . 1365, 1379
\__enumext_storing_starred: . . . . . 1376, 1389
\l__enumext_the_counter_vii_tl . . . . . 455
\l__enumext_the_counter_viii_tl . . . . . 465
\l__enumext_the_counter_X_tl . . . . . 148
\__enumext_tmp:n 32, 36, 48, 55, 56, 61, 68, 73, 74, 85,
    102, 112, 140, 144, 152, 158, 159, 178, 210, 214, 655,
    659, 1416, 1427, 1649, 1666, 1723, 1758, 1759, 1776,
    2002, 2009, 2010, 2031, 2045, 2048, 2060, 2136, 2143,
    2517, 2555, 2556, 2588
\__enumext_tmp:nn 357, 378, 379, 407, 408, 420, 572,
    591, 636, 654, 712, 720, 721, 735, 800, 816, 817, 831,
    1236, 1252, 1777, 1801, 2305, 2320
\__enumext_tmp:nnn 489, 505, 506, 507, 508, 509, 525,
    526
\__enumext_tmp:nnnnn 592, 617, 620, 623, 625, 627,
    630, 633
\__enumext_tmp:w . . . . . 3814, 3817
\l__enumext_tmpa_vii_int . . . . . 3054, 3057
\l__enumext_tmpa_viii_int . . . . . 3465, 3468
\l__enumext_tmpa_X_int . . . . . 159
\l__enumext_topsep_v_skip 889, 893, 1043, 1056,
    1064, 1069, 1089, 1093, 2987, 3018
\l__enumext_topsep_vii_skip . . 1120, 1129, 1133
\l__enumext_topsep_viii_skip . 1142, 1164, 1168
\__enumext_use_key_ref: . . . . 33, 441, 441, 2546
\__enumext_use_key_ref_h: . . 33, 473, 473, 2574
\l__enumext_vspace_a_star_v_bool . . . . . 1285
\l__enumext_vspace_a_star_vii_bool . . . 1307
\l__enumext_vspace_a_star_viii_bool . . . 1318
\l__enumext_vspace_a_star_X_bool . . . . . 74
\__enumext_vspace_above: . . 49, 1253, 1253, 2713
\__enumext_vspace_above_v: . 50, 1281, 1281, 2858

\l__enumext_vspace_above_v_skip . . 1283, 1287,
    1289
\__enumext_vspace_above_vii: . . 50, 1303, 1303,
    3247
\l__enumext_vspace_above_vii_skip 1305, 1309,
    1311
\__enumext_vspace_above_viii: . 50, 1303, 1314,
    3615
\l__enumext_vspace_above_viii_skip 1316, 1320,
    1322
\l__enumext_vspace_b_star_v_bool . . . . . 1296
\l__enumext_vspace_b_star_vii_bool . . . 1329
\l__enumext_vspace_b_star_viii_bool . . . 1340
\l__enumext_vspace_b_star_X_bool . . . . . 74
\__enumext_vspace_below: . . 50, 1267, 1267, 2798
\__enumext_vspace_below_v: . 50, 1292, 1292, 2926
\l__enumext_vspace_below_v_skip . . 1294, 1298,
    1300
\__enumext_vspace_below_vii: . . 51, 1325, 1325,
    3256
\l__enumext_vspace_below_vii_skip 1327, 1331,
    1333
\__enumext_vspace_below_viii: . 51, 1325, 1336,
    3623
\l__enumext_vspace_below_viii_skip 1338, 1342,
    1344
\__enumext_widest_from:nnnn . . 35, 556, 556, 571,
    583
\g__enumext_widest_label_tl 23, 31, 44, 345, 349,
    353
\l__enumext_wrap_label_opt_v_bool . . . . 2414
\l__enumext_wrap_label_opt_vii_bool 92, 3308
\l__enumext_wrap_label_opt_viii_bool 98, 3652
\l__enumext_wrap_label_opt_X_bool . . . . . 74
\l__enumext_wrap_label_v_bool 2410, 2414, 2422,
    2477
\l__enumext_wrap_label_vii_bool 92, 3307, 3312,
    3320, 3388
\l__enumext_wrap_label_viii_bool . . 98, 3651,
    3656, 3664, 3742
\l__enumext_wrap_label_X_bool . . . . . 74
\__enumext_wrapper_label_v:n . . . . . 2479, 2966
\__enumext_wrapper_label_vii:n . . . . . 3391
\__enumext_wrapper_label_viii:n . . . . . 3745
\__enumext_zero_count_level: . . . . . 208, 208
\__enumext_zero_parsep: . . . . . 44, 940, 995, 995
enumext* . . . . . 5, 3160
enumXi . . . . . 317
enumXii . . . . . 317
enumXiii . . . . . 317
enumXiv . . . . . 317
enumXv . . . . . 317
enumXvi . . . . . 317
enumXvii . . . . . 317
enumXviii . . . . . 317
Environments provide by enumext:
enumext* 22, 23, 25–27, 30, 32–34, 37, 38, 40, 47, 50–54,
    57–64, 67, 70, 76–78, 89–92, 94, 96, 98, 100, 102, 105,
    107
enumext 22, 23, 25, 27, 30, 31, 33–46, 48–54, 57–64, 67, 70,
    72–78, 81, 85, 86, 88, 91, 102, 105, 106
keyans* 22–24, 26, 27, 30, 32–34, 37, 38, 40, 47, 50, 51, 57,
    59, 60, 66, 70, 76, 97, 105, 107

```

keyanspic	22–25, 30, 31, 34, 48, 51, 57, 60, 66–68, 83–86, 106
keyans	22–25, 27, 30, 31, 34–40, 42, 45, 46, 48–51, 57, 59, 60, 66–68, 72–75, 81–85, 88, 98, 105, 106
Environments:	
list	27, 28, 74–76
lrbox	86, 93, 100
minipage	27, 28, 40, 43, 83, 85, 86, 93, 100
multicols	41–43, 48, 79, 80, 82, 83
exp commands:	
\exp_after:wN	3817
\exp_args:Ne	2637, 3805
\exp_not:N	156, 348, 437, 458, 468, 669, 683, 684, 695, 696, 707, 708, 1994, 2100, 2101, 2213, 2276, 2277, 2289, 2290, 3704, 3705, 3814
\exp_not:n	437, 438, 458, 459, 468, 469, 670, 1444, 1455, 1785, 1792, 1959, 1970, 1980, 1994, 1995, 2072, 2180, 2215, 2217, 2657, 2670, 3227, 3240
F	
\fbox	1728
file commands:	
\file_input_stop:	4123
first	721
font	357
\footnote	70
\footnote	70, 2329
\footnotemark	2339
\footnotesize	2101, 2277, 2290, 3705
\footnotetext	2323
G	
\getkeyans	13, 101, 3803
group commands:	
\group_begin:	1921, 2099, 2275, 2288, 3367, 3386, 3703, 3730, 3740, 3825, 3859
\group_end:	1930, 2106, 2282, 2295, 3396, 3408, 3710, 3750, 3769, 3827, 3866
H	
\hbadness	3415, 3776
hbox commands:	
\hbox_set:Nn	337
\hfill	387, 391, 396, 397, 1207, 1225, 1994, 2213, 3136, 3547
hook commands:	
\hook_gput_code:nnn	9, 186, 190, 255
\hook_gset_rule:nnnn	256
\hspace	3426, 3787
\hyperlink	64, 68
\hyperlink	1994, 2213
\hypertarget	29
\hypertarget	285
I	
\IfHyperBoolean	263
\IfPackageLoadedTF	11, 259, 273
\ignorespaces	672
\inputlineno	227, 238
int commands:	
\int_add:Nn	3096, 3507
\int_case:nn	846, 997, 1669, 1692
\int_compare:nNnTF	451, 475, 922, 1041, 1186, 1190, 1194, 1712, 1913, 1917, 2114, 2153, 2158, 2163, 2188, 2271, 2612, 2631, 2684, 2697, 2736, 2752, 2766, 2787, 2804, 2843, 2847, 2876, 2901, 2914, 2934, 2938, 2993, 3066, 3076, 3092, 3187, 3265, 3275, 3421, 3430, 3444, 3477, 3487, 3503, 3595, 3601, 3782, 3791, 3926
\int_compare_p:nNn	224, 235, 1675, 2015, 2025, 2037, 2038, 2053, 2055, 2618, 2619, 3195, 3196
\int_decr:N	3095, 3506
\int_eval:n	1807, 2067, 2101, 2171, 2277, 2290, 2532, 2573, 3084, 3495, 3705
\int_from_alph:n	550, 564
\int_from_roman:n	552, 566
\int_gadd:Nn	3097, 3508
\int_gdecr:N	1678, 1683, 1686, 1689, 1697
\int_gincr:N	1532, 1537, 1926, 2224, 2360, 2390, 2726, 2867, 3285, 3363, 3629
\int_gset:Nn	227, 238, 2337
\int_gset_eq:NN	1604, 1608, 1614, 1620, 1628, 1632, 1638, 1644, 2334
\int_gzero:N	212, 1215, 1232, 1720, 1721, 2792, 2919, 3439, 3800
\int_if_exist:NTF	1411, 1470, 1602, 1618, 1626, 1642
\int_incr:N	2611, 2839, 2992, 3186, 3284, 3594, 3628
\int_mod:nn	3432, 3793
\int_new:N	20, 21, 22, 23, 24, 37, 38, 62, 78, 90, 99, 107, 120, 121, 128, 129, 130, 131, 132, 133, 134, 145, 162, 163, 164, 165, 166, 1413, 1472
\int_set:Nn	546, 550, 552, 1545, 1552, 1564, 1573, 1782, 1954, 3031, 3032, 3054, 3065, 3071, 3087, 3415, 3465, 3476, 3482, 3498, 3776, 3922
\int_set_eq:NN	1533, 1538, 2652, 3094, 3222, 3505
\int_step_function:nnN	2031, 2045, 2060
\int_step_inline:nnn	3033, 3949
\int_to_roman:n	194, 2011, 2049
\int_use:N	923, 1547, 1554, 1566, 1575, 2532, 2550, 2573, 2638, 2737, 2746, 2761, 2767, 3069, 3070, 3082, 3480, 3481, 3493
\int_zero:N	3424, 3785
\c_one_int	3054, 3073, 3079, 3085, 3089, 3092, 3465, 3484, 3490, 3496, 3500, 3503
\c_zero_int	224, 235, 2015, 2025, 2037, 2038, 2053, 2055, 3265, 3275, 3435, 3796
\item	28, 39, 40, 61, 70, 83, 84, 86, 89, 96
\item	70, 72, 91, 92, 98, 100, 248, 1825, 1831, 1856, 1864, 1951, 2190, 2193, 2397, 2431, 3172, 3174, 3581, 3583, 3693
\item*	6, 12, 2429
item-pos*	2305
item-sym*	2305
\itemindent	23, 75
\itemindent	74
itemindent	636
\itemsep	85, 86
\itemsep	3007, 3013
\itemwidth	3061, 3105, 3109, 3472, 3516, 3520
K	
keyans	11, 2815
keyans*	11, 3570
keyanspic	12, 2968
Keys for environments provide by enumext:	
above*	24, 49, 50
above	24, 49, 50, 78, 82, 90, 97
after	38–40, 80, 83, 90, 97
align	24, 31, 32, 73, 93
before*	38, 39, 78, 90, 97
before	38–40, 82
below*	24, 49–51
below	24, 49–51, 80, 83, 90, 97

check-ans	24, 25, 27, 57, 62, 68, 70–72, 78, 80, 81, 94, 105
columns-sep*	25, 59, 77, 90
columns-sep	40, 60, 77, 79, 82, 90
columns*	25, 59, 77, 90
columns	23, 40, 43, 49, 60, 77, 79, 82, 90
first	38–40, 93
font	31, 73, 93
item-pos*	62, 63, 70
item-sym*	23, 62, 63, 70, 71
item*-sep	71
itemindent	24, 37, 72, 93
itemsep	36, 76
labelsep	31, 71, 74, 93
labelwidth	30, 31, 34, 35, 74
label	23, 30, 31, 34, 35, 86
lisparindent	76
list-indent	23, 37, 85
list-offset	37
listparindent	37, 93
mark-ans	25, 58, 65
mark-pos	58, 59
mark-ref	25, 58, 64
mini-env	24, 40, 43, 48, 49, 70, 79, 82, 88, 90, 96, 97
mini-sep	24, 40, 79, 82
miniright*	24, 40
miniright	24, 40, 47, 88
minirigth*	27
minirigth	27
no-store	25, 57, 58
noitemsep	36, 44
nosep	36, 44
parindent	76
parsep	36, 76, 93
partopsep	36
ref	26, 32, 33
resume*	52–54
resume	23, 51–54, 75, 80, 90
rightmargin	37
save-ans	24, 51, 53, 54, 60, 62, 66, 67, 72, 80, 81, 84, 90, 98, 99, 101, 102, 105
save-key	25, 53
save-ref	25, 29, 58, 63, 64, 66, 68, 72, 99
save-sep	58
series	52–54
show-ans	25, 58, 59, 61, 63, 65, 72, 99
show-length	27, 38, 75, 105
show-pos	25, 58, 59, 61, 63, 65, 68, 72, 99
start	24, 27, 35, 75
store-brk	62, 63
topsep	36
widest	23, 27, 35
wrap-ans	58, 61, 65
wrap-label*	31, 71, 73, 92, 93, 98
wrap-label	31, 73, 92, 93, 98
wrap-opt	58
keys commands:	
\keys_define:nn	359, 381, 410, 491, 511, 527, 574, 594, 638, 657, 714, 723, 802, 819, 1238, 1347, 1352, 1418, 1651, 1725, 1761, 1779, 1897, 2307, 3829, 3892
\l_keys_key_str	3983
\keys_set:nn	373, 826, 1243, 1248, 1503, 1508, 1586, 1594, 1943, 2633, 2637, 2854, 3208, 3610, 3894, 3895, 3896, 3897, 3898, 3899, 3900, 3901, 3902, 3903, 3904, 3905, 3943
keyval commands:	
\keyval_parse:NNn	1432
L	
label	489, 509, 527
Labels provide by enumext:	
\Alph*	30, 31
\Roman*	30, 31
\alph*	30, 31
\arabic*	30–32
\roman*	30, 31
\labelsep	86
\labelsep	3008, 3011
labelsep	357
\labelwidth	31, 86
\labelwidth	3008, 3009
labelwidth	357
\leftmargin	23, 75
\leftmargin	74, 3008
legacy commands:	
\legacy_if:nTF	3351, 3354, 3718, 3721
\legacy_if_gset_false:n	916
\legacy_if_set_false:n	3353, 3720
\legacy_if_set_true:n	3313, 3338, 3345, 3358, 3657, 3685, 3725
\linewidth	79, 82
\linewidth	2721, 2864, 3030, 3057, 3118, 3468, 3529
\list	28
\list	246
list-indent	636
list-offset	636
\listparindent	3010
listparindent	636
\lrbox	3368, 3731
M	
\makebox	86
\makebox	1887, 1889, 2451, 3382, 3390, 3394, 3744, 3748
\makeLabel	70, 73, 86
\makeLabel	73, 2457, 2473
\makesavenoteenv	279
mark-ans	1723
mark-pos	1723, 1759
mark-ref	1723
mini-env	800
mini-sep	800
\minipage	28
\minipage	252
\miniright	10, 48, 1184, 2790, 2917
\miniright*	10
mode commands:	
\mode_if_vertical:TF	871, 899, 1022, 1101
\mode_leave_vertical:	669, 683, 695, 707, 1856, 1864, 1885, 2449, 3380
msg commands:	
\msg_error:nn	2845, 2849, 2936, 2995, 3189, 3597, 3603, 3906
\msg_error:nnn	1188, 1192, 1217, 1234, 1362, 1373, 1512, 3819, 3824, 3889, 3959
\msg_error:nnnn	1911, 1915, 1919, 2837, 2932, 2940
\msg_fatal:nn	2613
\msg_fatal:nnn	311
\msg_info:nnn	13, 16, 261, 275
\msg_line_context:	3987, 3992, 3997, 4012, 4027, 4031, 4043, 4047, 4051, 4056, 4061, 4066, 4071, 4075,

4080, 4085, 4089, 4094, 4098, 4103, 4108, 4113, 4117, 4121	
\msg_new:nnn	3960, 3964, 3968, 3972, 3977, 3981, 3985, 3990, 3995, 4010, 4025, 4029, 4033, 4040, 4045, 4049, 4054, 4059, 4064, 4069, 4073, 4078, 4083, 4087, 4092, 4096, 4101, 4106, 4111, 4115, 4119
\msg_term:nnn	1715
\msg_term:nnnn	2540, 2550, 2579, 2584
\msg_warning:nn	2789, 2916
\msg_warning:nnn	1386, 1396, 1718
\msg_warning:nnnn	2231, 2489, 2494, 3068, 3081, 3479, 3492
\multicolsep	79, 82
\multicolsep	2751, 2889
N	
\NeedsTeXFormat	3
\newcounter	314
\NewDocumentCommand	1184, 1907, 2928, 3803, 3857, 3913
\NewDocumentEnvironment	2589, 2815, 2968, 3160, 3570
\newlabel	29
\newlabel	297
no-store	1649
\noindent	89, 96
\noindent	2728, 2869, 3127, 3173, 3423, 3538, 3582, 3784
\nointerlineskip	2728, 2869, 3127, 3538
noitemsep	592
\nopagebreak	882, 910, 1033, 1112, 1175, 1181
\normalfont	2100, 2276, 2289, 3704
nosep	592
P	
Packages:	
enumext	22, 51, 74, 84, 104
enumitem	30
expl3	86
footnotehyper	29
hyperref	25, 27, 29, 33, 64, 68, 92, 93, 104
lua-visual-debug	43
multicol	22, 104
shortlst	86
\par	882, 910, 1033, 1112, 1175, 1181, 1210, 1227, 2079, 2772, 2794, 2906, 2921, 3042, 3145, 3152, 3423, 3437, 3556, 3563, 3784, 3798
\parindent	3400, 3754
\parsep	41, 44, 85, 86
\parsep	1857, 1865, 2570, 3007, 3014, 3019
parsep	592
\parskip	3401, 3755
\partopsep	86
\partopsep	2571, 3012
partopsep	592
peek commands:	
\peek_meaning:N	3290, 3304, 3321, 3332, 3634, 3648, 3665
\peek_meaning_remove:N	3297, 3641
\peek_remove_spaces:n	2435
\phantomsection	29
\phantomsection	286
prg commands:	
\prg_do_nothing:	290
\prg_new_protected_conditional:Npnn	196
\prg_replicate:nn	205, 4038
\prg_return_false:	200
\prg_return_true:	199
\printkeyans	13, 102, 3857

prop commands:	
\prop_count:N	1807, 2067, 2103, 2171, 2279, 2292, 3707
\prop_gput_if_not_in:Nnn	1802, 1805
\prop_if_exist:N	1401, 3823
\prop_item:Nn	3826
\prop_new:N	1403
\ProvidesExplPackage	4
R	
\raggedcolumns	2760, 2895
\ref	64, 66
ref	489, 509
\refstepcounter	3360, 3727
regex commands:	
\regex_match:nnTF	198, 549, 551, 563, 565, 2650, 2663, 3220, 3233
\regex_replace_once:nnN	425
\renewcommand	437, 458, 468
\RenewDocumentCommand	2329, 2397, 2431, 2457, 2473
\RequirePackage	17
resume	1416
resume*	1416
rightmargin	636
\Roman	31, 35
\Roman	333
\roman	31, 35
\roman	334, 507, 3845
S	
save-ans	1347
save-ref	1723
save-sep	1723
scan commands:	
\scan_stop:	86, 3021, 3172, 3581, 3814, 3817
seq commands:	
\seq_clear:N	3920
\seq_const_from_clist:Nn	3908
\seq_count:N	2981, 3924
\seq_gclear:N	2327, 2328
\seq_gput_right:Nn	1814, 2340, 2341
\seq_if_empty:N	2346, 3872, 3938
\seq_if_exist:N	1405, 3870
\seq_item:Nn	3039
\seq_map_function:NN	3929
\seq_map_inline:Nn	3878, 3883, 3917, 3939, 3940
\seq_map_pairwise_function:NNN	2348
\seq_new:N	100, 101, 118, 146, 147, 1407
\seq_pop_left:NN	3928
\seq_put_right:Nn	2942, 3936, 3953
\seq_set_from_clist:Nn	3921
\seq_set_map_e:NNn	3930
\seq_show:N	3874
series	1416
\setcounter	560, 564, 566, 2532, 2573, 2986
\setenumext	6-9, 103, 3833, 3838, 3843, 3848, 3853, 3913
\setlength	1858, 1866
show-ans	1723, 1759
show-length	712
skip commands:	
\skip_add:Nn	851, 857, 863, 873, 877, 901, 905, 1002, 1008, 1014, 1024, 1028, 1050, 1103, 1107, 3007
\skip_eval:n	1857, 1865
\skip_gset:Nn	1123, 1127, 1131
\skip_gzero_new:N	1118, 1119
\skip_horizontal:N	684, 696, 708, 3383, 3397, 3751

<code>\skip_horizontal:n</code>	670, 1886, 1894, 2450, 2452, 3381, 3759
<code>\skip_if_eq:nnTF</code>	849, 855, 861, 925, 959, 1000, 1006, 1012, 1043, 1048, 1069, 1120, 1142, 1255, 1269, 1283, 1294, 1305, 1316, 1327, 1338
<code>\skip_new:N</code>	58, 59, 63, 64, 65, 66, 67, 122, 176
<code>\skip_set:Nn</code>	834, 838, 887, 891, 928, 932, 936, 943, 947, 951, 962, 967, 971, 977, 982, 987, 1045, 1046, 1047, 1054, 1058, 1062, 1071, 1076, 1080, 1083, 1087, 1091, 1122, 1126, 1144, 1148, 1152, 1158, 1162, 1166, 3001, 3015
<code>\skip_set_eq:NN</code>	2530, 2569, 2570, 3400, 3401, 3754, 3755
<code>\skip_use:N</code>	836, 840, 875, 879, 883, 903, 907, 926, 945, 954, 960, 965, 969, 980, 984, 985, 990, 1026, 1030, 1056, 1256, 1260, 1263, 1270, 1274, 1277, 2772
<code>\skip_zero:N</code>	2571, 2751, 2889, 3012, 3013
<code>\skip_zero_new:N</code>	1038, 1039, 1040, 1117, 1139, 1140, 1141
<code>\c_zero_skip</code>	849, 855, 861, 926, 960, 1000, 1006, 1012, 1043, 1048, 1069, 1120, 1142, 1256, 1270, 1283, 1294, 1305, 1316, 1327, 1338
<code>\small</code>	3835, 3840, 3845, 3850, 3855
<code>\star</code>	2311
<code>start</code>	572
<code>\stepcounter</code>	2333, 2949
str commands:	
<code>\c_backslash_str</code>	4047, 4056, 4057, 4061, 4062, 4066, 4067, 4098, 4099, 4103, 4108, 4109
<code>\c_colon_str</code>	2066, 2170, 3814
<code>\str_case:nn</code>	218
<code>\str_case:nnTF</code>	1439, 1448
<code>\str_clear:N</code>	2630, 3207
<code>\str_count:n</code>	205, 4038
<code>\str_if_empty:NTF</code>	1459, 1612, 1636
<code>\str_if_eq:nnTF</code>	2533, 2575
<code>\str_if_in:nnTF</code>	3810
<code>\str_new:N</code>	117, 171
<code>\str_set:Nn</code>	413, 414, 415, 1740, 1741, 1764, 1765
<code>\string</code>	279
<code>\strutbox</code>	930, 934, 938, 949, 953, 964, 973, 979, 989, 1002, 1008, 1014, 1045, 1046, 1047, 1050, 1060, 1064, 1073, 1080, 1085, 1093, 1122, 1123, 1126, 1133, 1146, 1154, 1160, 1168, 3017

T

TeX and L^AT_EX commands:

<code>\@auxout</code>	295
<code>\@currentenv</code>	218
<code>\protected@write</code>	295

text commands:

<code>\text_expand:n</code>	3806
<code>\textasteriskcentered</code>	1737, 1751
<code>\thepage</code>	301

tl commands:

<code>\c_space_tl</code>	2258, 3997, 4012
<code>\tl_clear:N</code>	386, 392, 1940, 2113, 2187, 3671
<code>\tl_clear_new:N</code>	343
<code>\tl_const:Nn</code>	148, 327
<code>\tl_gclear:N</code>	1481, 1487, 2233, 2468, 2812, 3156, 3384, 3452, 3567
<code>\tl_gclear_new:N</code>	1467
<code>\tl_gput_right:Nn</code>	328
<code>\tl_greplace_all:Nnn</code>	349
<code>\tl_gset:Nn</code>	1468, 1482, 1488, 2221, 2780, 3327
<code>\tl_gset_eq:NN</code>	345, 2378, 3377

<code>\tl_if_blank:nTF</code>	3375
<code>\tl_if_empty:NTF</code>	443, 477, 483, 1360, 1371, 1519, 1583, 1591, 1606, 1610, 1630, 1634, 1821, 1852, 1966, 2124, 2197, 2229, 2252, 2447, 3674, 3951
<code>\tl_if_empty:nTF</code>	1493
<code>\tl_if_exist:NTF</code>	1498
<code>\tl_if_novalue:nTF</code>	1941, 1952, 2121, 2195, 2237, 2331, 2356, 2374, 2379, 2408, 2628, 2979, 3205, 3608, 3672, 3915
<code>\tl_map_inline:Nn</code>	346, 423
<code>\tl_new:N</code>	34, 39, 41, 42, 43, 45, 46, 79, 80, 81, 87, 88, 89, 91, 92, 93, 94, 95, 97, 98, 104, 105, 115, 116, 127, 137, 138, 139, 142, 150, 151, 154, 155, 170, 173
<code>\tl_put_left:Nn</code>	1829, 1862, 1949, 2264, 2298, 3690, 3693
<code>\tl_put_right:Nn</code>	344, 435, 456, 466, 1783, 1790, 1833, 1868, 1951, 1957, 1965, 1968, 1978, 1983, 1986, 1992, 2018, 2028, 2042, 2058, 2064, 2069, 2116, 2119, 2126, 2128, 2155, 2160, 2165, 2168, 2177, 2190, 2193, 2199, 2201, 2211, 2655, 2668, 3225, 3238, 3676, 3677, 3831, 3836, 3841, 3846, 3851
<code>\tl_remove_all:Nn</code>	3950
<code>\tl_remove_once:Nn</code>	2006, 2140
<code>\tl_replace_all:Nnn</code>	348
<code>\tl_reverse:N</code>	2005, 2007, 2139, 2141
<code>\tl_set:Nn</code>	156, 313, 387, 391, 396, 397, 431, 450, 667, 681, 693, 705, 1359, 1370, 1518, 2097, 2239, 2273, 2286, 2376, 3679, 3701, 3948
<code>\tl_set_eq:NN</code>	354, 432, 434, 453, 455, 463, 465, 2004, 2138, 2151, 2420, 2424, 2960, 2962
<code>\tl_to_str:n</code>	1498, 1504, 1509, 3806
<code>\tl_trim_spaces:n</code>	344, 3936, 3948, 3954
<code>\tl_use:N</code>	350, 353, 445, 479, 485, 738, 742, 746, 750, 754, 758, 762, 766, 770, 774, 778, 782, 786, 790, 794, 798, 1891, 2011, 2019, 2030, 2044, 2049, 2061, 2363, 2369, 2393, 2411, 2415, 2423, 2459, 2460, 2467, 2475, 2476, 2482, 2596, 2821, 2965, 3150, 3387, 3398, 3402, 3561, 3741, 3752, 3758, 3762, 3860, 3861, 3862, 3863, 3864, 3932

token commands:

<code>\token_to_str:N</code>	297
<code>\topsep</code>	1858, 1866
<code>topsep</code>	592
<code>\typeout</code>	228, 239, 265, 268, 278, 279, 1480, 1486, 1679, 1698, 2622, 3199

U

<code>\u</code>	426
use commands:	
<code>\use:N</code>	206, 2464, 2598
<code>\use:n</code>	1430, 3812
<code>\use_none:nn</code>	289
<code>\usecounter</code>	2531, 2572

V

<code>\value</code>	1604, 1608, 1614, 1620, 1628, 1632, 1638, 1644
<code>\vspace</code>	917, 1260, 1263, 1274, 1277, 1287, 1289, 1298, 1300, 1309, 1311, 1320, 1322, 1331, 1333, 1342, 1344, 1857, 1865, 2976, 2987, 3438, 3799

W

<code>widest</code>	572
<code>wrap-ans</code>	1723
<code>wrap-label</code>	357
<code>wrap-label*</code>	357
<code>wrap-opt</code>	1723