

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 multicol	4	4.5	The environment <code>keyans</code>	10
1.3.3	Support for minipage	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	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 label and ref	6	6	The way of non-enumerated lists	16
3.2	Keys for spaces	6	7	References	18
3.2.1	Vertical spaces	7	8	Change history	18
3.2.2	Horizontal spaces	7	9	Index of Documentation	19
3.3	Keys for add code	8	10	Implementation	21
3.4	Keys for start and resume	8	11	Index of Implementation	107
3.5	Keys for multicols	8			
3.6	Keys for minipage	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/l^{pp}l.txt>). The software has the status “maintained”.

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

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

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

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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

(A) value

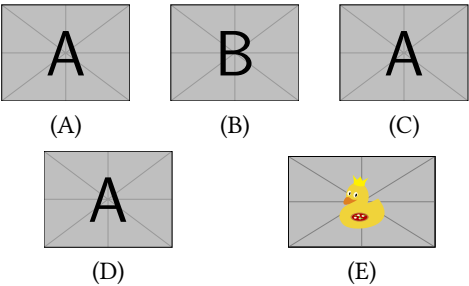
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
-

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

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

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

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

1.1 Description and usage

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

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

```
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 $[\langle key = val \rangle]$ options.

1.3.1 Internal counters

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

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

1.3.2 Support for multicol

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

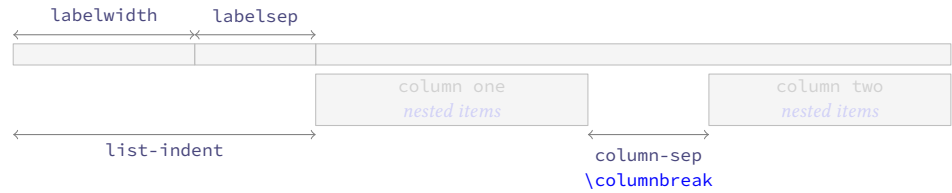


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

1.3.3 Support for minipage

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



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

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

1.3.4 The \label and \ref system

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

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

1.3.5 Support for \footnote

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

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

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

2 The environment enumext

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

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

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

Example

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

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

2.1 The `\item*` in `enumext`

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

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

3.4 Keys for start and resume

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

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

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

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

3.5 Keys for multicol

`columns = {⟨integer⟩}` default: `1`

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

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

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

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

3.6 Keys for minipage

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

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

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

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

3.6.1 The command `\miniright`

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

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

3.6.2 The key `miniright`

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

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

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

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

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

4 The storage system

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

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

4.1 Keys for storage

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

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

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

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

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

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

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

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

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

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

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

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

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.

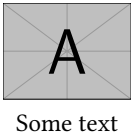
Example

```
\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



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.

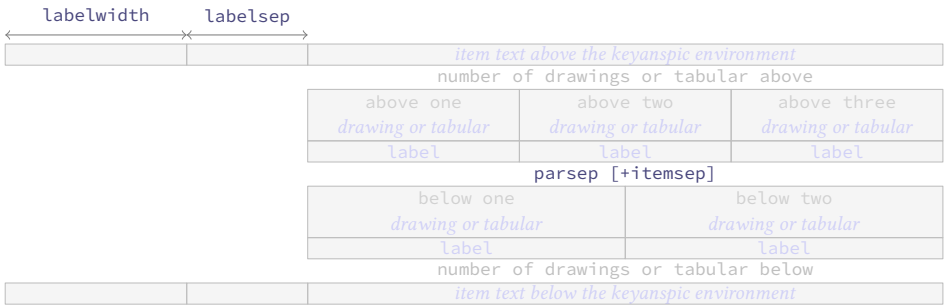


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

4.6.1 The command `\anspic`

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

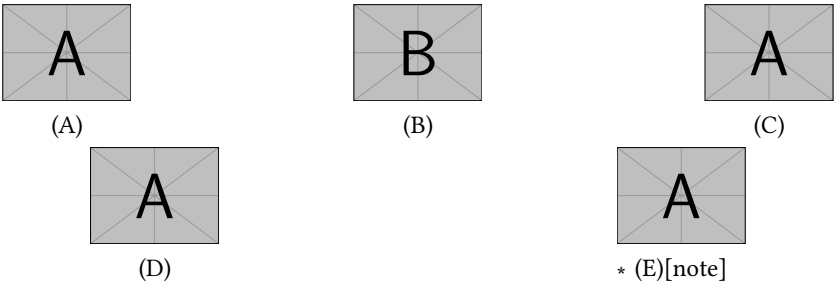
The `\anspic` command take three arguments, the *starred version* ‘`*`’ store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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

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



Test keys

1. B, $x = 5$
2. D
3. C, some note
4. E, A duck
5. D, other note

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

 True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?
- 4

 Related to Linux
- (a)

 You use linux?
- (b)

 Usually uses the package manager?
- (c)

 Rate the following package and class
- i.

 xsim-exam
- ii.

 xsim
- iii.

 exsheets

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

1. $(x - 1)^2$
2. $3(x + y + z)$
3. (a) False
- (b) Very True!
4. (a) Yes
- (b) Yes, dnf
- (c) i. doesn't exist for now :(
- ii. very good
- iii. obsolete

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

Here we set the label key using the default settings in L^AT_EX for the four levels \textbullet, \textendash, \textasteriskcentered and \textperiodcentered together with the 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:

article2

book2

exam3

letter2

report2

\columnbreak5

\columnsep9

Commands provide by enumext:

\anskey4, 10, 11

\anspic*4, 10–13

\anspic10, 12, 13

\getkeyans4, 11, 13

\item*4–7, 10–12

\item6, 7, 9–11

\miniright4, 5, 10

\printkeyans4, 6, 11, 13

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

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

enumext*4, 5, 10

enumext4–6, 9–14, 17

keyans*4, 5, 10

keyanspic4, 7, 10–13, 17

keyans4–12, 17

Environments:

enumerate1, 3, 4, 6, 19

list4, 9, 19

minipage3–5, 9, 10, 19

multicols3, 5, 9

I

\item4, 5

\itemsep8

K

Keys for environments provide by enumext:

above*8

above8

after9, 10

align7, 18

before*9

before9

below*8

below8

check-ans11

columns-sep5, 9

columns5, 8, 9

first9

font7

L

\label5

Labels provide by enumext:

\Alph*7, 12

\Roman*7

\alph*7

\arabic*7

\roman*7

\labelsep4, 7

\labelwidth4, 7

\linewidth9

\listparindent8

P

Packages:

enumerate18

enumext1–4, 13, 18, 19

enumitem4, 5, 9, 18, 19

footnotehyper5

hyperref5, 11, 19

l3prop1, 19

l3seq1, 19

item-pos*6

item-sym*6

itemindent8

itemsep8, 12

labelsep4, 6–10, 18

labelwidth4, 6, 7, 9, 10, 18

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

list-indent4, 8, 9

list-offset4, 8, 18

listparindent8

mark-ans10

mark-pos10

mark-ref11

mini-env5, 8, 9

mini-sep5, 9

miniright*10

miniright10

no-store11

noitemsep8

nosep8, 17

parsep8, 12

partopsep8

ref5, 7

resume9

rightmargin8

save-ans5, 9–13

save-ref5, 7, 11, 13

save-sep10

show-ans10, 11

show-length7

show-pos10, 11, 13

start9

topsep8

widest7

wrap-ans10

wrap-label*7, 18

wrap-label7

wrap-opt10

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_vi_tl
\l__enumext_counter_vii_tl
\l__enumext_counter_viii_tl

```

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

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

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

```

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
\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 \tl_new:N \g__enumext_item_symbol_tl
41 \tl_new:N \g__enumext_standar_series_tl
42 \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.

```

43 \dim_new:N \l__enumext_current_widest_dim
44 \tl_new:N \g__enumext_counter_styles_tl
45 \tl_new:N \g__enumext_widest_label_tl
46 \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`.

```

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

```

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

```

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

```

73 \cs_set_protected:Npn \__enumext_tmp:n #1
74 {
75   \bool_new:c { \l__enumext_wrap_label_#1_bool }
76   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
77   \int_new:c { \l__enumext_start_#1_int }
78   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
79   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
80   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
81   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
82   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
83 }
84 \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.

```

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

```

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

```

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

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

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

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

```
112 \bool_new:N \l__enumext_show_answer_bool
113 \bool_new:N \l__enumext_show_position_bool
114 \tl_new:N \l__enumext_mark_ref_sym_tl
115 \tl_new:N \l__enumext_mark_answer_sym_tl
116 \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).

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

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

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

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

```

138 \tl_new:N \l__enumext_store_write_aux_file_tl
139 \cs_set_protected:Npn \__enumext_tmp:n #1
140 {
141     \tl_new:c { l__enumext_label_copy_#1_tl }
142 }
143 \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`

```

144 \int_new:N \g__enumext_footnote_int

```

`\g__enumext_footnote_int_seq`

```

145 \seq_new:N \g__enumext_footnote_arg_seq

```

```

146 \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`

```

147 \tl_const:Nn \c__enumext_counter_style_tl

```

`\l__enumext_ref_aux_tl`

```

148 { { arabic } { roman } { Roman } { alph } { Alph } }

```

`\l__enumext_the_counter_X_tl`

```

149 \tl_new:N \l__enumext_ref_key_arg_tl

```

`\l__enumext_counter_style_for_ref_X_tl`

```

150 \tl_new:N \l__enumext_ref_aux_tl

```

```

151 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

152 {

```

```

153     \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }

```

```

154     \tl_new:c { l__enumext_the_counter_#1_tl }

```

```

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

```

```

156 }

```

```

157 \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`

```

158 \cs_set_protected:Npn \__enumext_tmp:n #1

```

`\g__enumext_item_count_all_X_int`

```

159 {

```

`\l__enumext_joined_item_X_int`

```

160     \bool_new:c { l__enumext_item_starred_#1_bool }

```

`\l__enumext_joined_item_aux_X_int`

```

161     \int_new:c { l__enumext_item_column_pos_#1_int }

```

`\l__enumext_tmpa_X_int`

```

162     \int_new:c { g__enumext_item_count_all_#1_int }

```

`\l__enumext_item_text_X_box`

```

163     \int_new:c { l__enumext_joined_item_#1_int }

```

`\l__enumext_joined_width_X_dim`

```

164     \int_new:c { l__enumext_joined_item_aux_#1_int }

```

`\l__enumext_item_width_X_dim`

```

165     \int_new:c { l__enumext_tmpa_#1_int }

```

`\g__enumext_item_symbol_aux_X_tl`

```

166     \box_new:c { l__enumext_item_text_#1_box }

```

`\l__enumext_align_label_X_str`

```

167     \dim_new:c { l__enumext_joined_width_#1_dim }

```

`\g__enumext_minipage_active_X_bool`

```

168     \dim_new:c { l__enumext_item_width_#1_dim }

```

`\g__enumext_miniright_code_X_tl`

```

169     \tl_new:c { g__enumext_item_symbol_aux_#1_tl }

```

`\g__enumext_minipage_center_X_bool`

```

170     \str_new:c { l__enumext_align_label_#1_str }

```

`\g__enumext_minipage_right_X_dim`

```

171     \bool_new:c { g__enumext_minipage_active_#1_bool }

```

`\g__enumext_minipage_right_X_skip`

```

172     \tl_new:c { g__enumext_miniright_code_#1_tl }

```

```

173     \bool_new:c { g__enumext_minipage_center_#1_bool }

```

```

174     \dim_new:c { g__enumext_minipage_right_#1_dim }

```

```

175     \skip_new:c { g__enumext_minipage_right_#1_skip }

```

```

176 }

```

```

177 \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`.

```

178 \clist_const:Nn \c__enumext_all_envs_clist

```

```

179 {

```

```

180     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},

```

```

181     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}

```

```

182 }

```

(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.

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

(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.

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

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

`__enumext_level:`

Function for check current level in `enumext`.

```
191 \cs_new:Nn \__enumext_level:
192 {
193   \int_to_roman:n { \__enumext_level_int }
194 }
```

(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`

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

(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.

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

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

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

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

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

```

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

```

(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.

```

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

```

(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.

```

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

```

(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.

```
254 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
255 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

```
\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:
```

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

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

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`.

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

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

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

(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.

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

```

297         {
298             {#2}
299             \bool_if:NT \l__enumext_hyperref_bool
300             { { \thepage } {#2} {#1} }
301             { }
302         }
303     }
304     \__enumext_hypertarget:nn {#1} { }
305     \__enumext_phantomsection:
306 }

```

(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.

```

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

```

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

```

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

(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`.

```

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

```



```

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

```

(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`.

```

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

```

(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`.

```

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

```

(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*`.

```

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

```

```

479     }
480   }
481   {
482     \tl_if_empty:NF \__enumext_counter_style_for_ref_viii_tl
483     {
484       \tl_use:N \__enumext_counter_style_for_ref_viii_tl
485     }
486   }
487 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

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

```

\__enumext_label_i_tl 488 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_ii_tl 489 {
\__enumext_label_iii_tl 490   \keys_define:nn { enumext / #1 }
\__enumext_label_iv_tl 491   {
492     label .code:n = {
493       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
494       { \__enumext_counter_#2_tl } {##1}
495       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
496       \__enumext_current_widest_dim
497     },
498     label .initial:n = #3,
499     label .value_required:n = true,
500     ref .code:n = \__enumext_set_label_ref:n {##1},
501     ref .value_required:n = true,
502   }
503 }
504 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
505 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
506 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
507 \__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 `\labels` for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key.

```

\__enumext_label_vii_tl 508 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_viii_tl 509 {
510   \keys_define:nn { enumext / #1 }
511   {
512     label .code:n = {
513       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
514       { \__enumext_counter_#2_tl } {##1}
515       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
516       \__enumext_current_widest_dim
517     },
518     label .initial:n = #3,
519     label .value_required:n = true,
520     ref .code:n = \__enumext_set_label_ref_h:n {##1},
521     ref .value_required:n = true,
522   }
523 }
524 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
525 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

10.11.3 Define and set label key for keyans and keyanspic environment

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

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

```

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

```

```

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

```

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

10.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

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

```

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

```

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

```

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

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

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

```

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

```

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

```

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

`start`

`widest`

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

```

\__enumext_start_X_int

```

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

```

691     {
692         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
693         {
694             \exp_not:N \mode_leave_vertical:
695             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
696         }
697     }
698 }
699 \cs_set_protected:Nn \__enumext_fake_item_viii:
700 {
701     \dim_compare:nNt
702     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
703     {
704         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
705         {
706             \exp_not:N \mode_leave_vertical:
707             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
708         }
709     }
710 }

```

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

10.15 Setting show-length key

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

```

711 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
712 {
713     \keys_define:nn { enumext / #1 }
714     {
715         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
716         show-length .initial:n = false,
717     }
718 }
719 \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
720 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
721 {
722     \keys_define:nn { enumext / #1 }
723     {
724         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
725         before .value_required:n = true,
726         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
727         before* .value_required:n = true,
728         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
729         after .value_required:n = true,
730         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
731         first .value_required:n = true,
732     }
733 }
734 \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:
735 \cs_new_protected:Nn \__enumext_before_args_exec:
736 {
737     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
738 }

```

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

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

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

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

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

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

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

10.16.2 Functions for before, after and first keys in keyans

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

```
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
751 \cs_new_protected:Nn \__enumext_before_args_exec_v:
752 {
753   \tl_use:N \l__enumext_before_starred_key_v_tl
754 }
```

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

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

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

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

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

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

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

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

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

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


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`.

```

775 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
776 {
777     \tl_use:N \__enumext_before_no_starred_key_vii_tl
778 }
779 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
780 {
781     \tl_use:N \__enumext_before_no_starred_key_viii_tl
782 }

```

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

```

783 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
784 {
785     \tl_use:N \__enumext_after_stop_list_vii_tl
786 }
787 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
788 {
789     \tl_use:N \__enumext_after_stop_list_viii_tl
790 }

```

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`.

```

791 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
792 {
793     \tl_use:N \__enumext_after_list_args_vii_tl
794 }
795 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
796 {
797     \tl_use:N \__enumext_after_list_args_viii_tl
798 }

```

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

10.17 Setting keys for multicol and minipage

`mini-env` The default value of the `columns-sep` key is handled by the state of the boolean variable `__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`

```

799 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
800 {
801     \keys_define:nn { enumext / #1 }
802     {
803         mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
804         mini-env .value_required:n = true,
805         mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
806         mini-sep .initial:n = 0.3333em,
807         mini-sep .value_required:n = true,
808         columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
809         columns-sep .value_required:n = true,
810         columns .int_set:c = { \__enumext_columns_#2_int },
811         columns .initial:n = 1,
812         columns .value_required:n = true,
813     }
814 }
815 \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`.

```

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

```

```

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

```

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

10.18 Adjustment of vertical spaces for multicols

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

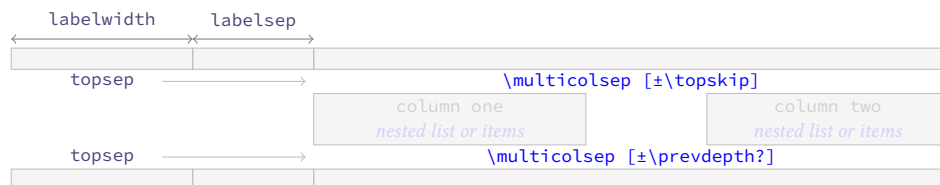


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

10.18.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

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

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

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

```

```

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

```

(End of definition for `\l__enumext_add_pre_parsep:`)

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

```

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

```

(End of definition for `\l__enumext_multi_addvspace:`)

10.18.2 Adjustment of vertical spaces for multicols in keyans

`\l__enumext_keyans_multi_set_vskip:` The function `\l__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`.

`\l__enumext_keyans_multi_addvspace:`

```

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

```

(End of definition for `\l__enumext_keyans_multi_set_vskip:` and `\l__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.

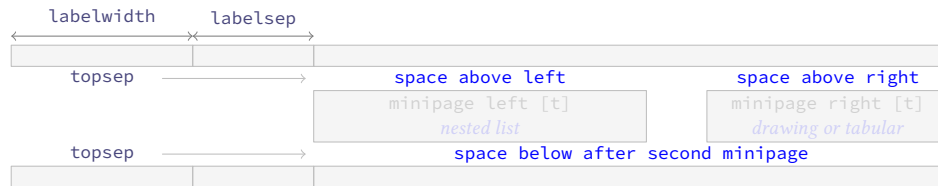


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```

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

```

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

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.

```

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

```

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

```

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.

```

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

```

(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.

```

994 \cs_new_protected:Nn \__enumext_zero_parsep:
995 {

```

```

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

```

(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 $\langle horizontal mode \rangle$ or $\langle vertical mode \rangle$. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1018 \cs_new_protected:Nn \__enumext_mini_addvspace:
1019 {
1020   \__enumext_mini_set_vskip:
1021   \mode_if_vertical:T
1022   {
1023     \skip_add:Nn \l__enumext_minipage_left_skip
1024     {
1025       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1026     }
1027     \skip_add:Nn \l__enumext_minipage_after_skip
1028     {
1029       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1030     }
1031   }
1032   \par\nopagebreak
1033   \addvspace { \l__enumext_minipage_left_skip }
1034 }

```

(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`.

```

1035 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1036 {
1037   \skip_zero_new:N \l__enumext_minipage_after_skip
1038   \skip_zero_new:N \l__enumext_minipage_left_skip
1039   \skip_zero_new:N \l__enumext_minipage_right_skip
1040   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1041   {
1042     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1043     {
1044       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1045       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1046       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1047       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1048       {
1049         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1050       }
1051     }
1052   }

```



```

1052     {
1053         \skip_set:Nn \l__enumext_minipage_left_skip
1054         {
1055             \skip_use:N \l__enumext_topsep_v_skip
1056         }
1057         \skip_set:Nn \l__enumext_minipage_right_skip
1058         {
1059             0.705\box_dp:N \strutbox
1060         }
1061         \skip_set:Nn \l__enumext_minipage_after_skip
1062         {
1063             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1064         }
1065     }
1066 }
1067 {
1068     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1069     {
1070         \skip_set:Nn \l__enumext_minipage_left_skip
1071         {
1072             0.5\box_dp:N \strutbox
1073             + \l__enumext_partopsep_v_skip
1074         }
1075         \skip_set:Nn \l__enumext_minipage_right_skip
1076         {
1077             \l__enumext_partopsep_v_skip
1078         }
1079         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1080     }
1081     {
1082         \skip_set:Nn \l__enumext_minipage_left_skip
1083         {
1084             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1085         }
1086         \skip_set:Nn \l__enumext_minipage_right_skip
1087         {
1088             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1089         }
1090         \skip_set:Nn \l__enumext_minipage_after_skip
1091         {
1092             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1093         }
1094     }
1095 }
1096 }

```

(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 $\langle horizontal\ mode\rangle$ or $\langle vertical\ mode\rangle$. 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`.

```

1097 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1098 {
1099     \__enumext_keyans_mini_set_vskip:
1100     \mode_if_vertical:T
1101     {
1102         \skip_add:Nn \l__enumext_minipage_left_skip
1103         {
1104             \l__enumext_partopsep_v_skip
1105         }
1106         \skip_add:Nn \l__enumext_minipage_after_skip
1107         {
1108             \l__enumext_partopsep_v_skip
1109         }
1110     }
1111     \par\nopagebreak
1112     \addvspace { \l__enumext_minipage_left_skip }
1113 }

```

(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*`.

```

1114 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1115 {
1116   \skip_zero_new:N \l__enumext_minipage_left_skip
1117   \skip_gzero_new:N \g__enumext_minipage_right_skip
1118   \skip_gzero_new:N \g__enumext_minipage_after_skip
1119   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1120   {
1121     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1122     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1123   }
1124   {
1125     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1126     \skip_gset:Nn \g__enumext_minipage_right_skip
1127     {
1128       \l__enumext_topsep_vii_skip
1129     }
1130     \skip_gset:Nn \g__enumext_minipage_after_skip
1131     {
1132       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1133     }
1134   }
1135 }
1136 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1137 {
1138   \skip_zero_new:N \l__enumext_minipage_after_skip
1139   \skip_zero_new:N \l__enumext_minipage_left_skip
1140   \skip_zero_new:N \l__enumext_minipage_right_skip
1141   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1142   {
1143     \skip_set:Nn \l__enumext_minipage_left_skip
1144     {
1145       0.5\box_dp:N \strutbox
1146     }
1147     \skip_set:Nn \l__enumext_minipage_right_skip
1148     {
1149       \l__enumext_partopsep_viii_skip
1150     }
1151     \skip_set:Nn \l__enumext_minipage_after_skip
1152     {
1153       1.6\box_dp:N \strutbox
1154     }
1155   }
1156   {
1157     \skip_set:Nn \l__enumext_minipage_left_skip
1158     {
1159       0.5875\box_dp:N \strutbox
1160     }
1161     \skip_set:Nn \l__enumext_minipage_right_skip
1162     {
1163       \l__enumext_topsep_viii_skip
1164     }
1165     \skip_set:Nn \l__enumext_minipage_after_skip
1166     {
1167       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1168     }
1169   }
1170 }

```

(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.

```

1171 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:

```

```

1172 {
1173   \__enumext_mini_set_vskip_vii:
1174   \par\nopagebreak
1175   \addvspace { \__enumext_minipage_left_skip }
1176 }
1177 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1178 {
1179   \__enumext_mini_set_vskip_viii:
1180   \par\nopagebreak
1181   \addvspace { \__enumext_minipage_left_skip }
1182 }

```

(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.

```

1183 \NewDocumentCommand \miniright { s }
1184 {
1185   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1186   {
1187     \msg_error:nnn { enumext } { wrong-miniright-place }
1188   }
1189   \int_compare:nNnT { \__enumext_level_int } = { 0 }
1190   {
1191     \msg_error:nnn { enumext } { wrong-miniright-place }
1192   }
1193   \int_compare:nNnTF { \__enumext_keyans_level_int } = { 1 }
1194   {
1195     \__enumext_keyans_mini_right_cmd:n {#1}
1196   }
1197   { \__enumext_mini_right_cmd:n {#1} }
1198 }

```

(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.

```

1199 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1200 {
1201   \dim_compare:nNnTF
1202   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1203   {
1204     \__enumext_multicols_stop:
1205     \end{\__enumext_mini_env*}
1206     \hfill
1207     \begin{\__enumext_mini_env*}
1208     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1209     \par\addvspace { \__enumext_minipage_right_skip }
1210     \bool_if:nF {#1}
1211     {
1212       \centering
1213     }
1214     \int_gzero:N \g__enumext_minipage_stat_int
1215   }
1216   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1217 }

```

(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.

```

1218 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1219 {
1220   \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1221   {
1222     \__enumext_keyans_multicols_stop:
1223     \end{__enumext_mini_env*}
1224     \hfill
1225     \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1226     \par\addvspace { \__enumext_minipage_right_skip }
1227     \bool_if:nF {#1}
1228     {
1229       \centering
1230     }
1231     \int_gzero:N \g__enumext_minipage_stat_int
1232   }
1233   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1234 }

```

(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*`

```

1235 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1236 {
1237   \keys_define:nn { enumext / #1 }
1238   {
1239     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1240     above .value_required:n = true,
1241     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1242                       \keys_set:nn { enumext / #1 } { above = {##1} },
1243     above* .value_required:n = true,
1244     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1245     below .value_required:n = true,
1246     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1247                       \keys_set:nn { enumext / #1 } { below = {##1} },
1248     below* .value_required:n = true,
1249   }
1250 }
1251 \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.

```

1252 \cs_new_protected:Nn \__enumext_vspace_above:
1253 {
1254   \skip_if_eq:nnF
1255   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1256   {
1257     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1258     {
1259       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1260     }
1261     {
1262       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1263     }
1264   }
1265 }

```

(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.

```

1266 \cs_new_protected:Nn \__enumext_vspace_below:
1267 {
1268   \skip_if_eq:nnF
1269     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1270   {
1271     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1272     {
1273       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1274     }
1275     {
1276       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1277     }
1278   }
1279 }

```

(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.

```

1280 \cs_new_protected:Nn \__enumext_vspace_above_v:
1281 {
1282   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1283   {
1284     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1285     {
1286       \vspace*{ \__enumext_vspace_above_v_skip }
1287     }
1288     { \vspace { \__enumext_vspace_above_v_skip } }
1289   }
1290 }

```

(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.

```

1291 \cs_new_protected:Nn \__enumext_vspace_below_v:
1292 {
1293   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1294   {
1295     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1296     {
1297       \vspace*{ \__enumext_vspace_below_v_skip }
1298     }
1299     { \vspace { \__enumext_vspace_below_v_skip } }
1300   }
1301 }

```

(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:`

```

1302 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1303 {
1304   \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1305   {
1306     \bool_if:NTF \__enumext_vspace_a_star_vii_bool
1307     {
1308       \vspace*{ \__enumext_vspace_above_vii_skip }
1309     }
1310     { \vspace { \__enumext_vspace_above_vii_skip } }
1311   }
1312 }
1313 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1314 {
1315   \skip_if_eq:nnF { \__enumext_vspace_above_viii_skip } { \c_zero_skip }

```

```

1316     {
1317         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1318         {
1319             \vspace*{ \l__enumext_vspace_above_viii_skip }
1320         }
1321         { \vspace { \l__enumext_vspace_above_viii_skip } }
1322     }
1323 }

```

(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.

```

1324 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1325 {
1326     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1327     {
1328         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1329         {
1330             \vspace*{ \l__enumext_vspace_below_vii_skip }
1331         }
1332         { \vspace { \l__enumext_vspace_below_vii_skip } }
1333     }
1334 }
1335 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1336 {
1337     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1338     {
1339         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1340         {
1341             \vspace*{ \l__enumext_vspace_below_viii_skip }
1342         }
1343         { \vspace { \l__enumext_vspace_below_viii_skip } }
1344     }
1345 }

```

(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***.

```

1346 \keys_define:nn { enumext / level-1 }
1347 {
1348     save-ans .code:n = \l__enumext_storing_set:n {#1},
1349     save-ans .value_required:n = true,
1350 }
1351 \keys_define:nn { enumext / enumext* }
1352 {
1353     save-ans .code:n = \l__enumext_storing_set_vii:n {#1},
1354     save-ans .value_required:n = true,
1355 }

```

(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).

```

1356 \cs_new_protected:Npn \l__enumext_storing_set:n #1
1357 {
1358     \tl_set:Ne \l__enumext_store_name_tl {#1}
1359     \tl_if_empty:NTF \l__enumext_store_name_tl
1360     {
1361         \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1362     }

```

```

1363     {
1364         \__enumext_storing_standar:
1365     }
1366 }
1367 \cs_new_protected:Npn \__enumext_storing_set_vii:n #1
1368 {
1369     \tl_set:Nx \l__enumext_store_name_tl {#1}
1370     \tl_if_empty:NTF \l__enumext_store_name_tl
1371     {
1372         \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1373     }
1374     {
1375         \__enumext_storing_starred:
1376     }
1377 }
1378 \cs_new_protected:Nn \__enumext_storing_standar:
1379 {
1380     \bool_if:NTF \l__enumext_standar_level_one_bool
1381     {
1382         \__enumext_storing_exec:
1383     }
1384     {
1385         \msg_warning:nnn { enumext } { save-ans-nested } { enumext }
1386     }
1387 }
1388 \cs_new_protected:Nn \__enumext_storing_starred:
1389 {
1390     \bool_if:NTF \l__enumext_starred_level_one_bool
1391     {
1392         \__enumext_storing_exec:
1393     }
1394     {
1395         \msg_warning:nnn { enumext } { save-ans-nested } { enumext* }
1396     }
1397 }
1398 \cs_new_protected:Nn \__enumext_storing_exec:
1399 {
1400     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1401     {
1402         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1403     }
1404     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1405     {
1406         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1407     }
1408     \bool_set_true:N \l__enumext_store_active_bool
1409     \bool_set_true:N \l__enumext_store_ans_bool
1410     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1411     {
1412         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1413     }
1414 }

```

(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*
1415 \cs_set_protected:Npn \__enumext_tmp:n #1
1416 {
1417     \keys_define:nn { enumext / #1 }
1418     {
1419         series .str_set:N = \l__enumext_series_str,
1420         series .value_required:n = true,
1421         resume .code:n = \__enumext_resume_series:n {##1},
1422         resume* .code:n = \__enumext_resume_starred:,
1423         resume* .value_forbidden:n = true,
1424     }

```



```

1425     }
1426     \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.

```

1427 \cs_new:Npn \__enumext_filter_series:n #1
1428 {
1429     \use:e
1430     {
1431         \keyval_parse:NNn
1432         \__enumext_filter_series_key:n
1433         \__enumext_filter_series_pair:nn {#1}
1434     }
1435 }

```

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.

```

1436 \cs_new:Npn \__enumext_filter_series_key:n #1
1437 {
1438     \str_case:nnF {#1}
1439     {
1440         { resume } {}
1441         { resume* } {}
1442     }
1443     { , { \exp_not:n {#1} } }
1444 }

```

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.

```

1445 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1446 {
1447     \str_case:nnF {#1}
1448     {
1449         { series } {}
1450         { resume } {}
1451         { save-key } {}
1452         { save-ans } {}
1453     }
1454     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1455 }

```

(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).

```

1456 \cs_new_protected:Npn \__enumext_parse_series_name:n #1
1457 {
1458     \str_if_empty:NTF \l__enumext_series_str
1459     {
1460         \__enumext_resume_last:n {#1}
1461     }
1462     {
1463         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1464         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1465         { \__enumext_filter_series:n {#1} }
1466         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1467         {
1468             \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1469         }
1470     }
1471 }

```

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.

```

1472 \cs_new_protected:Npn \__enumext_resume_last:n #1
1473 {
1474   \bool_if:NT \l__enumext_standar_level_one_bool
1475   {
1476     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT]]}
1477     \tl_gclear:N \g__enumext_standar_series_tl
1478     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1479   }
1480   \bool_if:NT \l__enumext_starred_level_one_bool
1481   {
1482     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT*]]}
1483     \tl_gclear:N \g__enumext_starred_series_tl
1484     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1485   }
1486 }

```

(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
\__enumext_resume_counter:n
\__enumext_resume_starred:
  \__enumext_resume_counter_set:

```

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.

```

1487 \cs_new_protected:Npn \__enumext_resume_series:n #1
1488 {
1489   \tl_if_empty:nTF {#1}
1490   {
1491     \__enumext_resume_counter:n { }
1492   }
1493   {
1494     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1495     {
1496       \__enumext_resume_counter:n {#1}
1497       \bool_if:NT \g__enumext_standar_bool
1498       {
1499         \keys_set:nv { enumext / level-1 }
1500         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1501       }
1502       \bool_if:NT \g__enumext_starred_bool
1503       {
1504         \keys_set:nv { enumext / enumext* }
1505         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1506       }
1507     }
1508     { \msg_error:nnn { enumext } { unknown-series } {#1} }
1509   }
1510 }
1511 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1512 {
1513   \tl_if_empty:nTF {#1}
1514   {
1515     \bool_if:NT \g__enumext_standar_bool
1516     {
1517       \int_gincr:N \g__enumext_resume_int
1518       \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int

```

```

1519     }
1520     \bool_if:NT \g__enumext_starred_bool
1521     {
1522         \int_gincr:N \g__enumext_resume_vii_int
1523         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1524     }
1525 }
1526 {
1527     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1528     \bool_if:NT \g__enumext_standar_bool
1529     {
1530         \int_set:Nn \l__enumext_start_i_int
1531         {
1532             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1533         }
1534     }
1535     \bool_if:NT \g__enumext_starred_bool
1536     {
1537         \int_set:Nn \l__enumext_start_vii_int
1538         {
1539             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1540         }
1541     }
1542 }
1543 \bool_lazy_and:nnT
1544 { \bool_if_p:N \l__enumext_standar_level_one_bool }
1545 { \bool_if_p:N \l__enumext_store_active_bool }
1546 {
1547     \int_set:Nn \l__enumext_start_i_int
1548     {
1549         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1550     }
1551 }
1552 \bool_lazy_and:nnT
1553 { \bool_if_p:N \l__enumext_starred_level_one_bool }
1554 { \bool_if_p:N \l__enumext_store_active_bool }
1555 {
1556     \int_set:Nn \l__enumext_start_vii_int
1557     {
1558         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1559     }
1560 }
1561 }
1562 \cs_new_protected:Nn \__enumext_resume_starred:
1563 {
1564     \bool_if:NT \g__enumext_standar_bool
1565     {
1566         \tl_if_empty:NF \g__enumext_standar_series_tl
1567         {
1568             \__enumext_resume_counter:n { }
1569             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1570         }
1571     }
1572     \bool_if:NT \g__enumext_starred_bool
1573     {
1574         \tl_if_empty:NF \g__enumext_standar_series_tl
1575         {
1576             \__enumext_resume_counter:n { }
1577             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1578         }
1579     }
1580 }
1581 \cs_new_protected:Nn \__enumext_resume_counter_set:
1582 {
1583     \bool_if:NT \g__enumext_standar_bool
1584     {
1585         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1586         {
1587             \int_gset_eq:cN { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1588         }

```

```

1589         \tl_if_empty:NF \l__enumext_series_str
1590         {
1591             \int_gset_eq:cN { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1592         }
1593     \tl_if_empty:NTF \l__enumext_resume_name_tl
1594     {
1595         \str_if_empty:NT \l__enumext_series_str
1596         {
1597             \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1598         }
1599     }
1600     {
1601         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1602         {
1603             \int_gset_eq:cN { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1604         }
1605     }
1606 }
1607 \bool_if:NT \g__enumext_starred_bool
1608 {
1609     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1610     {
1611         \int_gset_eq:cN { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1612     }
1613     \tl_if_empty:NF \l__enumext_series_str
1614     {
1615         \int_gset_eq:cN { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1616     }
1617     \tl_if_empty:NTF \l__enumext_resume_name_tl
1618     {
1619         \str_if_empty:NT \l__enumext_series_str
1620         {
1621             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1622         }
1623     }
1624     {
1625         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1626         {
1627             \int_gset_eq:cN { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1628         }
1629     }
1630 }
1631 }

```

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

10.23 The check answer mechanism

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

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

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

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

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

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

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

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

10.23.1 Setting check-ans key

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

check-ans

no-store

```

1632 \cs_set_protected:Npn \__enumext_tmp:n #1
1633 {
1634   \keys_define:nn { enumext / #1 }
1635   {
1636     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1637     check-ans .initial:n = false,
1638     no-store .code:n = {
1639                           \bool_set_false:N \l__enumext_store_ans_bool
1640                           \bool_set_false:N \l__enumext_check_ans_bool
1641                         },
1642     no-store .value_forbidden:n = true,
1643   }
1644 }
1645 \clist_map_inline:nn
1646 {
1647   level-1, level-2, level-3, level-4, enumext*
1648 }
1649 { \__enumext_tmp:n {#1} }

```

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

10.23.2 Set-up check answer mechanism

__enumext_check_ans_set:

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

```

1650 \cs_new_protected:Nn \__enumext_check_ans_set:
1651 {
1652   \int_case:nn { \l__enumext_level_int }
1653   {
1654     { 1 }{
1655       \bool_lazy_all:nT
1656       {
1657         { \bool_if_p:N \g__enumext_starred_bool }
1658         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1659       }
1660       {
1661         \int_gdecr:N \g__enumext_count_item_number_int
1662         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1663       }
1664     }
1665     { 2 }{
1666       \int_gdecr:N \g__enumext_count_item_number_int
1667     }
1668     { 3 }{
1669       \int_gdecr:N \g__enumext_count_item_number_int
1670     }
1671     { 4 }{
1672       \int_gdecr:N \g__enumext_count_item_number_int
1673     }
1674   }
1675   \int_case:nn { \l__enumext_level_h_int }
1676   {
1677     { 1 }{
1678       \bool_if:NT \g__enumext_standar_bool
1679       {
1680         \int_gdecr:N \g__enumext_count_item_number_int
1681         \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1682       }
1683     }
1684   }
1685 }

```

(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.

```

1686 \cs_new_protected:Nn \__enumext_check_ans_exec:
1687 {
1688   \bool_if:NT \l__enumext_check_ans_bool
1689   {
1690     \__enumext_check_ans_set:
1691   }
1692 }
```

(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.

```

1693 \cs_new_protected:Nn \__enumext_check_ans_show:
1694 {
1695   \int_compare:nNnTF
1696   { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1697   {
1698     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1699   }
1700   {
1701     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1702   }
1703   \int_gzero:N \g__enumext_count_item_number_int
1704   \int_gzero:N \g__enumext_count_item_anskey_int
1705 }
```

(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*`.

```

1706 \cs_set_protected:Npn \__enumext_tmp:n #1
1707 {
1708   \keys_define:nn { enumext / #1 }
1709   {
1710     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1711     wrap-ans .initial:n = \fbox{##1},
1712     wrap-ans .value_required:n = true,
1713     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1714     wrap-opt .initial:n = [{##1}],
1715     wrap-opt .value_required:n = true,
1716     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1717     save-sep .initial:n = {, ~ },
1718     save-sep .value_required:n = true,
1719     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1720     mark-ans .initial:n = \textasteriskcentered,
1721     mark-ans .value_required:n = true,
1722     mark-pos .choice:,
1723     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1724     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1725     mark-pos .initial:n = right,
1726     mark-pos .value_required:n = true,
1727     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1728     show-ans .initial:n = false,
1729     show-ans .value_required:n = true,
1730     show-pos .bool_set:N = \l__enumext_show_position_bool,
1731     show-pos .initial:n = false,
1732     show-pos .value_required:n = true,
1733     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
1734     mark-ref .initial:n = \textasteriskcentered,
1735     mark-ref .value_required:n = true,
1736     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1737     save-ref .initial:n = false,
```

```

1738         save-ref .value_required:n = true,
1739     }
1740 }
1741 \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`.

```

1742 \cs_set_protected:Npn \__enumext_tmp:n #1
1743 {
1744     \keys_define:nn { enumext / #1 }
1745     {
1746         mark-pos .choice:,
1747         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1748         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1749         mark-pos .initial:n = right,
1750         mark-pos .value_required:n = true,
1751         show-ans .bool_set:N = \l__enumext_show_answer_bool,
1752         show-ans .initial:n = false,
1753         show-ans .value_required:n = true,
1754         show-pos .bool_set:N = \l__enumext_show_position_bool,
1755         show-pos .initial:n = false,
1756         show-pos .value_required:n = true,
1757     }
1758 }
1759 \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*`. 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.

```

1760 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1761 {
1762     \keys_define:nn { enumext / #1 }
1763     {
1764         columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1765         \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1766         \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1767             {
1768                 columns = \exp_not:v { l__enumext_store_columns_#2_int },
1769             },
1770         columns* .value_required:n = true,
1771         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1772         \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1773         \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1774             {
1775                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1776             },
1777         columns-sep* .value_required:n = true,
1778     }
1779 }
1780 \clist_map_inline:nn
1781 {
1782     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1783 }
1784 { \__enumext_tmp:nn #1 }

```

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

10.24.1 Function for storing content in prop list

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

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

```

1785 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1786 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1

```



```

1787 {
1788   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1789   {
1790     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1791   }
1792   { #1 }
1793 }
1794 \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

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.

```

1795 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1796 {
1797   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1798 }
1799 \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

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

```

1800 \cs_new_protected:Nn \__enumext_store_level_open:
1801 {
1802   \bool_if:NT \l__enumext_store_ans_bool
1803   {
1804     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1805     {
1806       \__enumext_store_addto_seq:n
1807       {
1808         \item \begin{enumext}
1809       }
1810     }
1811     {
1812       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1813       {
1814         \item \begin{enumext} [
1815       }
1816       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1817       {
1818       ]
1819       }
1820       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1821     }
1822   }
1823 }
1824 \cs_new_protected:Nn \__enumext_store_level_close:
1825 {
1826   \bool_if:NT \l__enumext_store_ans_bool
1827   {
1828     \__enumext_store_addto_seq:n { \end{enumext} }
1829   }
1830 }

```

(End of definition for `__enumext_store_level_open:` and `__enumext_store_level_close:.`)

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

```

\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:

```

into account `\baselineskip`, the value of `\parsep` of the current level of `enumext` and the value of `\topsep` of the `enumext*` environment.

```

1831 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1832 {
1833   \bool_if:NT \l__enumext_store_ans_bool
1834   {
1835     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1836     {
1837       \__enumext_store_addto_seq:n
1838       {
1839         \item \mode_leave_vertical:
1840         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1841         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1842       }
1843     }
1844     {
1845       \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1846       {
1847         \item \mode_leave_vertical:
1848         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1849         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1850       }
1851       \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1852       {
1853       ]
1854       }
1855       \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1856     }
1857   }
1858 }
1859 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1860 {
1861   \bool_if:NT \l__enumext_store_ans_bool
1862   {
1863     \__enumext_store_addto_seq:n { \end{enumext*} }
1864   }
1865 }

```

(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`

```

1866 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1867 {
1868   \mode_leave_vertical:
1869   \skip_horizontal:n { -\dim_use:N #2 }
1870   \makebox[\opt][ r ]
1871   {
1872     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1873     {
1874       \tl_use:N \l__enumext_mark_answer_sym_tl
1875     }
1876   }
1877   \skip_horizontal:n { \dim_use:N #2 }
1878 }
1879 \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`.

```

1880 \keys_define:nn { enumext / anskey }
1881 {
1882   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1883   item-sym* .value_required:n = true,
1884   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1885   item-pos* .value_required:n = true,
1886   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1887   store-brk .default:n = true,
1888   store-brk .value_forbidden:n = true,
1889 }

```

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>`.

```

1890 \NewDocumentCommand \anskey { d() s o +m }
1891 {
1892   \bool_if:NF \l__enumext_store_active_bool
1893   {
1894     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1895   }
1896   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
1897   {
1898     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1899   }
1900   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1901   {
1902     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1903   }
1904   \group_begin:
1905     \bool_if:NT \l__enumext_store_ans_bool
1906     {
1907       \bool_if:NT \l__enumext_check_ans_bool
1908       {
1909         \int_gincr:N \g__enumext_count_item_anskey_int
1910       }
1911       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1912     }
1913   \group_end:
1914 }

```

(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.

```

1915 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1916 {
1917   \__enumext_store_addto_prop:n {#4}
1918   \bool_if:NT \l__enumext_store_ref_key_bool
1919   {
1920     \__enumext_store_internal_ref:
1921   }
1922   \__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`.

```

1923   \tl_clear:N \l__enumext_store_anskey_arg_tl
1924   \tl_if_novalue:nF {#3}
1925   {
1926     \keys_set:nn { enumext / anskey } {#3}
1927   }
1928   \bool_lazy_and:nnT
1929   { \bool_if_p:N \l__enumext_store_columns_break_bool }

```

```

1930     { \bool_not_p:n { \l__enumext_starred_bool } }
1931     {
1932         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1933     }
1934     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

1935     \tl_if_novalue:nF {#1}
1936     {
1937         \int_set:Nn \l__enumext_store_columns_join_int {#1}
1938         \bool_if:NT \l__enumext_starred_bool
1939         {
1940             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1941             {
1942                 ( \exp_not:V \l__enumext_store_columns_join_int )
1943             }
1944         }
1945     }

```

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`.

```

1946     \bool_if:nTF {#2}
1947     {
1948         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1949         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1950         {
1951             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1952             {
1953                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1954             }
1955         }
1956         \dim_compare:nT
1957         {
1958             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1959         }
1960         {
1961             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1962             {
1963                 [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1964             }
1965         }
1966         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1967     }
1968     {
1969         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1970     }

```

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

```

1971     \bool_lazy_and:nnT
1972     { \bool_if_p:N \l__enumext_store_ref_key_bool }
1973     { \bool_if_p:N \l__enumext_hyperref_bool }
1974     {
1975         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1976         {
1977             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1978             { \exp_not:V \l__enumext_mark_ref_sym_tl }
1979         }
1980     }
1981     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1982 }

```

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

`__enumext_store_internal_ref:`

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

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

```

1983 \cs_new_protected:Nn \__enumext_store_internal_ref:
1984 {

```

```

1985 \cs_set_protected:Npn \__enumext_tmp:n ##1
1986 {
1987   \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
1988   \tl_reverse:c { \__enumext_label_copy_##1_tl }
1989   \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
1990   \tl_reverse:c { \__enumext_label_copy_##1_tl }
1991 }
1992 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1993 \cs_set:Npn \__enumext_tmp:n ##1
1994 { . \tl_use:c { \__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.

```

1995 \bool_lazy_all:nT
1996 {
1997   { \bool_if_p:N \g__enumext_starred_bool }
1998   { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
1999 }
2000 {
2001   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2002   { \tl_use:N \__enumext_label_copy_vii_tl }
2003 }
2004 \bool_lazy_all:nT
2005 {
2006   { \bool_if_p:N \l__enumext_standar_bool }
2007   { \bool_if_p:N \g__enumext_starred_bool }
2008   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2009 }
2010 {
2011   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2012   {
2013     \tl_use:N \__enumext_label_copy_vii_tl
2014     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2015   }
2016 }

```

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.

```

2017 \bool_lazy_all:nT
2018 {
2019   { \bool_if_p:N \l__enumext_standar_bool }
2020   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2021   { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_zero_int } }
2022   { \bool_not_p:n { \__enumext_starred_bool } }
2023 }
2024 {
2025   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2026   {
2027     \tl_use:N \__enumext_label_copy_i_tl
2028     \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
2029   }
2030 }
2031 \cs_set:Npn \__enumext_tmp:n ##1
2032 { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2033 \bool_lazy_all:nT
2034 {
2035   { \bool_if_p:N \l__enumext_standar_bool }
2036   { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2037   { \bool_not_p:n { \g__enumext_starred_bool } }
2038   { \int_compare_p:nNn { \__enumext_level_h_int } > { \c_zero_int } }
2039 }
2040 {
2041   \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2042   {
2043     \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2044     . \tl_use:N \__enumext_label_copy_vii_tl
2045   }
2046 }

```

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

```

2047 \tl_put_right:Ne \__enumext_newlabel_arg_one_tl

```

```

2048     {
2049         \l__enumext_store_name_tl \c_colon_str
2050         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2051     }

```

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

```

2052     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2053     {
2054         \__enumext_newlabel:n
2055         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2056         { \l__enumext_newlabel_arg_two_tl }
2057     }
2058     \l__enumext_store_write_aux_file_tl
2059 }

```

(End of definition for `__enumext_store_internal_ref:.`)

`__enumext_store_anskey_show_wrap:n`

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

```

2060 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2061 {
2062     \par
2063     \bool_if:NT \l__enumext_starred_bool
2064     {
2065         \cs_set:Nn \__enumext_level: { vii }
2066     }
2067     \__enumext_print_keyans_box:cc
2068     { \l__enumext_labelwidth_ \__enumext_level: _dim }
2069     { \l__enumext_labelsep_ \__enumext_level: _dim }
2070     \__enumext_anskey_wrapper:n { #1 }
2071 }

```

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

`__enumext_store_anskey_show_left:n`

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

```

2072 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2073 {
2074     \bool_if:NT \l__enumext_show_answer_bool
2075     {
2076         \__enumext_store_anskey_show_wrap:n { #1 }
2077     }
2078     \bool_if:NT \l__enumext_show_position_bool
2079     {
2080         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2081         {
2082             \group_begin:
2083             \exp_not:N \normalfont
2084             \exp_not:N \footnotesize [ \int_eval:n
2085                 {
2086                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2087                 }
2088             ]
2089             \group_end:
2090         }
2091         \__enumext_store_anskey_show_wrap:n { #1 }
2092     }
2093 }

```

(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 $\langle prop\ list \rangle$ defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

2094 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2095 {
2096   \tl_clear:N \__enumext_store_keyans_label_tl
2097   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2098   {
2099     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_vi_tl }
2100   }
2101   {
2102     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_v_tl }
2103   }
2104   \tl_if_novalue:nF { #1 }
2105   {
2106     % Set save-sep
2107     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
2108     {
2109       \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
2110     }
2111     \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
2112   }
2113   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
2114 }

```

(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 $\langle label \rangle$. 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 $\langle labels \rangle$ and remove the dots “.” from them, we do not want to get double dots in our references.

```

2115 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2116 {
2117   \bool_if:NT \__enumext_store_ref_key_bool
2118   {
2119     \cs_set_protected:Npn \__enumext_tmp:n ##1
2120     {
2121       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2122       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2123       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2124       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2125     }
2126     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2127     \__enumext_keyans_store_ref_aux_i:
2128   }
2129 }

```

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

```

2130 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2131 {
2132   \bool_if:NT \g__enumext_starred_bool
2133   {
2134     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2135   }
2136   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2137   {
2138     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2139     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2140   }
2141   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2142   {
2143     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2144     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2145   }

```



```

2146 \int_compare:nNtT { \l__enumext_keyans_level_h_int } = { 1 }
2147 {
2148   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2149   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2150 }
2151 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2152 {
2153   \l__enumext_store_name_tl \c_colon_str
2154   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2155 }
2156 \__enumext_keyans_store_ref_aux_ii:
2157 }

```

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

```

2158 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2159 {
2160   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2161   {
2162     \__enumext_newlabel:nn
2163     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2164     { \l__enumext_newlabel_arg_two_tl }
2165   }
2166   \l__enumext_store_write_aux_file_tl
2167 }

```

(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 *label* `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the optional argument of both commands to the `\l__enumext_store_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

2168 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2169 {
2170   \tl_clear:N \l__enumext_store_keyans_label_tl
2171   \int_compare:nNtTF { \l__enumext_keyans_pic_level_int } = { 1 }
2172   {
2173     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2174   }
2175   {
2176     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2177   }
2178   \tl_if_novalue:nF { #1 }
2179   {
2180     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2181     {
2182       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2183     }
2184     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2185   }
2186   \__enumext_keyans_addto_seq_link:
2187 }

```

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.

```

2188 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2189 {
2190   \bool_lazy_and:nnT
2191   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2192   { \bool_if_p:N \l__enumext_hyperref_bool }
2193   {
2194     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2195     {
2196       \hfill \exp_not:N \hyperlink

```

```

2197         {
2198             \exp_not:V \l__enumext_newlabel_arg_one_tl
2199         }
2200         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2201     }
2202 }
2203 \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2204 \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2205 \bool_if:NT \l__enumext_check_ans_bool
2206 {
2207     \int_gincr:N \g__enumext_count_item_anskey_int
2208 }
2209 }

```

(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.

```

2210 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2211 {
2212     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2213     {
2214         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2215     }
2216     { \tl_gclear:N \g__enumext_check_ans_item_tl }
2217 }

```

(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.

```

2218 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2219 {
2220     \tl_if_novalue:nF { #1 }
2221     {
2222         \tl_set:Nc \l__enumext_keyans_item_opt_tl { #1 }
2223     }
2224     \bool_if:NT \l__enumext_show_answer_bool
2225     {
2226         \__enumext_keyans_show_ans:
2227     }
2228     \bool_if:NT \l__enumext_show_position_bool
2229     {
2230         \__enumext_keyans_show_pos:
2231     }
2232 }
2233 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2234 {
2235     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2236     {
2237         \bool_lazy_or:nnT
2238         { \bool_if_p:N \l__enumext_show_answer_bool }
2239         { \bool_if_p:N \l__enumext_show_position_bool }
2240         {
2241             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2242         }
2243     }
2244 }
2245 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2246 {
2247     \tl_put_left:Nn \l__enumext_label_v_tl
2248     {
2249         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2250     }

```

```

2251   }
2252   \cs_new_protected:Nn \__enumext_keyans_show_pos:
2253   {
2254     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2255     {
2256       \tl_set:Nc \__enumext_mark_answer_sym_tl
2257       {
2258         \group_begin:
2259         \exp_not:N \normalfont
2260         \exp_not:N \footnotesize [ \int_eval:n
2261         {
2262           \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
2263         }
2264         ]
2265         \group_end:
2266       }
2267     }
2268     {
2269       \tl_set:Nc \__enumext_mark_answer_sym_tl
2270       {
2271         \group_begin:
2272         \exp_not:N \normalfont
2273         \exp_not:N \footnotesize [ \int_eval:n
2274         {
2275           \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1
2276         }
2277         ]
2278         \group_end:
2279       }
2280     }
2281     \tl_put_left:Nn \__enumext_label_v_tl
2282     {
2283       \__enumext_print_keyans_box:NN
2284       \__enumext_labelwidth_i_dim
2285       \__enumext_labelsep_i_dim
2286     }
2287   }

```

(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*
2288 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2289 {
2290   \keys_define:nn { enumext / #1 }
2291   {
2292     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2293     item-sym* .value_required:n = true,
2294     item-sym* .initial:n = { $\star$ },
2295     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2296     item-pos* .value_required:n = true,
2297   }
2298 }
2299 \clist_map_inline:nn
2300 {
2301   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2302 }
2303 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining `\footnote` command

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

```

2304 \cs_new_protected:Nn \__enumext_footnotetext:nn
2305 {

```

```

2306     \footnotetext[#1]{#2}
2307   }
2308   \cs_new_protected:Nn \__enumext_renew_footnote:
2309   {
2310     \seq_gclear:N \g__enumext_footnote_arg_seq
2311     \seq_gclear:N \g__enumext_footnote_int_seq
2312     \RenewDocumentCommand \footnote { o +m }
2313     {
2314       \tl_if_novalue:nTF {##1}
2315       {
2316         \stepcounter{footnote}
2317         \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2318       }
2319       {
2320         \int_gset:Nn \g__enumext_footnote_int { ##1 }
2321       }
2322       \footnotemark [ \g__enumext_footnote_int ]
2323       \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2324       \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2325     }
2326   }
2327   \cs_new_protected:Nn \__enumext_print_footnote:
2328   {
2329     \seq_if_empty:NF \g__enumext_footnote_int_seq
2330     {
2331       \seq_map_pairwise_function:NNN
2332       \g__enumext_footnote_int_seq
2333       \g__enumext_footnote_arg_seq
2334       \__enumext_footnotetext:nn
2335     }
2336   }

```

(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 `\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).

```

2337   \cs_new_protected:Npn \__enumext_default_item:n #1
2338   {
2339     \tl_if_novalue:nTF {#1}
2340     {
2341       \bool_if:NT \l__enumext_check_ans_bool
2342       {
2343         \int_gincr:N \g__enumext_count_item_number_int
2344       }
2345       \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2346       \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2347     }
2348     {
2349       \bool_set_eq:cc
2350       { l__enumext_wrap_label_ \__enumext_level: _bool }
2351       { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2352       \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2353     }
2354   }

```

(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).

```

2355 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2356 {
2357   \tl_if_novalue:nF {#1}
2358   {
2359     \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2360   }
2361   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2362   \tl_if_novalue:nTF {#2}
2363   {
2364     \dim_set_eq:cc
2365     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2366     { \l__enumext_labelsep_ \__enumext_level: _dim }
2367   }
2368   {
2369     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2370   }
2371   \bool_if:NT \l__enumext_check_ans_bool
2372   {
2373     \int_gincr:N \g__enumext_count_item_number_int
2374   }
2375   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2376   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2377 }

```

(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).

```

2378 \cs_new_protected:Nn \__enumext_redefine_item:
2379 {
2380   \RenewDocumentCommand \item { s o o }
2381   {
2382     \bool_if:nTF {##1}
2383     {
2384       \__enumext_starred_item:nn {##2} {##3}
2385     }
2386     { \__enumext_default_item:n {##2} }
2387   }
2388 }

```

(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`.

```

2389 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2390 {
2391   \tl_if_novalue:nTF { #1 }
2392   {
2393     \bool_set_true:N \__enumext_wrap_label_v_bool
2394     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2395   }
2396   {
2397     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2398     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2399   }
2400 }
```

(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 $\langle label \rangle$.

```

2401 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2402 {
2403   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2404   \__enumext_keyans_show_left:n { #1 }
2405   \bool_set_true:N \__enumext_wrap_label_v_bool
2406   \__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 $\langle sequence \rangle$.

```

2407   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2408   \__enumext_keyans_addto_prop:n { #1 }
2409   \__enumext_keyans_store_ref:
2410   \__enumext_keyans_addto_seq:n { #1 }
2411 }
```

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

`\item*`

`__enumext_keyans_redefine_item:`

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.32).

```

2412 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2413 {
2414   \RenewDocumentCommand \item { s o }
2415   {
2416     \bool_if:nTF {##1}
2417     {
2418       \peek_remove_spaces:n
2419       {
2420         \__enumext_keyans_starred_item:n {##2}
2421       }
2422     }
2423     {
2424       \__enumext_keyans_default_item:n {##2}
2425     }
2426   }
2427 }
```

(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.

```
2428 \cs_new_protected:Nn \__enumext_item_starred:
2429 {
2430   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2431   {
2432     \mode_leave_vertical:
2433     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2434     \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
2435     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2436   }
2437 }
```

(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).

```
2438 \cs_new_protected:Nn \__enumext_make_label:
2439 {
2440   \RenewDocumentCommand \makeLabel { m }
2441   {
2442     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2443     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2444     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2445     {
2446       \__enumext_item_starred:
2447       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2448     }
2449     { ##1 }
2450     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2451     \tl_gclear:N \g__enumext_item_symbol_tl
2452   }
2453 }
```

(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).

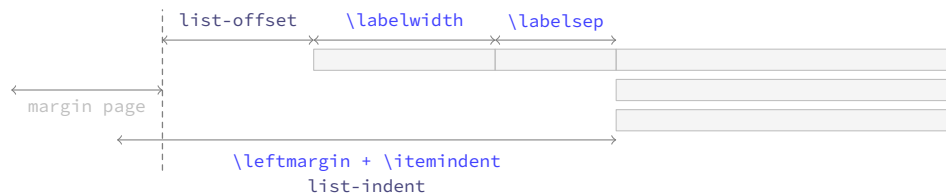
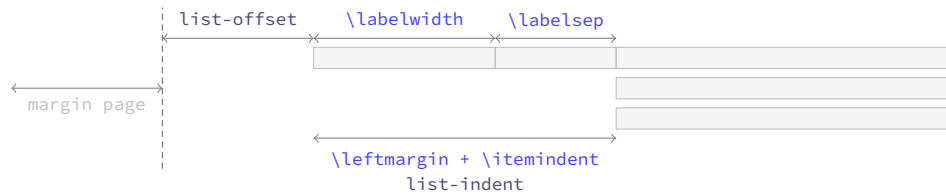
```
2454 \cs_new_protected:Nn \__enumext_keyans_make_label:
2455 {
2456   \RenewDocumentCommand \makeLabel { m }
2457   {
2458     \tl_use:N \l__enumext_label_fill_left_v_tl
2459     \tl_use:N \l__enumext_label_font_style_v_tl
2460     \bool_if:NTF \l__enumext_wrap_label_v_bool
2461     {
2462       \__enumext_wrapper_label_v:n { ##1 }
2463     }
2464     { ##1 }
2465     \tl_use:N \l__enumext_label_fill_right_v_tl
2466   }
2467 }
```

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

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

Figure 9: Representation of standard horizontal lengths in `list` environment.Figure 10: Representation of horizontal lengths concept in list in `enumext`.

```
#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).

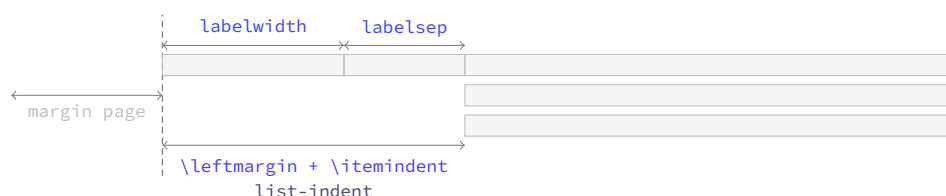
```
2468 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
2469 {
2470   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2471   {
2472     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2473     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2474   }
2475   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2476   {
2477     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2478     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2479   }
2480 }
```

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

```
2480 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2481 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2482 {
2483   \dim_set:Nn #6 { #1 + #2 - #4 }
2484   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2485 }
2486 {
2487   \dim_compare:nNnT { #4 } = { #1 + #2 }
2488   { \dim_set:Nn #6 { \c_zero_dim } }
2489   \dim_compare:nNnT { #4 } < { #1 + #2 }
2490   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2491   \dim_compare:nNnT { #4 } > { #1 + #2 }
2492   {
2493     \dim_set:Nn #6 { -#1 - #2 + #4 }
2494     \dim_set:Nn #6 { #6*-1 }
2495   }
2496 }
```

Figure 11: Default horizontal lengths in `enumext`.

```

2496         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2497     }
2498 }
2499 \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.

```

2500 \cs_set_protected:Npn \__enumext_tmp:n #1
2501 {
2502     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2503     {
2504         \__enumext_calc_hspace:ccccc
2505         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2506         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2507         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2508         { \__enumext_leftmargin_tmp_#1_bool }
2509         \clist_map_inline:nn
2510         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2511         { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2512         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2513         { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2514         \usecounter { enumX#1 }
2515         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2516         \str_if_eq:nnTF {#1} { v }
2517         {
2518             \__enumext_keyans_redefine_item:
2519             \__enumext_keyans_make_label:
2520             \__enumext_keyans_fake_item:
2521             \bool_if:cT { \__enumext_show_length_#1_bool }
2522             {
2523                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2524             }
2525         }
2526         {
2527             \__enumext_redefine_item:
2528             \__enumext_make_label:
2529             \__enumext_use_key_ref:
2530             \__enumext_fake_item:
2531             \bool_if:cT { \__enumext_show_length_#1_bool }
2532             {
2533                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2534             }
2535         }
2536     }
2537 }
2538 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

2539 \cs_set_protected:Npn \__enumext_tmp:n #1
2540 {
2541     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2542     {
2543         \__enumext_calc_hspace:ccccc
2544         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2545         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }

```

```

2546     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2547     { \__enumext_leftmargin_tmp_#1_bool }
2548 \clist_map_inline:nn
2549     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2550     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2551 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2552     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2553 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2554 \skip_zero:N \partopsep
2555 \usecounter { enumX#1 }
2556 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2557 \__enumext_use_key_ref_h:
2558 \str_if_eq:nnTF {#1} { vii }
2559     {
2560         \__enumext_fake_item_vii:
2561         \bool_if:cT { \__enumext_show_length_vii_bool }
2562             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2563     }
2564     {
2565         \__enumext_fake_item_viii:
2566         \bool_if:cT { \__enumext_show_length_#1_bool }
2567             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2568     }
2569 }
2570 }
2571 \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.

```

2572 \NewDocumentEnvironment{enumext}{0}{ }
2573 {
2574     \__enumext_safe_exec:
2575     \__enumext_parse_keys:n {#1}
2576     \__enumext_before_list:
2577     \__enumext_start_store_level:
2578     \__enumext_start_list:nn
2579     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2580     {
2581         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2582         \__enumext_before_keys_exec:
2583     }
2584     \__enumext_after_args_exec:
2585 }
2586 {
2587     \__enumext_stop_list:
2588     \__enumext_stop_store_level:
2589     \__enumext_after_list:
2590 }

```

(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.

```

2591 \cs_new_protected:Nn \__enumext_safe_exec:
2592 {
2593     \__enumext_current_env_set_bool:
2594     \int_incr:N \__enumext_level_int
2595     \int_compare:nNnT { \__enumext_level_int } > { 4 }
2596         { \msg_fatal:nn { enumext } { list-too-deep } }
2597     \bool_set_true:N \l__enumext_standar_bool
2598     \bool_lazy_all:nT
2599     {
2600         { \bool_if_p:N \g__enumext_standar_bool }
2601         { \int_compare_p:nNn { \__enumext_level_int } = { 1 } }
2602         { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2603     }
2604     {
2605         \typeout{[[ON-FIRST-LEVEL-ENUMEXT-NOT-NESTED]]}

```

```

2606         \bool_set_true:N \l__enumext_standar_level_one_bool
2607     }
2608 }

```

(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.

```

2609 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2610 {
2611     \tl_if_novalue:nF {#1}
2612     {
2613         \str_clear:N \l__enumext_series_str
2614         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2615         {
2616             \keys_set:nn { enumext / level-1 } {#1}
2617             \__enumext_parse_series_name:n {#1}
2618         }
2619         {
2620             \exp_args:Ne \keys_set:nn
2621             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2622         }
2623         \bool_if:NT \l__enumext_store_active_bool
2624         {
2625             \__enumext_parse_store_keys:n {#1}
2626         }
2627     }
2628 }

```

(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:`.

```

2629 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2630 {
2631     \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2632     {
2633         \regex_match:nnT { \b columns\b } {#1}
2634         {
2635             \int_set_eq:cc
2636             { \l__enumext_store_columns_ \__enumext_level: _int }
2637             { \l__enumext_columns_ \__enumext_level: _int }
2638             \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2639             {
2640                 columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2641             }
2642         }
2643     }
2644     \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2645     {
2646         \regex_match:nnT { \b columns-sep\b } {#1}
2647         {
2648             \dim_set_eq:cc
2649             { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2650             { \l__enumext_columns_sep_ \__enumext_level: _dim }
2651             \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2652             {
2653                 columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2654             }
2655         }
2656     }
2657 }

```

(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.

```

2658 \cs_new_protected:Nn \__enumext_start_store_level:
2659 {
2660   \bool_lazy_all:nT
2661   {
2662     { \bool_if_p:N \l__enumext_store_active_bool }
2663     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2664     { \bool_not_p:n { \g__enumext_starred_bool } }
2665   }
2666   {
2667     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2668     {
2669       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2670       \__enumext_store_level_open:
2671     }
2672   }
2673   \bool_lazy_all:nT
2674   {
2675     { \bool_if_p:N \l__enumext_store_active_bool }
2676     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2677     { \bool_if_p:N \g__enumext_starred_bool }
2678   }
2679   {
2680     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2681     {
2682       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2683       \__enumext_store_level_open:
2684     }
2685   }
2686 }
2687 \cs_new_protected:Nn \__enumext_stop_store_level:
2688 {
2689   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2690   {
2691     \__enumext_store_level_close:
2692   }
2693 }

```

(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.

```

2694 \cs_new_protected:Nn \__enumext_before_list:
2695 {
2696   \__enumext_vspace_above:
2697   \__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.

```

2698   \__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.

```

2699   \dim_compare:nNnT
2700   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2701   {
2702     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2703     {
2704       \linewidth
2705       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2706       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2707     }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_mini_addvspace:` is called and the `__enumext_mini_env*` environment on the “left side” will be initialized followed by the “vertical spacing” applied to preserve the “baseline” between

the *left* and *right* side environments. After these actions, the function `__enumext_multicols_start:` is called to handle the `multicols` environment.

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

```

2708     \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
2709     \int_gincr:N \g__enumext_minipage_stat_int
2710     \__enumext_mini_addvspace:
2711     \nointerlineskip\noindent
2712     \begin{\__enumext_mini_env*}
2713     { \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
2714     }
2715     \__enumext_multicols_start:
2716     }

```

(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.

```

2717 \cs_new_protected:Nn \__enumext_multicols_start:
2718 {
2719     \int_compare:nNt
2720     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2721     {
2722         \dim_compare:nNt
2723         { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2724         {
2725             \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
2726             {
2727                 ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
2728                   + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
2729                   ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
2730                   - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
2731                 )
2732             }
2733             \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
2734             \skip_zero:N \multicolsep
2735             \int_compare:nNt { \__enumext_level_int } > { 1 }
2736             {
2737                 \dim_zero:N \columnseprule
2738             }
2739         }
2740     }

```

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.

```

2739     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2740     {
2741         \__enumext_multi_addvspace:
2742     }
2743     \raggedcolumns
2744     \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2745     }
2746     }

```

(End of definition for `__enumext_multicols_start:`)

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

```

2747 \cs_new_protected:Nn \__enumext_multicols_stop:
2748 {
2749     \int_compare:nNt
2750     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2751     {
2752         \end{multicols}
2753         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2754         {
2755             \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2756         }
2757     }

```

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.

```

2758   \bool_lazy_and:nnT
2759   { \bool_if_p:N \l__enumext_check_ans_bool }
2760   { \bool_not_p:n { \g__enumext_starred_bool } }
2761   {
2762     \bool_gset_true:N \g__enumext_check_ans_show_bool
2763     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2764   }
2765 }

```

(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.

```

2766 \cs_new_protected:Nn \__enumext_after_list:
2767 {
2768   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2769   {
2770     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2771     {
2772       \msg_warning:nn { enumext } { missing-miniright }
2773       \miniright
2774     }
2775     \int_gzero:N \g__enumext_minipage_stat_int
2776     \end{\__enumext_mini_env*}
2777     \par\addvspace { \l__enumext_minipage_after_skip }
2778   }
2779   { \__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.

```

2780   \__enumext_after_stop_list:
2781   \__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.

```

2782   \bool_set_false:N \l__enumext_standar_bool
2783   \__enumext_resume_counter_set:
2784 }

```

(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`.

```

2785 \__enumext_after_env:nn {enumext}
2786 {
2787   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2788   {
2789     \bool_if:NT \g__enumext_check_ans_show_bool
2790     {
2791       \__enumext_check_ans_show:
2792     }
2793     \bool_gset_false:N \g__enumext_standar_bool
2794     \bool_gset_false:N \g__enumext_check_ans_show_bool
2795     \tl_gclear:N \g__enumext_store_name_tl
2796   }
2797 }

```

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.

```

2798 \NewDocumentEnvironment{keyans}{ 0{ } }
2799 {
2800   \__enumext_keyans_safe_exec:
2801   \__enumext_keyans_parse_keys:n {#1}
2802   \__enumext_before_list_v:
2803   \__enumext_start_list:nn
2804   { \tl_use:N \l__enumext_label_v_tl }
2805   {
2806     \__enumext_list_arg_two_v:
2807     \__enumext_before_keys_exec_v:
2808   }
2809   \__enumext_after_args_exec_v:
2810 }
2811 {
2812   \__enumext_keyans_check_ans:nn { item }{ keyans }
2813   \__enumext_stop_list:
2814   \__enumext_after_list_v:
2815 }

```

(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.

```

2816 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2817 {
2818   \bool_if:NF \l__enumext_store_active_bool
2819   {
2820     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2821   }
2822   \int_incr:N \l__enumext_keyans_level_int
2823   \bool_set_true:N \l__enumext_keyans_env_bool
2824   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2825   \bool_set_false:N \l__enumext_store_active_bool
2826   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2827   {
2828     \msg_error:nn { enumext } { keyans-nested }
2829   }
2830   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2831   {
2832     \msg_error:nn { enumext } { keyans-wrong-level }
2833   }
2834 }

```

(End of definition for *__enumext_keyans_safe_exec:*.)

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

```

2835 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2836 {
2837   \keys_set:nn { enumext / keyans } {#1}
2838 }

```

(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.

```

2839 \cs_new_protected:Nn \__enumext_before_list_v:
2840 {
2841   \__enumext_vspace_above_v:
2842   \__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.

```

2843   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2844   {
2845     \dim_set:Nn \l__enumext_minipage_left_v_dim
2846     {

```

```

2847         \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2848     }

```

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_minienv*` 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*.

```

2849     \bool_set_true:N \l__enumext_minipage_active_v_bool
2850     \int_gincr:N \g__enumext_minipage_stat_int
2851     \__enumext_keyans_mini_addvspace:
2852     \nointerlineskip\noindent
2853     \begin{\__enumext_minienv*}{ \l__enumext_minipage_left_v_dim }
2854 }

```

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

```

2855 \__enumext_keyans_multicols_start:
2856 }

```

(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.

```

2857 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2858 {
2859     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2860     {

```

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

```

2861         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2862         {
2863             \dim_set:Nn \l__enumext_columns_sep_v_dim
2864             {
2865                 (
2866                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2867                 ) / \l__enumext_columns_v_int
2868                 - \l__enumext_listoffset_v_dim
2869             }
2870         }
2871         \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).

```

2872         \skip_zero:N \multicolsep
2873         \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.

```

2874         \bool_if:NF \l__enumext_minipage_active_v_bool
2875         {
2876             \__enumext_keyans_multi_addvspace:
2877         }
2878         \raggedcolumns
2879         \begin{multicols}{ \l__enumext_columns_v_int }
2880     }
2881 }

```

(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_minienv*`) we will apply our vertical “adjust” spacing.

```

2882 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2883 {
2884     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2885     {
2886         \end{multicols}
2887         \bool_if:NF \l__enumext_minipage_active_v_bool
2888         {

```

```

2889         \par\addvspace{ \l__enumext_multicols_below_v_skip }
2890     }
2891 }
2892 }

```

(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.

```

2893 \cs_new_protected:Nn \__enumext_after_list_v:
2894 {
2895     \bool_if:NTF \l__enumext_minipage_active_v_bool
2896     {
2897         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2898         {
2899             \msg_warning:nn { enumext } { missing-miniright }
2900             \miniright
2901         }
2902         \int_gzero:N \g__enumext_minipage_stat_int
2903         \end{\__enumext_mini_env*}
2904         \par\addvspace{ \l__enumext_minipage_after_skip }
2905     }
2906     { \__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.

```

2907     \bool_set_false:N \l__enumext_keyans_env_bool
2908     \__enumext_after_stop_list_v:
2909     \__enumext_vspace_below_v:
2910 }

```

(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.

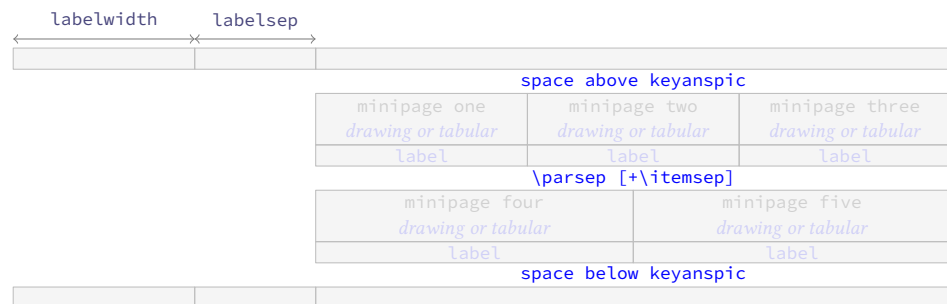


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

This implementation is adapted from the answer given by Enrico Gregorio in [How to process the body of an environment and divide it by a \macro?](#).

10.35.1 The command `\anspic`

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

```

2911 \NewDocumentCommand \anspic { s o +m }
2912 {

```

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

```

2913 \bool_if:NF \l__enumext_store_active_bool
2914 {
2915   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2916 }
2917 \int_compare:nNt { \l__enumext_level_int } > { 1 }
2918 {
2919   \msg_error:nn { enumext } { keyanspic-wrong-level }
2920 }
2921 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2922 {
2923   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2924 }

```

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.

```

2925 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2926 {
2927   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2928 }
2929 }

```

(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.

```

2930 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2931 {
2932   \stepcounter { enumXvi }
2933   #3 \\\
2934   \bool_if:nT { #1 }
2935   {
2936     \__enumext_keyans_addto_prop:n { #2 }
2937     \__enumext_keyans_store_ref:
2938     \__enumext_keyans_addto_seq:n { #2 }
2939     \bool_lazy_or:nnT
2940     { \bool_if_p:N \l__enumext_show_answer_bool }
2941     { \bool_if_p:N \l__enumext_show_position_bool }
2942     {
2943       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2944       \__enumext_keyans_show_left:n { #2 }
2945       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2946     }
2947   }
2948   \tl_use:N \l__enumext_label_font_style_v_tl
2949   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2950 }

```

(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.

```

2951 \NewDocumentEnvironment{keyanspic}{ o }
2952 {
2953   \__enumext_keyans_pic_safe_exec:
2954   \__enumext_start_list:nn
2955   { }
2956   {
2957     \__enumext_keyans_pic_arg_two:
2958   }

```

We apply the “adjusted” vertical spacing above the environment

```

2959 \vspace { \l__enumext_keyans_pic_above_skip }
2960 }

```

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.

```

2961 {
2962   \tl_if_novalue:nTF { #1 }
2963   {
2964     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2965   }
2966   { \__enumext_keyans_pic_do:n { #1 } }
2967   \__enumext_stop_list:
2968   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2969   \setcounter { enumXvi } { 0 }
2970   \vspace { \l__enumext_topsep_v_skip }
2971   %\bool_set_false:N \l__enumext_store_active_bool
2972 }

```

(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.

```

2973 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2974 {
2975   \int_incr:N \l__enumext_keyans_pic_level_int
2976   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2977   {
2978     \msg_error:nn { enumext } { keyanspic-nested }
2979   }
2980 }

```

(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`.

```

2981 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2982 {
2983   \dim_compare:nNnT { #1 } < { 0pt }
2984   { \skip_set:Nn #1 { -#1 } }
2985 }

```

(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.

```

2986 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2987 {

```

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.

```

2988   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2989   \__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.

```

2990   \skip_add:Nn \parsep { \itemsep }
2991   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2992   \dim_zero:N \labelwidth
2993   \dim_zero:N \listparindent
2994   \dim_zero:N \labelsep
2995   \skip_zero:N \partopsep
2996   \skip_zero:N \itemsep

```

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

```

2997   \__enumext_keyans_pic_skip_abs:N \parsep
2998   \skip_set:Nn \l__enumext_keyans_pic_above_skip
2999   {
3000     \box_dp:N \strutbox

```

```

3001         + \l__enumext_topsep_v_skip
3002         - \parsep
3003     }
3004     \__enumext_item_std:w \scan_stop:
3005 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

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

```

3006 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3007 {
3008     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3009 }
3010 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

__enumext_keyans_pic_row:n

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

```

3011 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3012 {
3013     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3014     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3015     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3016     \int_step_inline:nnn
3017     { \l__enumext_keyans_pic_above_int + 1 }
3018     { \l__enumext_keyans_pic_below_int }
3019     {
3020         \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3021         \centering
3022         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3023         \__enumext_endminipage:
3024     }
3025     \par
3026 }

```

(End of definition for __enumext_keyans_pic_row:n.)

10.36 The environment enumext*

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

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

10.36.1 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.

```

3027 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3028 {
3029     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3030     {
3031         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3032         {
3033             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3034             / \l__enumext_columns_vii_int
3035         }
3036     }
3037     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3038     \dim_set:Nn \l__enumext_item_width_vii_dim
3039     {
3040         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3041         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim

```

```

3042     - \l__enumext_labelsep_vii_dim
3043   }
3044   \dim_zero_new:N \itemwidth
3045 }

```

(End of definition for \l__enumext_starred_columns_set_vii:.)

\l__enumext_starred_joined_item_vii:n

The function \l__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.

```

3046 \cs_new_protected:Npn \l__enumext_starred_joined_item_vii:n #1
3047 {
3048   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3049   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3050   {
3051     \msg_warning:nnee { enumext } { item-joined }
3052     { \int_use:N \l__enumext_joined_item_vii_int }
3053     { \int_use:N \l__enumext_columns_vii_int }
3054     \int_set:Nn \l__enumext_joined_item_vii_int
3055     {
3056       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3057     }
3058   }
3059   \int_compare:nNnT
3060   { \l__enumext_joined_item_vii_int }
3061   >
3062   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3063   {
3064     \msg_warning:nnee { enumext } { item-joined-columns }
3065     { \int_use:N \l__enumext_joined_item_vii_int }
3066     {
3067       \int_eval:n
3068       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3069     }
3070     \int_set:Nn \l__enumext_joined_item_vii_int
3071     {
3072       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3073     }
3074   }
}

```

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

```

3075   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3076   {
3077     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3078     \int_decr:N \l__enumext_joined_item_aux_vii_int
3079     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3080     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3081     \dim_set:Nn \l__enumext_joined_width_vii_dim
3082     {
3083       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3084       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3085         + \l__enumext_columns_sep_vii_dim
3086         ) * \l__enumext_joined_item_aux_vii_int
3087     }
3088     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3089   }
3090   {
3091     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3092     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3093   }
3094 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n.)

\l__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`.

```

3095 \cs_new_protected:Nn \l__enumext_start_mini_vii:
3096 {
3097   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3098   {

```



```

3099     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3100     {
3101         \linewidth
3102         - \l__enumext_minipage_right_vii_dim
3103         - \l__enumext_minipage_hsep_vii_dim
3104     }
3105     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3106     \dim_gset_eq:NN
3107         \g__enumext_minipage_right_vii_dim
3108         \l__enumext_minipage_right_vii_dim
3109     \__enumext_mini_addvspace_vii:
3110     \nointerlineskip\noindent
3111     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3112 }
3113 }

```

(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”.

```

3114 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3115 {
3116     \bool_if:NT \l__enumext_minipage_active_vii_bool
3117     {
3118         \end{__enumext_mini_env*}
3119         \hfill
3120         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3121     }
3122 }

```

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”.

```

3123 \__enumext_after_env:nn {enumext*}
3124 {
3125     \bool_if:NT \g__enumext_minipage_active_vii_bool
3126     {
3127         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3128         \par\addvspace { \g__enumext_minipage_right_skip }
3129         \bool_if:NF \g__enumext_minipage_center_vii_bool
3130         {
3131             \centering
3132         }
3133         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3134         \end{__enumext_mini_env*}
3135         \par\addvspace{ \g__enumext_minipage_after_skip }
3136     }
3137     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3138     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3139     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3140     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3141     \bool_gset_false:N \g__enumext_starred_bool
3142 }

```

(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.

```

3143 \NewDocumentEnvironment{enumext*}{ o }
3144 {
3145     \__enumext_safe_exec_vii:
3146     \__enumext_parse_keys_vii:n {#1}
3147     \__enumext_before_list_vii:
3148     \__enumext_start_store_level_vii:
3149     \__enumext_start_list:nn { }
3150     {
3151         \__enumext_list_arg_two_vii:
3152         \__enumext_before_keys_exec_vii:
3153     }

```

```

3154     \__enumext_starred_columns_set_vii:
3155     \item[] \scan_stop:
3156     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3157     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3158   }
3159   {
3160     \__enumext_stop_item_tmp_vii:
3161     \__enumext_remove_extra_parsep_vii:
3162     \__enumext_stop_list:
3163     \__enumext_stop_store_level_vii:
3164     \__enumext_after_list_vii:
3165   }

```

(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`.

```

3166 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3167 {
3168   \__enumext_current_env_set_bool:
3169   \int_incr:N \l__enumext_level_h_int
3170   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3171   {
3172     \msg_error:nn { enumext } { nested }
3173   }
3174   \bool_set_true:N \l__enumext_starred_bool
3175   \bool_lazy_all:nT
3176   {
3177     { \bool_if_p:N \g__enumext_starred_bool }
3178     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3179     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
3180   }
3181   {
3182     \typeout{[[ON-FIRST-LEVEL-ENUMEXT*-NOT-NESTED]]}
3183     \bool_set_true:N \l__enumext_starred_level_one_bool
3184   }
3185 }

```

(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.

```

3186 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3187 {
3188   \tl_if_novalue:nF {#1}
3189   {
3190     \str_clear:N \l__enumext_series_str
3191     \keys_set:nn { enumext / enumext* } {#1}
3192     \__enumext_parse_series_name:n {#1}
3193     \bool_if:NT \l__enumext_store_active_bool
3194     {
3195       \__enumext_parse_store_keys_vii:n {#1}
3196     }
3197   }
3198 }

```

(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:`.

```

3199 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3200 {
3201   \bool_if:NF \l__enumext_store_columns_vii_bool
3202   {
3203     \regex_match:nnT { \b columns\b } {#1}
3204     {

```

```

3205         \int_set_eq:NN
3206         \l__enumext_store_columns_vii_int
3207         \l__enumext_columns_vii_int
3208         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3209         {
3210             columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3211         }
3212     }
3213 }
3214 \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3215 {
3216     \regex_match:nnT { \b columns-sep \b} {#1}
3217     {
3218         \dim_set_eq:NN
3219         \l__enumext_store_columns_sep_vii_dim
3220         \l__enumext_columns_sep_vii_dim
3221         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3222         {
3223             columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3224         }
3225     }
3226 }
3227 }

```

(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`.

```

3228 \cs_new_protected:Nn \__enumext_before_list_vii:
3229 {
3230     \__enumext_vspace_above_vii:
3231     \__enumext_check_ans_exec: % need by chek-ans
3232     \__enumext_before_args_exec_vii:
3233     \__enumext_start_mini_vii:
3234 }

```

(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.

```

3235 \cs_new_protected:Nn \__enumext_after_list_vii:
3236 {
3237     \__enumext_stop_mini_vii:
3238     \__enumext_after_stop_list_vii:
3239     \__enumext_vspace_below_vii:
3240     %\bool_gset_false:N \g__enumext_starred_bool
3241     \bool_set_false:N \l__enumext_starred_bool
3242     \__enumext_resume_counter_set:
3243 }

```

(End of definition for `__enumext_after_list_vii:.`)

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

`__enumext_stop_store_level_vii:`

```

3244 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3245 {
3246     \bool_if:NT \l__enumext_store_active_bool
3247     {
3248         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3249         {
3250             \__enumext_store_level_open_vii:
3251         }
3252     }
3253 }

```

```

3254 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3255 {
3256   \bool_if:NT \l__enumext_store_active_bool
3257   {
3258     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3259     {
3260       \__enumext_store_level_close_vii:
3261     }
3262   }
3263 }

```

(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.

```

3264 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3265 {
3266   \__enumext_stop_item_tmp_vii:
3267   \int_incr:N \l__enumext_item_column_pos_vii_int
3268   \int_gincr:N \g__enumext_item_count_all_vii_int
3269   \__enumext_item_peek_args_vii:
3270 }

```

(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).

```

3271 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3272 {
3273   \peek_meaning:NTF (
3274     { \__enumext_joined_item_vii:w }
3275     { \__enumext_joined_item_vii:w (1) }
3276   }

```

(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.

```

3277 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3278 {
3279   \__enumext_starred_joined_item_vii:n {#1}
3280   \peek_meaning_remove:NTF *
3281   { \__enumext_starred_item_vii:w }
3282   { \__enumext_standard_item_vii:w }
3283 }

```

(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].

```

3284 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3285 {
3286   \bool_set_false:N \l__enumext_item_starred_vii_bool
3287   \peek_meaning:NTF [
3288     {
3289       \bool_set_eq:NN

```

```

3290         \l__enumext_wrap_label_vii_bool
3291         \l__enumext_wrap_label_opt_vii_bool
3292     \__enumext_start_item_vii:w
3293 }
3294 {
3295     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3296     \legacy_if_set_true:n { @noitemarg }
3297     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3298 }
3299 }

```

(End of definition for __enumext_standard_item_vii:w.)

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

```

\__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
3300 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3301 {
3302     \bool_set_true:N \l__enumext_item_starred_vii_bool
3303     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3304     \peek_meaning:NTF [
3305         { \__enumext_starred_item_vii_aux_i:w }
3306         { \__enumext_starred_item_vii_aux_ii:w }
3307     ]
3308     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3309     {
3310         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3311         \__enumext_starred_item_vii_aux_ii:w
3312     }
3313     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3314     {
3315         \peek_meaning:NTF [
3316             { \__enumext_starred_item_vii_aux_iii:w }
3317             {
3318                 \dim_set_eq:NN
3319                 \l__enumext_item_symbol_sep_vii_dim
3320                 \l__enumext_labelsep_vii_dim
3321                 \legacy_if_set_true:n { @noitemarg }
3322                 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3323             }
3324         ]
3325     }
3326     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3327     {
3328         \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3329         \legacy_if_set_true:n { @noitemarg }
3330         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3331     }

```

(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.

```

3331 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3332 {
3333     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3334     \legacy_if:nT { @noitemarg }
3335     {
3336         \legacy_if_set_false:n { @noitemarg }
3337         \legacy_if:nT { @nmbrrlist }
3338         {
3339             \bool_if:NT \l__enumext_hyperref_bool
3340             {
3341                 \legacy_if_set_true:n { @hyper@item }
3342             }
3343             \refstepcounter{enumXvii}

```

```

3344         \bool_if:NT \l__enumext_check_ans_bool
3345         {
3346             \int_gincr:N \g__enumext_count_item_number_int
3347         }
3348     }
3349 }

```

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.

```

3350 \group_begin:
3351   \lrbox{ \l__enumext_item_text_vii_box }
3352   \bool_if:NF \l__enumext_footnotes_key_bool
3353   {
3354     \__enumext_renew_footnote:
3355   }
3356   \bool_if:NT \l__enumext_item_starred_vii_bool
3357   {
3358     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3359     {
3360       \tl_gset_eq:NN
3361         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3362     }
3363     \mode_leave_vertical:
3364     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3365     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3366     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3367     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3368   }
3369   \group_begin:
3370     \tl_use:N \l__enumext_label_font_style_vii_tl
3371     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3372     {
3373       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3374         { \__enumext_wrapper_label_vii:n {#1} }
3375     }
3376     {
3377       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3378     }
3379   \group_end:
3380   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3381   \tl_use:N \l__enumext_after_list_args_vii_tl
3382   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3383     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3384     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3385     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3386 }

```

(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.

```

3387 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3388 {
3389   \__enumext_endminipage:
3390   \endlrbox
3391   \group_end:
3392   \box_set_wd:Nn \l__enumext_item_text_vii_box
3393   {
3394     \l__enumext_joined_width_vii_dim
3395     + \l__enumext_labelwidth_vii_dim
3396     + \l__enumext_labelsep_vii_dim
3397   }
3398   \int_set:Nn \hbadness { 10000 }
3399   \box_use:N \l__enumext_item_text_vii_box
3400   \bool_if:NF \l__enumext_footnotes_key_bool

```

```

3401     {
3402         \__enumext_print_footnote:
3403     }
3404     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3405     {
3406         \par\noindent
3407         \int_zero:N \l__enumext_item_column_pos_vii_int
3408     }
3409     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3410 }

```

(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.

```

3411 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3412 {
3413     \int_compare:nNnT
3414     {
3415         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3416     }
3417     =
3418     { \c_zero_int }
3419     {
3420         \par
3421         \vspace{ -\l__enumext_itemsep_vii_skip }
3422         \int_gzero:N \g__enumext_item_count_all_vii_int
3423     }
3424 }

```

(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`.

```

3425 \__enumext_after_env:nn {enumext*}
3426 {
3427     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3428     {
3429         \bool_if:NT \g__enumext_check_ans_show_h_bool
3430         {
3431             \__enumext_check_ans_show:
3432         }
3433         \bool_gset_false:N \g__enumext_starred_bool
3434         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3435         \tl_gclear:N \g__enumext_store_name_tl
3436     }
3437 }

```

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.

```

3438 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3439 {
3440     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3441     {
3442         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3443         {
3444             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3445             / \l__enumext_columns_viii_int
3446         }
3447     }
3448     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3449     \dim_set:Nn \l__enumext_item_width_viii_dim
3450     {
3451         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3452         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3453         - \l__enumext_labelsep_viii_dim
3454     }
3455     \dim_zero_new:N \itemwidth
3456 }

```

(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`.

```

3457 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3458 {
3459   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3460   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3461   {
3462     \msg_warning:nnee { enumext } { item-joined }
3463     { \int_use:N \l__enumext_joined_item_viii_int }
3464     { \int_use:N \l__enumext_columns_viii_int }
3465     \int_set:Nn \l__enumext_joined_item_viii_int
3466     {
3467       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3468     }
3469   }
3470   \int_compare:nNnT
3471   { \l__enumext_joined_item_viii_int }
3472   >
3473   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3474   {
3475     \msg_warning:nnee { enumext } { item-joined-columns }
3476     { \int_use:N \l__enumext_joined_item_viii_int }
3477     {
3478       \int_eval:n
3479       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3480     }
3481     \int_set:Nn \l__enumext_joined_item_viii_int
3482     {
3483       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3484     }
3485   }
3486   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3487   {
3488     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3489     \int_decr:N \l__enumext_joined_item_aux_viii_int
3490     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3491     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3492     \dim_set:Nn \l__enumext_joined_width_viii_dim
3493     {
3494       \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3495       + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3496         + \l__enumext_columns_sep_viii_dim
3497       ) * \l__enumext_joined_item_aux_viii_int
3498     }
3499     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3500   }
3501   {
3502     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3503     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3504   }
3505 }

```

(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.

`__enumext_stop_mini_viii:`

```

3506 \cs_new_protected:Npn \__enumext_start_mini_viii:
3507 {
3508   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3509   {
3510     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3511     {
3512       \linewidth
3513       - \l__enumext_minipage_right_viii_dim
3514       - \l__enumext_minipage_hsep_viii_dim
3515     }
3516     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3517     \dim_gset_eq:NN
3518     \g__enumext_minipage_right_viii_dim
3519     \l__enumext_minipage_right_viii_dim

```



```

3520     \__enumext_mini_addvspace_viii:
3521     \nointerlineskip\noindent
3522     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3523   }
3524 }
3525 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3526 {
3527   \bool_if:NT \l__enumext_minipage_active_viii_bool
3528   {
3529     \end{__enumext_mini_env*}
3530     \hfill
3531     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3532   }
3533 }
3534 \__enumext_after_env:nn {keyans*}
3535 {
3536   \bool_if:NT \g__enumext_minipage_active_viii_bool
3537   {
3538     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3539     \par\addvspace { \g__enumext_minipage_right_skip }
3540     \bool_if:NF \g__enumext_minipage_center_viii_bool
3541     {
3542       \centering
3543     }
3544     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3545     \end{__enumext_mini_env*}
3546     \par\addvspace{ \g__enumext_minipage_after_skip }
3547   }
3548   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3549   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3550   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3551   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3552 }

```

(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.

```

3553 \NewDocumentEnvironment{keyans*}{ o }
3554 {
3555   \__enumext_safe_exec_viii:
3556   \__enumext_parse_keys_viii:n {#1}
3557   \__enumext_before_list_viii:
3558   \__enumext_start_list:nn { }
3559   {
3560     \__enumext_list_arg_two_viii:
3561     \__enumext_before_keys_exec_viii:
3562   }
3563   \__enumext_starred_columns_set_viii:
3564   \item[] \scan_stop:
3565   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3566   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3567 }
3568 {
3569   \__enumext_stop_item_tmp_viii:
3570   \__enumext_remove_extra_parsep_viii:
3571   \__enumext_keyans_check_ans:nn { item }{ keyans* }
3572   \__enumext_stop_list:
3573   \__enumext_after_list_viii:
3574 }

```

(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.

```

3575 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3576 {
3577   \int_incr:N \l__enumext_keyans_level_h_int
3578   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3579   {
3580     \msg_error:nn { enumext } { nested }

```

```

3581     }
3582     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3583     \bool_set_false:N \l__enumext_store_active_bool
3584     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
3585     {
3586         \msg_error:nn { enumext } { keyans-wrong-level }
3587     }
3588 }

```

(End of definition for __enumext_safe_exec_viii:.)

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

```

3589 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3590 {
3591     \tl_if_novalue:nF {#1}
3592     {
3593         \keys_set:nn { enumext / keyans* } {#1}
3594     }
3595 }

```

(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*.

```

3596 \cs_new_protected:Nn \__enumext_before_list_viii:
3597 {
3598     \__enumext_vspace_above_viii:
3599     \__enumext_before_args_exec_viii:
3600     \__enumext_start_mini_viii:
3601 }

```

(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.

```

3602 \cs_new_protected:Nn \__enumext_after_list_viii:
3603 {
3604     \__enumext_stop_mini_viii:
3605     \__enumext_after_stop_list_viii:
3606     \__enumext_vspace_below_viii:
3607 }

```

(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*.

```

3608 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3609 {
3610     \__enumext_stop_item_tmp_viii:
3611     \int_incr:N \l__enumext_item_column_pos_viii_int
3612     \int_gincr:N \g__enumext_item_count_all_viii_int
3613     \__enumext_item_peek_args_viii:
3614 }

```

(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).

```

3615 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3616 {
3617   \peek_meaning:NTF (
3618     { \__enumext_joined_item_viii:w }
3619     { \__enumext_joined_item_viii:w (1) }
3620   }

```

(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`.

```

3621 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3622 {
3623   \__enumext_starred_joined_item_viii:n {#1}
3624   \peek_meaning_remove:NTF *
3625   { \__enumext_starred_item_viii:w }
3626   { \__enumext_standard_item_viii:w }
3627 }

```

(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 [\l__enumext_label_viii_tl]`.

```

3628 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3629 {
3630   \bool_set_false:N \l__enumext_item_starred_viii_bool
3631   \peek_meaning:NTF [
3632     {
3633       \bool_set_eq:NN
3634       \l__enumext_wrap_label_viii_bool
3635       \l__enumext_wrap_label_opt_viii_bool
3636       \__enumext_start_item_viii:w
3637     }
3638     {
3639       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3640       \legacy_if_set_true:n { @noitemarg }
3641       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3642     }
3643   }

```

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

`__enumext_starred_item_viii: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>]`.

```

3644 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3645 {
3646   \bool_set_true:N \l__enumext_item_starred_viii_bool
3647   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3648   \peek_meaning:NTF [
3649     { \__enumext_starred_item_viii_aux_i:w }
3650     { \__enumext_starred_item_viii_aux_ii:w }
3651   }

```

The optional argument will be captured in the variables `\l__enumext_keyans_tmpa_tl` and `\l__enumext_keyans_tmppb_tl` which we will use later for the implementation of the `show-ans` and `show-pos` keys together with the stored in *(sequence)* and *(prop list)*.

```

3652 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3653 {

```

```

3654 \tl_clear:N \l__enumext_store_keyans_label_tl
3655 \tl_if_novalue:nF { #1 }
3656 {
3657   \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3658   {
3659     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
3660     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3661   }
3662   \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3663 }
3664 \__enumext_starred_item_viii_aux_ii:w
3665 }
3666 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3667 {
3668   \legacy_if_set_true:n { @noitemarg }
3669   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3670 }

```

(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.

```

3671 \cs_new_protected:Nn \__enumext_starred_item_exec:
3672 {
3673   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3674   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3675   \__enumext_keyans_store_ref:
3676   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3677   \__enumext_keyans_addto_seq_link:
3678   \bool_if:NT \l__enumext_show_answer_bool
3679   {
3680     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3681   }
3682   \bool_if:NT \l__enumext_show_position_bool
3683   {
3684     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3685     {
3686       \group_begin:
3687       \exp_not:N \normalfont
3688       \exp_not:N \footnotesize [ \int_eval:n
3689       {
3690         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3691       }
3692       ]
3693       \group_end:
3694     }
3695     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3696   }
3697 }

```

(End of definition for `__enumext_starred_item_exec:`.)

Real definition of `\item` in `keyans*`

`__enumext_start_item_viii:w`

The implementation at this point is very similar to that of the `enumext*` environment.

```

3698 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3699 {
3700   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3701   \legacy_if:nT { @noitemarg }
3702   {
3703     \legacy_if_set_false:n { @noitemarg }
3704     \legacy_if:nT { @nmbrrlist }
3705     {
3706       \bool_if:NT \l__enumext_hyperref_bool
3707       {
3708         \legacy_if_set_true:n { @hyper@item }
3709       }
3710       \refstepcounter{enumXviii}
3711     }

```

```
3712 }
```

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

```
3713 \group_begin:
3714 \lrbox{ \l__enumext_item_text_viii_box }
3715 \bool_if:NF \l__enumext_footnotes_key_bool
3716 {
3717   \__enumext_renew_footnote:
3718 }
3719 \bool_if:NT \l__enumext_item_starred_viii_bool
3720 {
3721   \__enumext_starred_item_exec:
3722 }
3723 \group_begin:
3724 \tl_use:N \l__enumext_label_font_style_viii_tl
3725 \bool_if:NTF \l__enumext_wrap_label_viii_bool
3726 {
3727   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3728   { \__enumext_wrapper_label_viii:n {#1} }
3729 }
3730 {
3731   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3732 }
3733 \group_end:
3734 \skip_horizontal:N \l__enumext_labelsep_viii_dim
3735 \tl_use:N \l__enumext_after_list_args_viii_tl
3736 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3737 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3738 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3739 \bool_if:NT \l__enumext_item_starred_viii_bool
3740 {
3741   \tl_use:N \l__enumext_fake_item_indent_viii_tl
3742   \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3743 }
3744 {
3745   \tl_use:N \l__enumext_fake_item_indent_viii_tl
3746 }
3747 }
```

(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.

```
3748 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3749 {
3750   \__enumext_endminipage:
3751   \endlrbox
3752   \group_end:
3753   \box_set_wd:Nn \l__enumext_item_text_viii_box
3754   {
3755     \l__enumext_joined_width_viii_dim
3756     + \l__enumext_labelwidth_viii_dim
3757     + \l__enumext_labelsep_viii_dim
3758   }
3759   \int_set:Nn \hbadness { 10000 }
3760   \box_use:N \l__enumext_item_text_viii_box
3761   \bool_if:NF \l__enumext_footnotes_key_bool
3762   {
3763     \__enumext_print_footnote:
3764   }
3765   \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3766   {
3767     \par\noindent
3768     \int_zero:N \l__enumext_item_column_pos_viii_int
3769   }
3770   { \hspace{ \l__enumext_columns_sep_viii_dim } }
3771 }
```

(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.

```

3772 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3773 {
3774   \int_compare:nNnT
3775   {
3776     \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3777   }
3778   =
3779   { \c_zero_int }
3780   {
3781     \par
3782     \vspace{ -\l__enumext_itemsep_viii_skip }
3783     \int_gzero:N \g__enumext_item_count_all_viii_int
3784   }
3785 }

```

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

```

3786 \NewDocumentCommand \getkeyans { m }
3787 {
3788   \exp_args:Ne \__enumext_getkeyans_aux:n
3789   { \tl_to_str:e { \text_expand:n {#1} } }
3790 }

```

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

`__enumext_getkeyans_aux:n`

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

```

3791 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3792 {
3793   \str_if_in:nnTF {#1} { : }
3794   {
3795     \use:e
3796     {
3797       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3798       { {##1} {##2} }
3799     }
3800     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3801   }
3802   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3803 }

```

(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⟩`.

```

3804 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3805 {
3806   \prop_if_exist:cF { g__enumext_#1_prop }
3807   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3808   \group_begin:
3809     \prop_item:cn { g__enumext_#1_prop }{#2}
3810   \group_end:
3811 }

```

(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`.

```

3812 \keys_define:nn { keyanskey / print }
3813 {
3814   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3815                 {
3816                   \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3817                 },
3818   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3819   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3820                 {
3821                   \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3822                 },
3823   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3824   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3825                 {
3826                   \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3827                 },
3828   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3829   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3830                 {
3831                   \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3832                 },
3833   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3834   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3835                 {
3836                   \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3837                 },
3838   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3839 }

```

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

```

3840 \NewDocumentCommand \printkeyans { s O{} m }
3841 {
3842   \group_begin:
3843     \tl_use:N \l__enumext_print_keyans_i_tl
3844     \tl_use:N \l__enumext_print_keyans_ii_tl
3845     \tl_use:N \l__enumext_print_keyans_iii_tl
3846     \tl_use:N \l__enumext_print_keyans_iv_tl
3847     \tl_use:N \l__enumext_print_keyans_vii_tl
3848     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3849   \group_end:
3850 }

```

(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

```

3851 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3852 {
3853   \seq_if_exist:cTF { g__enumext_#3_seq }
3854   {
3855     \seq_if_empty:cF { g__enumext_#3_seq }
3856     {
3857       %%\seq_show:c { g__enumext_#3_seq }
3858       \bool_if:nTF {#1}
3859       {
3860         \begin{enumext*}[#2]
3861         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3862         \end{enumext*}
3863       }
3864     }

```

```

3865         \begin{enumext}[#2]
3866         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3867         \end{enumext}
3868     }
3869 }
3870 }
3871 {
3872     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3873 }
3874 }

```

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

10.40 The command `\setenumext`

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

```

3875 \keys_define:nn { enumext / meta-families }
3876 {
3877     level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3878     level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3879     level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3880     level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3881     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3882     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3883     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3884     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3885     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3886     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3887     print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3888     print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3889     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3890 }

```

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

```

3891 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3892 {
3893     level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3894     keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3895 }

```

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

```

3896 \NewDocumentCommand \setenumext { o +m }
3897 {
3898     \tl_if_novalue:nTF {#1}
3899     {
3900         \seq_map_inline:Nn \c__enumext_all_families_seq
3901     }
3902     {
3903         \seq_clear:N \l__enumext_setkey_tmpa_seq
3904         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3905         \int_set:Nn \l__enumext_setkey_tmpa_int
3906         {
3907             \seq_count:N \l__enumext_setkey_tmpb_seq
3908         }
3909         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3910         {
3911             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3912             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3913             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3914             {
3915                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3916             }
3917         }
3918         {
3919             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3920         }
3921         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3922         { \seq_map_inline:Nn \c__enumext_all_families_seq }
3923         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3924     }
3925     {
3926         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }

```



```

3927     }
3928 }

```

(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.

```

3929 \cs_new_protected:Npn \__enumext_set_parse:n #1
3930 {
3931   \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3932   \int_step_inline:nnn { 0 } { 4 } {%<- max level
3933     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3934   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3935     {
3936       \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
3937         { \tl_trim_spaces:n {#1} }
3938     }
3939     { \__enumext_set_error:nn {#1} { } }
3940   }
3941   \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3942   { \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.

```

3943 \msg_new:nnn { enumext } { package-load }
3944 {
3945   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3946 }
3947 \msg_new:nnn { enumext } { package-not-load }
3948 {
3949   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3950 }
3951 \msg_new:nnn { enumext } { package-load-foot }
3952 {
3953   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3954 }

```

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

```

3955 \msg_new:nnn { enumext } { counters }
3956 {
3957   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
3958   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3959 }

```

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

```

3960 \msg_new:nnn { enumext } { invalid-key }
3961 {
3962   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3963 }
3964 \msg_new:nnn { enumext } { unknown-key-family }
3965 {
3966   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3967 }

```

Messages used in length calculation.

```

3968 \msg_new:nnn { enumext } { width-negative }
3969 {
3970   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3971   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
3972 }
3973 \msg_new:nnn { enumext } { width-zero }
3974 {
3975   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3976   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
3977 }

```

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

```

3978 \msg_new:nnn { enumext } { list-lengths }
3979 {
3980   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
3981   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3982   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3983   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3984   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3985   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3986   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3987   \__enumext_show_length:nnn { skip } { topsep } {#1}
3988   \__enumext_show_length:nnn { skip } { parsep } {#1}
3989   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3990   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3991   *****
3992 }

```

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

```

3993 \msg_new:nnn { enumext } { list-lengths-not-nested }
3994 {
3995   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
3996   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3997   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3998   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3999   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4000   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4001   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4002   \__enumext_show_length:nnn { skip } { topsep } {#1}
4003   \__enumext_show_length:nnn { skip } { parsep } {#1}
4004   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4005   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4006   *****
4007 }

```

Messages used by `save-ans` key.

```

4008 \msg_new:nnn { enumext } { save-ans-empty }
4009 {
4010   The ~ 'save-ans' ~ key ~ cannot ~ be ~ empty~ in ~ '#1'. ~ \msg_line_context:.
4011 }
4012 \msg_new:nnn { enumext } { save-ans-nested }
4013 {
4014   The ~ 'save-ans' ~ key ~ cannot ~ be ~ used ~ in ~ nested ~ '#1'. ~ \msg_line_context:.
4015 }

```

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

```

4016 \msg_new:nnn { enumext } { items-same-answer }
4017 {
4018   *****~Checking~answers~on~'#1'~OK~*****\\
4019   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4020   *****
4021   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4022 }
4023 \msg_new:nnn { enumext } { item-different-answer }
4024 {
4025   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4026   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4027 }

```

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

```

4028 \msg_new:nnn { enumext } { missing-starred }
4029 {
4030   Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4031 }

```

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

```

4032 \msg_new:nnn { enumext } { list-too-deep }
4033 {
4034   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4035   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4036 }

```

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

```

4037 \msg_new:nnn { enumext } { anskey-wrong-place }
4038 {
4039   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4040   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4041 }
4042 \msg_new:nnn { enumext } { anspic-wrong-place }
4043 {
4044   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4045   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4046 }
4047 \msg_new:nnn { enumext } { command-wrong-place }
4048 {
4049   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4050   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4051 }

```

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

```

4052 \msg_new:nnn { enumext } { keyans-nested }
4053 {
4054   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4055 }
4056 \msg_new:nnn { enumext } { keyans-wrong-level }
4057 {
4058   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4059   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4060 }
4061 \msg_new:nnn { enumext } { wrong-place }
4062 {
4063   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4064   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4065 }
4066 \msg_new:nnn { enumext } { keyanspic-nested }
4067 {
4068   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4069 }
4070 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4071 {
4072   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4073   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4074 }

```

Messages used by `\getkeyans` command.

```

4075 \msg_new:nnn { enumext } { undefined-storage-anskey }
4076 {
4077   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4078 }

```

Messages used by `\miniright` command.

```

4079 \msg_new:nnn { enumext } { missing-miniright }
4080 {
4081   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
4082   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4083 }
4084 \msg_new:nnn { enumext } { wrong-miniright-place }
4085 {
4086   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4087   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4088 }
4089 \msg_new:nnn { enumext } { wrong-miniright-use }
4090 {
4091   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4092   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4093 }

```

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

```

4094 \msg_new:nnn { enumext } { nested }
4095 {
4096   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4097 }
4098 \msg_new:nnn { enumext } { item-joined }
4099 {
4100   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.

```

```
4101     }  
4102 \msg_new:nnn { enumext } { item-joined-columns }  
4103 {  
4104     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4105 }
```

10.42 Finish package

Finish package implementation.

```
4106 \file_input_stop:  
4107 </package>
```

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

```
enumXiii ..... 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 339, 355, 554, 570, 1785,
 1794, 1799, 1879, 2499, 3010
 \cs_if_exist:NTF 309
 \cs_new:Nn 191
 \cs_new:Npn 201, 1427, 1436, 1445
 \cs_new_eq:NN 245, 246, 247, 251, 252, 284, 285, 288,
 289
 \cs_new_protected:Nn . 215, 256, 420, 440, 472, 735,
 739, 743, 747, 751, 755, 759, 763, 767, 771, 775, 779,
 783, 787, 791, 795, 831, 843, 867, 884, 895, 919, 994,
 1018, 1035, 1097, 1114, 1136, 1171, 1177, 1252, 1266,
 1280, 1291, 1302, 1313, 1324, 1335, 1378, 1388, 1398,
 1562, 1581, 1650, 1686, 1693, 1800, 1824, 1831, 1859,
 1866, 1983, 2115, 2130, 2158, 2188, 2233, 2245, 2252,
 2304, 2308, 2327, 2378, 2412, 2428, 2438, 2454, 2591,
 2658, 2687, 2694, 2717, 2747, 2766, 2816, 2839, 2857,
 2882, 2893, 2930, 2973, 2986, 3006, 3011, 3027, 3095,
 3114, 3166, 3228, 3235, 3244, 3254, 3271, 3411, 3438,
 3506, 3525, 3575, 3596, 3602, 3615, 3671, 3772
 \cs_new_protected:Npn 183, 187, 292, 307, 324, 334,
 340, 428, 447, 541, 555, 1199, 1218, 1356, 1367, 1456,
 1472, 1487, 1511, 1786, 1795, 1915, 2060, 2072, 2094,
 2168, 2210, 2218, 2337, 2355, 2389, 2401, 2468, 2502,
 2541, 2609, 2629, 2835, 2981, 3046, 3186, 3199, 3277,
 3284, 3300, 3308, 3313, 3325, 3457, 3589, 3621, 3628,
 3644, 3652, 3666, 3791, 3804, 3851, 3929, 3941
 \cs_new_protected_nopar:Nn . . . 3264, 3387, 3608,
 3748
 \cs_new_protected_nopar:Npn 3331, 3698
 \cs_set:Nn 2065
 \cs_set:Npn 1993, 2031, 3797
 \cs_set_eq:NN . . 3156, 3157, 3333, 3565, 3566, 3700
 \cs_set_protected:Nn 207, 659, 675, 687, 699
 \cs_set_protected:Npn 32, 47, 55, 67, 73, 101, 139,
 151, 158, 209, 356, 378, 407, 488, 508, 571, 591, 635,
 654, 711, 720, 799, 816, 1235, 1415, 1632, 1706, 1742,
 1760, 1985, 2119, 2288, 2500, 2539
 \cs_to_str:N 326, 349

D

\d 197
 \DeclareDocumentEnvironment 912
 dim commands:

\dim_abs:n 2473, 2478
 \dim_add:Nn 2991
 \dim_compare:nNnTF . 661, 677, 689, 701, 1201, 1220,
 2470, 2475, 2481, 2487, 2489, 2491, 2699, 2722, 2843,
 2861, 2983, 3029, 3097, 3440, 3508
 \dim_compare:nTF 1956
 \dim_gset_eq:NN 3106, 3517
 \dim_gzero:N 3140, 3551
 \dim_new:N 43, 50, 51, 52, 69, 95, 108, 118, 167, 168, 174
 \dim_set:Nn . . 337, 649, 1772, 2369, 2473, 2478, 2480,
 2483, 2484, 2488, 2490, 2493, 2494, 2496, 2702, 2725,
 2845, 2863, 3013, 3031, 3038, 3081, 3099, 3327, 3442,
 3449, 3492, 3510
 \dim_set_eq:NN 495, 515, 531, 535, 2364, 2511, 2550,
 2648, 2733, 2871, 3088, 3091, 3092, 3218, 3318, 3499,
 3502, 3503

\dim_use:N 662, 670, 1202, 1208, 1869, 1872, 1877, 2433,
 2435, 2700, 2705, 2706, 2713, 2723, 2727, 2728, 2730
 \dim_zero:N 2737, 2873, 2992, 2993, 2994
 \dim_zero_new:N 3044, 3455
 \c_zero_dim 664, 678, 690, 702, 1202, 1220, 1958, 2470,
 2475, 2481, 2488, 2700, 2723, 2843, 2861, 3029, 3097,
 3440, 3508

E

\end . . 1205, 1223, 1828, 1863, 2752, 2776, 2886, 2903, 3118,
 3134, 3529, 3545, 3862, 3867
 \endlist 28
 \endlist 246
 \endlrbox 3390, 3751
 \endminipage 28
 \endminipage 252
 enumext 5, 2572
 enumext internal commands:

\g__enumext_ __enumext_store_name_tl
 _prop 68
 __enumext_add_pre_parsep: . . . 41, 841, 843, 843
 __enumext_after_args_exec: . 39, 735, 747, 2584
 __enumext_after_args_exec_v: . 39, 40, 751, 763,
 2809
 __enumext_after_args_exec_vii: . . . 767, 791
 __enumext_after_args_exec_viii: 795
 __enumext_after_env:n 80
 __enumext_after_env:nn . . 80, 94, 187, 187, 2785,
 3123, 3425, 3534
 __enumext_after_hyperref: . . . 29, 254, 256, 256
 __enumext_after_list: 80, 90, 97, 2589, 2766, 2766
 \l__enumext_after_list_args_v_tl 765
 \l__enumext_after_list_args_vii_tl 793, 3381
 \l__enumext_after_list_args_viii_tl 797, 3735
 __enumext_after_list_v: . . 83, 2814, 2893, 2893
 __enumext_after_list_vii: . . . 3164, 3235, 3235
 __enumext_after_list_viii: . . 3573, 3602, 3602
 __enumext_after_star_env:nn 88
 __enumext_after_stop_list: . . . 39, 40, 735, 743,
 2780
 __enumext_after_stop_list_v: 39, 751, 759, 2908
 \l__enumext_after_stop_list_v_tl 761
 __enumext_after_stop_list_vii: 767, 783, 3238
 \l__enumext_after_stop_list_vii_tl . . . 785
 __enumext_after_stop_list_viii: . 787, 3605
 \l__enumext_after_stop_list_viii_tl . . . 789
 \l__enumext_align_label_vii_str . . 3373, 3377
 \l__enumext_align_label_viii_str . 3727, 3731
 \l__enumext_align_label_X_str 158
 \c__enumext_all_envs_clist . . 178, 377, 590, 653,
 719, 734, 815, 1251
 \c__enumext_all_families_seq . . 103, 3891, 3900,
 3922
 __enumext_anskey_wrapper:n 1710, 2070
 __enumext_at_begin_document:n . . 28, 183, 183,
 243, 249
 __enumext_before_args_exec: 38, 735, 735, 2697
 __enumext_before_args_exec_v: . . 39, 751, 751,
 2842
 __enumext_before_args_exec_vii: . . 767, 767,
 3232
 __enumext_before_args_exec_viii: 771, 3599
 __enumext_before_keys_exec: 39, 735, 739, 2582
 __enumext_before_keys_exec_v: . . 39, 751, 755,
 2807

```

__enumext_before_keys_exec_vii . . . . . 767
__enumext_before_keys_exec_viii: 40, 775, 3152
__enumext_before_keys_exec_viii: . . 40, 779,
    3561
__enumext_before_list: . . . 78, 2576, 2694, 2694
__enumext_before_list_v: . 81, 2802, 2839, 2839
__enumext_before_list_vii: 90, 3147, 3228, 3228
__enumext_before_list_viii: . . 97, 3557, 3596,
    3596
\l__enumext_before_no_starred_key_v_tl 757
\l__enumext_before_no_starred_key_vii-
    tl . . . . . 777
\l__enumext_before_no_starred_key_viii-
    tl . . . . . 781
\l__enumext_before_starred_key_v_tl . . . 753
\l__enumext_before_starred_key_vii_tl . 769
\l__enumext_before_starred_key_viii_tl 773
__enumext_calc_hspace:NNNNNNN 73, 2468, 2468,
    2499, 2504, 2543
\l__enumext_check_ans_bool . . . 70, 71, 122, 1636,
    1640, 1688, 1907, 2205, 2341, 2371, 2759, 3344
__enumext_check_ans_exec: . . 58, 78, 1686, 1686,
    2698, 3231
\g__enumext_check_ans_item_tl . . 67, 122, 2204,
    2212, 2216
__enumext_check_ans_set: . 57, 1650, 1650, 1690
__enumext_check_ans_show: 58, 1693, 1693, 2791,
    3431
\g__enumext_check_ans_show_bool 80, 122, 2762,
    2789, 2794
\g__enumext_check_ans_show_h_bool 122, 3429,
    3434
\l__enumext_columns_sep_v_dim 2861, 2863, 2871
\l__enumext_columns_sep_vii_dim . . 3029, 3031,
    3040, 3085, 3220, 3409
\l__enumext_columns_sep_viii_dim . 3440, 3442,
    3451, 3496, 3770
\l__enumext_columns_v_int 1040, 2859, 2867, 2879,
    2884
\l__enumext_columns_vii_int . . 3034, 3037, 3041,
    3049, 3053, 3056, 3062, 3068, 3072, 3207, 3404, 3415
\l__enumext_columns_viii_int . 3445, 3448, 3452,
    3460, 3464, 3467, 3473, 3479, 3483, 3765, 3776
\g__enumext_count_item_anskey_int . . 67, 122,
    1696, 1704, 1909, 2207
\g__enumext_count_item_number_int 122, 1661,
    1666, 1669, 1672, 1680, 1696, 1703, 2343, 2373, 3346
\g__enumext_count_item_with_ans_int . . . 62
\l__enumext_counter_i_tl . . . . . 32, 316
\l__enumext_counter_ii_tl . . . . . 32, 317
\l__enumext_counter_iii_tl . . . . . 32, 318
\l__enumext_counter_iv_tl . . . . . 32, 319
\l__enumext_counter_style_for_ref_vii-
    tl . . . . . 455, 465, 476, 478
\l__enumext_counter_style_for_ref_viii-
    tl . . . . . 482, 484
\l__enumext_counter_style_for_ref_X_tl 147
\c__enumext_counter_style_tl . . . 32, 147, 422
\g__enumext_counter_styles_tl . 23, 31, 43, 327,
    345
\l__enumext_counter_v_tl . . . . . 32, 320
\l__enumext_counter_vi_tl . . . . . 32, 321
\l__enumext_counter_vii_tl . . . . . 32, 322, 452
\l__enumext_counter_viii_tl . . . . . 32, 323, 462
__enumext_current_env_set_bool: 27, 215, 215,
    2593, 3168
\l__enumext_current_widest_dim 23, 43, 351, 496,
    516, 532, 536
__enumext_default_item:n . . . 2337, 2337, 2386
__enumext_define_counters:Nn 23, 307, 307, 316,
    317, 318, 319, 320, 321, 322, 323
__enumext_endminipage: . 28, 249, 252, 918, 3023,
    3389, 3750
__enumext_fake_item: . . . . . 659, 659, 2530
\l__enumext_fake_item_indent_v_dim 678, 683
\l__enumext_fake_item_indent_v_tl 680, 2394,
    2398, 2406
\l__enumext_fake_item_indent_vii_dim 690, 695
\l__enumext_fake_item_indent_vii_tl 692, 3385
\l__enumext_fake_item_indent_viii_dim . 702,
    707, 3742
\l__enumext_fake_item_indent_viii_tl . . 704,
    3741, 3745
\l__enumext_fake_item_indent_X_tl . . . . . 73
__enumext_fake_item_vii: . . . 659, 687, 2560
__enumext_fake_item_viii: . . . 659, 699, 2565
__enumext_filter_series:n 53, 1427, 1427, 1465,
    1478, 1484
__enumext_filter_series_key:n 53, 1427, 1432,
    1436
__enumext_filter_series_pair:nn . . 53, 1427,
    1433, 1445
\g__enumext_footnote_arg_seq . 144, 2310, 2323,
    2333
\g__enumext_footnote_int . 144, 2317, 2320, 2322,
    2324
\g__enumext_footnote_int_seq . 144, 2311, 2324,
    2329, 2332
__enumext_footnotes_key_bool . . . . . 29
\l__enumext_footnotes_key_bool 25, 29, 93, 134,
    265, 270, 279, 3352, 3400, 3715, 3761
__enumext_footnotetext:nn . . . 2304, 2304, 2334
__enumext_getkeyans:nn . . 101, 3800, 3804, 3804
__enumext_getkeyans_aux:n 101, 3788, 3791, 3791
\l__enumext_hyperref_bool 25, 29, 134, 261, 282,
    299, 1973, 2192, 3339, 3706
__enumext_hypertarget:nn 29, 256, 284, 288, 304
__enumext_if_is_int:n . . . . . 195
__enumext_if_is_int:nTF . . . . . 195, 543, 557
\l__enumext_item_column_pos_vii_int 91, 3056,
    3062, 3068, 3072, 3079, 3267, 3404, 3407
\l__enumext_item_column_pos_viii_int . . . 97,
    3467, 3473, 3479, 3483, 3490, 3611, 3765, 3768
\l__enumext_item_column_pos_X_int . . . . . 158
\g__enumext_item_count_all_vii_int 91, 3080,
    3268, 3415, 3422
\g__enumext_item_count_all_viii_int 97, 3491,
    3612, 3776, 3783
\g__enumext_item_count_all_X_int . . . . . 158
__enumext_item_peek_args_vii: 91, 3269, 3271,
    3271
__enumext_item_peek_args_viii: . 97, 98, 3613,
    3615, 3615
__enumext_item_starred: . . 73, 2428, 2428, 2446
\l__enumext_item_starred_vii_bool 3286, 3302,
    3356
\l__enumext_item_starred_viii_bool 3630, 3646,
    3719, 3739

```


<code>\l__enumext_item_starred_X_bool</code>	158
<code>__enumext_item_std:w</code> 28 , 70–72 , 85 , 243 , 247 , 2346 , 2352 , 2376 , 2394 , 2398 , 2406 , 3004	
<code>\g__enumext_item_symbol_aux_vii_tl</code> 3310 , 3358 , 3361 , 3365 , 3367	
<code>\g__enumext_item_symbol_aux_X_tl</code>	158
<code>\l__enumext_item_symbol_sep_vii_dim</code>	3319 , 3327 , 3364 , 3366
<code>\g__enumext_item_symbol_tl</code> 23 , 71 , 40 , 2361 , 2434 , 2451	
<code>\l__enumext_item_symbol_vii_tl</code>	3361
<code>\l__enumext_item_text_vii_box</code> 3351 , 3392 , 3399	
<code>\l__enumext_item_text_viii_box</code> 3714 , 3753 , 3760	
<code>\l__enumext_item_text_X_box</code>	158
<code>\l__enumext_item_width_vii_dim</code>	3038 , 3083 , 3091 , 3092
<code>\l__enumext_item_width_viii_dim</code>	3449 , 3494 , 3502 , 3503
<code>\l__enumext_item_width_X_dim</code>	158
<code>\l__enumext_itemindent_X_dim</code>	47
<code>\l__enumext_itemsep_vii_skip</code>	3421
<code>\l__enumext_itemsep_viii_skip</code>	3782
<code>\l__enumext_joined_item_aux_vii_int</code>	3077 , 3078 , 3079 , 3080 , 3086
<code>\l__enumext_joined_item_aux_viii_int</code>	3488 , 3489 , 3490 , 3491 , 3497
<code>\l__enumext_joined_item_aux_X_int</code>	158
<code>__enumext_joined_item_vii:w</code>	91 , 3274 , 3275 , 3277 , 3277
<code>\l__enumext_joined_item_vii_int</code>	3048 , 3049 , 3052 , 3054 , 3060 , 3065 , 3070 , 3075 , 3077 , 3083
<code>__enumext_joined_item_viii:w</code>	98 , 3618 , 3619 , 3621 , 3621
<code>\l__enumext_joined_item_viii_int</code>	3459 , 3460 , 3463 , 3465 , 3471 , 3476 , 3481 , 3486 , 3488 , 3494
<code>\l__enumext_joined_item_X_int</code>	158
<code>\l__enumext_joined_width_vii_dim</code>	3081 , 3088 , 3091 , 3382 , 3394
<code>\l__enumext_joined_width_viii_dim</code> 3492 , 3499 , 3502 , 3736 , 3755	
<code>\l__enumext_joined_width_X_dim</code>	158
<code>__enumext_keyans_addto_prop:n</code> 65 , 2094 , 2094 , 2408 , 2936	
<code>__enumext_keyans_addto_seq:n</code>	67 , 2168 , 2168 , 2410 , 2938
<code>__enumext_keyans_addto_seq_link:</code> 2168 , 2186 , 2188 , 3677	
<code>__enumext_keyans_anspic_code:nnn</code>	84 , 2927 , 2930 , 2930
<code>__enumext_keyans_check_ans:nn</code>	67 , 68 , 2210 , 2210 , 2812 , 2968 , 3571
<code>__enumext_keyans_default_item:n</code>	72 , 2389 , 2389 , 2424
<code>\l__enumext_keyans_env_bool</code> 20 , 2663 , 2676 , 2823 , 2907	
<code>__enumext_keyans_fake_item:</code>	659 , 675 , 2520
<code>\l__enumext_keyans_item_opt_tl</code> 85 , 2222 , 2235 , 2241 , 3662	
<code>\l__enumext_keyans_level_h_int</code> 20 , 2146 , 3577 , 3578	
<code>\l__enumext_keyans_level_int</code>	20 , 1193 , 1896 , 2141 , 2822 , 2826 , 2921
<code>__enumext_keyans_make_label:</code> 31 , 73 , 2454 , 2454 , 2519	
<code>__enumext_keyans_mini_addvspace:</code> 46 , 82 , 1097 , 1097 , 2851	
<code>__enumext_keyans_mini_right_cmd:n</code> 49 , 1195 , 1218 , 1218	
<code>__enumext_keyans_mini_set_vskip:</code>	45 , 1035 , 1035 , 1099
<code>__enumext_keyans_multi_addvspace:</code>	82 , 884 , 895 , 2876
<code>__enumext_keyans_multi_set_vskip:</code>	42 , 884 , 884 , 897
<code>__enumext_keyans_multicols_start:</code> 82 , 2855 , 2857 , 2857	
<code>__enumext_keyans_multicols_stop:</code>	82 , 1222 , 2882 , 2882 , 2906
<code>__enumext_keyans_parse_keys:n</code> 2801 , 2835 , 2835	
<code>\l__enumext_keyans_pic_above_int</code>	117 , 3014 , 3015 , 3017
<code>\l__enumext_keyans_pic_above_skip</code>	85 , 117 , 2959 , 2998
<code>__enumext_keyans_pic_arg_two:</code> 85 , 2957 , 2986 , 2986	
<code>\l__enumext_keyans_pic_below_int</code>	117 , 3014 , 3015 , 3018
<code>\l__enumext_keyans_pic_body_seq</code>	84–86 , 117 , 2925 , 2964 , 3022
<code>__enumext_keyans_pic_do:n</code> 86 , 2964 , 2966 , 3006 , 3006 , 3010	
<code>\l__enumext_keyans_pic_level_int</code>	20 , 1185 , 1900 , 2097 , 2136 , 2171 , 2254 , 2975 , 2976
<code>__enumext_keyans_pic_row:n</code> 86 , 3008 , 3011 , 3011	
<code>__enumext_keyans_pic_safe_exec:</code>	85 , 2953 , 2973 , 2973
<code>__enumext_keyans_pic_skip_abs:N</code>	85 , 2981 , 2981 , 2997
<code>\l__enumext_keyans_pic_width_dim</code>	117 , 3013 , 3020
<code>__enumext_keyans_redefine_item:</code>	72 , 2412 , 2412 , 2518
<code>__enumext_keyans_safe_exec:</code>	2800 , 2816 , 2816
<code>__enumext_keyans_show_ans:</code>	2218 , 2226 , 2245
<code>__enumext_keyans_show_item_opt:</code>	2218 , 2233 , 2406 , 2949 , 3742
<code>__enumext_keyans_show_left:n</code>	72 , 2218 , 2218 , 2404 , 2944
<code>__enumext_keyans_show_pos:</code>	2218 , 2230 , 2252
<code>__enumext_keyans_starred_item:n</code>	72 , 2401 , 2401 , 2420
<code>__enumext_keyans_store_ref:</code>	66 , 2115 , 2115 , 2409 , 2937 , 3675
<code>__enumext_keyans_store_ref_aux_i:</code> 66 , 2115 , 2127 , 2130	
<code>__enumext_keyans_store_ref_aux_ii:</code> 67 , 2115 , 2156 , 2158	
<code>\l__enumext_keyans_tmpa_dim</code>	85
<code>\l__enumext_keyans_tmpa_tl</code> 24 , 98 , 85 , 2403 , 2407	
<code>\l__enumext_keyans_tmpb_tl</code>	98 , 85
<code>__enumext_keyans_wrapper_opt:n</code>	1713 , 2241
<code>\l__enumext_label_copy_i_tl</code>	2027 , 2134 , 2139 , 2144 , 2149
<code>\l__enumext_label_copy_v_tl</code>	2144
<code>\l__enumext_label_copy_vi_tl</code>	2139
<code>\l__enumext_label_copy_vii_tl</code> 2002 , 2013 , 2044 , 2134	
<code>\l__enumext_label_copy_viii_tl</code>	2149
<code>\l__enumext_label_copy_X_tl</code>	136


```

\l__enumext_label_fill_left_v_tl . . . . . 2458
\l__enumext_label_fill_left_X_tl . . . . . 73
\l__enumext_label_fill_right_v_tl . . . . . 2465
\l__enumext_label_fill_right_X_tl . . . . . 73
\l__enumext_label_font_style_v_tl 2459, 2948
\l__enumext_label_font_style_vii_tl . . . 3370
\l__enumext_label_font_style_viii_tl . . 3724
\l__enumext_label_i_tl . . . . . 488
\l__enumext_label_ii_tl . . . . . 488
\l__enumext_label_iii_tl . . . . . 488
\l__enumext_label_iv_tl . . . . . 488
\__enumext_label_style:Nnn 23, 31, 340, 340, 355,
493, 513, 529, 533
\l__enumext_label_v_tl . . 65, 67, 526, 2102, 2176,
2247, 2281, 2403, 2407, 2804, 2943, 2945
\l__enumext_label_vi_tl . 65, 67, 526, 2099, 2173,
2943, 2945, 2949
\l__enumext_label_vii_tl . 508, 3297, 3322, 3329
\l__enumext_label_viii_tl 508, 3641, 3669, 3673
\l__enumext_label_width_by_box . . 43, 336, 337
\__enumext_label_width_by_box:Nn 31, 334, 334,
339, 351, 567
\l__enumext_labelsep_i_dim . . . 2249, 2285, 3680,
3695
\l__enumext_labelsep_v_dim . . . . . 2866
\l__enumext_labelsep_vii_dim . 3033, 3042, 3084,
3320, 3380, 3396
\l__enumext_labelsep_viii_dim 3444, 3453, 3495,
3734, 3757
\l__enumext_labelwidth_i_dim . 2249, 2284, 3680,
3695
\l__enumext_labelwidth_v_dim . . . . . 2866
\l__enumext_labelwidth_vii_dim . . . 3033, 3041,
3084, 3373, 3377, 3395
\l__enumext_labelwidth_viii_dim . . 3444, 3452,
3495, 3727, 3731, 3756
\l__enumext_leftmargin_tmp_v_bool . 85, 2988
\l__enumext_leftmargin_tmp_X_bool . . . . . 47
\l__enumext_leftmargin_tmp_X_dim . . . . . 47
\l__enumext_leftmargin_X_dim . . . . . 47
\__enumext_level: 191, 191, 431, 433, 434, 442, 444,
662, 666, 670, 737, 741, 745, 749, 833, 835, 837, 839,
872, 874, 876, 878, 882, 922, 925, 944, 953, 959, 964,
968, 979, 983, 984, 989, 1025, 1029, 1202, 1208, 1255,
1257, 1259, 1262, 1269, 1271, 1273, 1276, 1804, 1812,
1816, 1820, 2065, 2068, 2069, 2345, 2346, 2350, 2351,
2352, 2359, 2361, 2365, 2366, 2369, 2375, 2376, 2430,
2433, 2435, 2442, 2443, 2444, 2447, 2450, 2579, 2581,
2631, 2636, 2637, 2638, 2640, 2644, 2649, 2650, 2651,
2653, 2669, 2682, 2689, 2700, 2702, 2705, 2706, 2708,
2713, 2720, 2723, 2725, 2727, 2728, 2729, 2730, 2733,
2739, 2744, 2750, 2753, 2755, 2768
\l__enumext_level_h_int . 20, 223, 450, 474, 1658,
1675, 2021, 2038, 2602, 3169, 3170, 3178
\l__enumext_level_int 20, 193, 234, 845, 996, 1189,
1652, 1998, 2008, 2014, 2020, 2028, 2036, 2043, 2533,
2594, 2595, 2601, 2614, 2621, 2667, 2680, 2735, 2787,
2830, 2917, 3179, 3248, 3258, 3427, 3584
\__enumext_list_arg_two_i: . . . . . 2500
\__enumext_list_arg_two_ii: . . . . . 2500
\__enumext_list_arg_two_iii: . . . . . 2500
\__enumext_list_arg_two_iv: . . . . . 2500
\__enumext_list_arg_two_v: . 72, 2500, 2806, 2989
\__enumext_list_arg_two_vii: . . . . . 2539, 3151
\__enumext_list_arg_two_viii: . . . . . 2539, 3560
\l__enumext_listoffset_v_dim . . . . . 2868
\l__enumext_listparindent_vii_dim . . . . . 3383
\l__enumext_listparindent_viii_dim . . . 3737
\__enumext_make_label: 31, 70, 71, 73, 2438, 2438,
2528
\l__enumext_mark_answer_sym_tl . 61, 112, 1719,
1874, 2080, 2256, 2269, 3684
\l__enumext_mark_position_str 112, 1723, 1724,
1747, 1748, 1872
\l__enumext_mark_ref_sym_tl . . 112, 1733, 1978,
2200
\__enumext_mini_addvspace: . . 45, 78, 1018, 1018,
2710
\__enumext_mini_addvspace_vii: 47, 1171, 1171,
3109
\__enumext_mini_addvspace_viii: 47, 1171, 1177,
3520
\__enumext_mini_env* . . . . . 912
\__enumext_mini_right_cmd:n . 48, 49, 1197, 1199,
1199
\__enumext_mini_set_vskip: . . 43, 919, 919, 1020
\__enumext_mini_set_vskip_vii: 47, 1114, 1114,
1173
\__enumext_mini_set_vskip_viii: 47, 1114, 1136,
1179
\__enumext_minipage:w 28, 249, 251, 914, 3020, 3382,
3736
\l__enumext_minipage_active_v_bool . . 82, 83,
2849, 2874, 2887, 2895
\g__enumext_minipage_active_vii_bool . . . 88,
3120, 3125, 3137
\l__enumext_minipage_active_vii_bool . 3105,
3116
\g__enumext_minipage_active_viii_bool 3531,
3536, 3548
\l__enumext_minipage_active_viii_bool 3516,
3527
\g__enumext_minipage_active_X_bool . . . 158
\l__enumext_minipage_active_X_bool . . . . . 61
\g__enumext_minipage_after_skip 61, 1118, 1130,
3135, 3546
\l__enumext_minipage_after_skip 43, 44, 80, 83,
61, 935, 950, 970, 986, 1001, 1007, 1013, 1027, 1037,
1046, 1049, 1061, 1079, 1090, 1106, 1138, 1151, 1165,
2777, 2904
\g__enumext_minipage_center_vii_bool . 3129,
3138
\g__enumext_minipage_center_viii_bool 3540,
3549
\g__enumext_minipage_center_X_bool . . . 158
\l__enumext_minipage_hsep_v_dim . . . 81, 2847
\l__enumext_minipage_hsep_vii_dim . . . . . 3103
\l__enumext_minipage_hsep_viii_dim . . . 3514
\l__enumext_minipage_left_skip 43, 82, 61, 927,
942, 961, 976, 1023, 1033, 1038, 1044, 1053, 1070,
1082, 1102, 1112, 1116, 1121, 1125, 1139, 1143, 1157,
1175, 1181
\l__enumext_minipage_left_v_dim 81, 2845, 2853
\l__enumext_minipage_left_vii_dim 3099, 3111
\l__enumext_minipage_left_viii_dim 3510, 3522
\l__enumext_minipage_left_X_dim . . . . . 61
\g__enumext_minipage_right_skip 61, 1117, 1122,
1126, 3128, 3539
\l__enumext_minipage_right_skip . . 43, 61, 931,

```

946, 966, 981, 1039, 1045, 1057, 1075, 1086, 1140,
 1147, 1161, 1209, 1226
 \l__enumext_minipage_right_v_dim .. 81, 1220,
 1225, 2843, 2847
 \g__enumext_minipage_right_vii_dim 87, 3107,
 3127, 3140
 \l__enumext_minipage_right_vii_dim 87, 3097,
 3102, 3108
 \g__enumext_minipage_right_viii_dim .. 3518,
 3538, 3551
 \l__enumext_minipage_right_viii_dim .. 3508,
 3513, 3519
 \g__enumext_minipage_right_X_dim 158
 \g__enumext_minipage_right_X_skip 158
 \g__enumext_minipage_stat_int . 78, 82, 61, 1214,
 1231, 2709, 2770, 2775, 2850, 2897, 2902
 \g__enumext_miniright_code_vii_tl . 88, 3133,
 3139
 \g__enumext_miniright_code_viii_tl 3544, 3550
 \g__enumext_miniright_code_X_tl 158
 __enumext_multi_addvspace: ... 42, 79, 867, 867,
 2741
 __enumext_multi_set_vskip: .. 41, 831, 831, 869
 \l__enumext_multicols_above_ii_skip ... 850
 \l__enumext_multicols_above_iii_skip .. 856
 \l__enumext_multicols_above_iv_skip ... 862
 \l__enumext_multicols_above_v_skip 886, 900,
 910
 \l__enumext_multicols_above_X_skip 55
 \l__enumext_multicols_below_v_skip 890, 904,
 2889
 \l__enumext_multicols_below_X_skip 55
 __enumext_multicols_start: 79, 2715, 2717, 2717
 __enumext_multicols_stop: 79, 1204, 2747, 2747,
 2779
 __enumext_newlabel:nn 25, 29, 65, 292, 292, 2054,
 2162
 \l__enumext_newlabel_arg_one_tl 25, 29, 64, 66,
 136, 1977, 2047, 2055, 2151, 2163, 2198
 \l__enumext_newlabel_arg_two_tl 25, 29, 63, 136,
 2001, 2011, 2025, 2041, 2056, 2138, 2143, 2148, 2164
 __enumext_parse_keys:n 2575, 2609, 2609
 __enumext_parse_keys_parse_keys:n 53
 __enumext_parse_keys_vii:n 53, 3146, 3186, 3186
 __enumext_parse_keys_viii:n . 3556, 3589, 3589
 __enumext_parse_series_name:n 53, 1456, 1456,
 2617, 3192
 __enumext_parse_store_keys:n . 77, 2625, 2629,
 2629
 __enumext_parse_store_keys_vii:n . 89, 3195,
 3199, 3199
 \l__enumext_parsep_i_skip . 848, 850, 999, 1047
 \l__enumext_parsep_ii_skip 854, 856, 1005
 \l__enumext_parsep_iii_skip ... 860, 862, 1011
 \l__enumext_parsep_vii_skip 3384
 \l__enumext_parsep_viii_skip 3738
 \l__enumext_partopsep_v_skip .. 902, 906, 1073,
 1077, 1084, 1088, 1104, 1108
 \l__enumext_partopsep_viii_skip 1149
 __enumext_phantomsection: 29, 256, 285, 289, 305
 __enumext_print_footnote: ... 2304, 2327, 3402,
 3763
 __enumext_print_keyans_box:NN 61, 1866, 1866,
 1879, 2067, 2249, 2283, 3680, 3695
 \l__enumext_print_keyans_i_tl 3814, 3843
 \l__enumext_print_keyans_ii_tl ... 3819, 3844
 \l__enumext_print_keyans_iii_tl .. 3824, 3845
 \l__enumext_print_keyans_iv_tl ... 3829, 3846
 \l__enumext_print_keyans_vii_tl .. 3834, 3847
 \l__enumext_print_keyans_X_tl 101
 __enumext_printkeyans:nnn 102, 3848, 3851, 3851
 __enumext_redefine_item: . 71, 2378, 2378, 2527
 \l__enumext_ref_aux_tl 147, 431, 433, 436, 452, 454,
 457, 462, 464, 467
 \l__enumext_ref_key_arg_tl .. 147, 425, 430, 437,
 449, 458, 468
 __enumext_regex_label_ref_key: .. 32, 33, 420,
 420, 432, 453, 463
 __enumext_register_counter_style:Nn .. 324,
 324, 329, 330, 331, 332, 333
 __enumext_remove_extra_parsep_vii: .. 3161,
 3411, 3411
 __enumext_remove_extra_parsep_viii: . 3570,
 3772, 3772
 __enumext_renew_footnote: ... 2304, 2308, 3354,
 3717
 \l__enumext_resume_bool 23
 __enumext_resume_counter: 54
 __enumext_resume_counter:n .. 1487, 1491, 1496,
 1511, 1568, 1576
 __enumext_resume_counter_set: ... 1487, 1581,
 2783, 3242
 \g__enumext_resume_int 23, 80, 37, 1517, 1518, 1597
 __enumext_resume_last:n 53, 54, 1456, 1460, 1472
 \l__enumext_resume_name_tl 39, 1527, 1532, 1539,
 1593, 1601, 1603, 1617, 1625, 1627
 \l__enumext_resume_name_
 tl\l__enumext_item_
 symbol_tl 37
 __enumext_resume_series:n . 54, 1421, 1487, 1487
 __enumext_resume_starred: ... 1422, 1487, 1562
 \g__enumext_resume_vii_int .. 90, 37, 1522, 1523,
 1621
 __enumext_safe_exec: 27, 2574, 2591, 2591
 __enumext_safe_exec_vii: . 27, 3145, 3166, 3166
 __enumext_safe_exec_viii: ... 3555, 3575, 3575
 \l__enumext_series_str .. 1419, 1458, 1463, 1464,
 1466, 1468, 1589, 1591, 1595, 1613, 1615, 1619, 2613,
 3190
 __enumext_set_error:nn 3929, 3939, 3941
 __enumext_set_label_ref:n ... 32, 428, 428, 500
 __enumext_set_label_ref_h:n . 33, 447, 447, 520
 __enumext_set_parse:n 3912, 3929, 3929
 \l__enumext_setkey_tmpa_int ... 96, 3905, 3909
 \l__enumext_setkey_tmpa_seq 96, 3903, 3913, 3919,
 3921, 3923, 3936
 \l__enumext_setkey_tmpa_tl 96, 3911, 3915
 \l__enumext_setkey_tmpb_seq 96, 3904, 3907, 3911,
 3912
 \l__enumext_setkey_tmpb_tl 96, 3931, 3933, 3934
 \l__enumext_show_answer_bool . 112, 1727, 1751,
 2074, 2224, 2238, 2940, 3678
 __enumext_show_length:nnn .. 38, 201, 201, 3981,
 3982, 3983, 3984, 3985, 3986, 3987, 3988, 3989, 3990,
 3996, 3997, 3998, 3999, 4000, 4001, 4002, 4003, 4004,
 4005
 \l__enumext_show_position_bool 112, 1730, 1754,
 2078, 2228, 2239, 2941, 3682

```

\g__enumext_standar_bool . 27, 20, 222, 225, 1497,
    1515, 1528, 1564, 1583, 1678, 2600, 2793
\l__enumext_standar_bool . 20, 2006, 2019, 2035,
    2597, 2782
\g__enumext_standar_keyans_pic_star_env-
    int ..... 133
\g__enumext_standar_keyans_star_env_int 132
\l__enumext_standar_level_one_bool 20, 1380,
    1474, 1544, 2606
\g__enumext_standar_series_tl . 37, 1477, 1478,
    1566, 1569, 1574
\g__enumext_standar_star_env_int .. 129, 226
\__enumext_standard_item_vii:w 91, 3282, 3284,
    3284
\__enumext_standard_item_viii:w 98, 3626, 3628,
    3628
\g__enumext_starred_bool 27, 89, 90, 20, 233, 236,
    1502, 1520, 1535, 1572, 1607, 1657, 1997, 2007, 2037,
    2132, 2664, 2677, 2760, 3141, 3177, 3240, 3433
\l__enumext_starred_bool . 89, 90, 20, 1930, 1938,
    2022, 2063, 3174, 3241
\__enumext_starred_columns_set_vii:.. 3027,
    3027, 3154
\__enumext_starred_columns_set_viii: . 3438,
    3438, 3563
\__enumext_starred_item:nn ... 2355, 2355, 2384
\__enumext_starred_item_exec: . 99, 3671, 3671,
    3721
\__enumext_starred_item_vii:w 91, 92, 3281, 3300,
    3300
\__enumext_starred_item_vii_aux_i:w .. 3300,
    3305, 3308
\__enumext_starred_item_vii_aux_ii:w . 3300,
    3306, 3311, 3313
\__enumext_starred_item_vii_aux_iii:w 3300,
    3316, 3325
\__enumext_starred_item_viii:w 98, 3625, 3644,
    3644
\__enumext_starred_item_viii_aux_i:w . 3644,
    3649, 3652
\__enumext_starred_item_viii_aux_ii:w 3644,
    3650, 3664, 3666
\__enumext_starred_joined_item_vii:n . 87, 91,
    3046, 3046, 3279
\__enumext_starred_joined_item_viii:n 95, 98,
    3457, 3457, 3623
\g__enumext_starred_keyans_star_env_int 131
\l__enumext_starred_level_one_bool 20, 1390,
    1480, 1553, 3183
\g__enumext_starred_series_tl . 37, 1483, 1484,
    1577
\g__enumext_starred_star_env_int .. 130, 237
\__enumext_start_from:NNn 35, 541, 541, 554, 576
\l__enumext_start_i_int ..... 1518, 1530, 1547
\__enumext_start_item_tmp_vii: 88, 3157, 3264,
    3264
\__enumext_start_item_tmp_viii: 96, 3566, 3608,
    3608
\__enumext_start_item_vii:w . 91, 92, 3292, 3297,
    3322, 3329, 3331, 3331
\__enumext_start_item_viii:w .. 98, 3636, 3641,
    3669, 3698, 3698
\__enumext_start_list:nn 28, 75, 85, 243, 245, 2578,
    2803, 2954, 3149, 3558
\__enumext_start_mini_vii: . 90, 3095, 3095, 3233
\__enumext_start_mini_viii: 97, 3506, 3506, 3600
\__enumext_start_store_level: . 77, 2577, 2658,
    2658
\__enumext_start_store_level_vii: . 90, 3148,
    3244, 3244
\l__enumext_start_vii_int ... 1523, 1537, 1556
\l__enumext_start_X_int ..... 73, 571
\__enumext_stop_item_tmp_vii: . 88, 91, 92, 3156,
    3160, 3266, 3333
\__enumext_stop_item_tmp_viii: .. 96, 97, 3565,
    3569, 3610, 3700
\__enumext_stop_item_vii: 92, 93, 3333, 3387, 3387
\__enumext_stop_item_viii: 100, 3700, 3748, 3748
\__enumext_stop_list: .. 28, 243, 246, 2587, 2813,
    2967, 3162, 3572
\__enumext_stop_mini_vii: 88, 90, 3114, 3114, 3237
\__enumext_stop_mini_viii: . 97, 3506, 3525, 3604
\__enumext_stop_store_level: .. 77, 2588, 2658,
    2687
\__enumext_stop_store_level_vii: .. 90, 3163,
    3244, 3254
\l__enumext_store_active_bool 24, 51, 77, 89, 85,
    1408, 1545, 1554, 1892, 2623, 2662, 2675, 2818, 2825,
    2913, 2971, 3193, 3246, 3256, 3583
\__enumext_store_addto_prop:n 59, 66, 1785, 1786,
    1794, 1917, 2113, 3674
\__enumext_store_addto_seq:n 60, 67, 1795, 1795,
    1799, 1806, 1820, 1828, 1837, 1855, 1863, 1981, 2203
\l__enumext_store_ans_bool 122, 1409, 1639, 1802,
    1826, 1833, 1861, 1905
\l__enumext_store_anskey_arg_tl 24, 62, 63, 85,
    1923, 1932, 1934, 1940, 1948, 1951, 1961, 1966, 1969,
    1975, 1981
\__enumext_store_anskey_code:nnnn . 62, 1911,
    1915, 1915
\__enumext_store_anskey_show_left:n 65, 1922,
    2072, 2072
\__enumext_store_anskey_show_wrap:n 65, 2060,
    2060, 2076, 2091
\l__enumext_store_columns_break_bool . 1886,
    1929
\l__enumext_store_columns_join_int 85, 1937,
    1942
\l__enumext_store_columns_sep_vii_bool 3214
\l__enumext_store_columns_sep_vii_dim 3219,
    3223
\l__enumext_store_columns_sep_X_bool .. 101
\l__enumext_store_columns_sep_X_dim ... 101
\l__enumext_store_columns_vii_bool ... 3201
\l__enumext_store_columns_vii_int 3206, 3210
\l__enumext_store_columns_X_bool ..... 101
\l__enumext_store_columns_X_int ..... 101
\__enumext_store_internal_ref: .. 62, 63, 1920,
    1983, 1983
\l__enumext_store_item_symbol_sep_dim 1884,
    1958, 1963
\l__enumext_store_item_symbol_tl . 1882, 1949,
    1953
\l__enumext_store_keyans_item_opt_sep-
    tl .... 1716, 2107, 2109, 2180, 2182, 3657, 3659
\l__enumext_store_keyans_item_opt_tl ... 85
\l__enumext_store_keyans_label_tl 24, 65, 67,
    85, 2096, 2099, 2102, 2109, 2111, 2113, 2170, 2173,
    2176, 2182, 2184, 2194, 2203, 2204, 3654, 3659, 3660,

```

```

    3673, 3674, 3676
\__enumext_store_level_close: . 60, 1800, 1824,
    2691
\__enumext_store_level_close_vii: 1831, 1859,
    3260
\__enumext_store_level_open: .. 59, 60, 77, 1800,
    1800, 2670, 2683
\__enumext_store_level_open_vii: .. 89, 1831,
    1831, 3250
\g__enumext_store_name_tl 24, 80, 85, 1698, 1701,
    2763, 2795, 3435
\l__enumext_store_name_tl 24, 51, 85, 1358, 1359,
    1369, 1370, 1400, 1402, 1404, 1406, 1410, 1412, 1549,
    1558, 1585, 1587, 1609, 1611, 1788, 1790, 1797, 2049,
    2050, 2086, 2153, 2154, 2262, 2275, 2763, 3690
\l__enumext_store_opt_vii_tl . 1835, 1845, 1851,
    1855, 3208, 3221
\l__enumext_store_opt_X_tl ..... 101
\l__enumext_store_ref_key_bool 62, 1736, 1918,
    1972, 2117, 2191
\l__enumext_store_upper_level_X_bool .. 101
\l__enumext_store_write_aux_file_tl 25, 65, 67,
    136, 2052, 2058, 2160, 2166
\__enumext_storing_exec: 1356, 1382, 1392, 1398
\__enumext_storing_set:n .. 51, 1348, 1356, 1356
\__enumext_storing_set_vii:n ..... 1353, 1367
\__enumext_storing_standar: ..... 1364, 1378
\__enumext_storing_starred: ..... 1375, 1388
\l__enumext_the_counter_vii_tl ..... 454
\l__enumext_the_counter_viii_tl ..... 464
\l__enumext_the_counter_X_tl ..... 147
\__enumext_tmp:n 32, 36, 47, 54, 55, 60, 67, 72, 73, 84,
    101, 111, 139, 143, 151, 157, 158, 177, 209, 213, 654,
    658, 1415, 1426, 1632, 1649, 1706, 1741, 1742, 1759,
    1985, 1992, 1993, 2014, 2028, 2031, 2043, 2119, 2126,
    2500, 2538, 2539, 2571
\__enumext_tmp:nn 356, 377, 378, 406, 407, 419, 571,
    590, 635, 653, 711, 719, 720, 734, 799, 815, 816, 830,
    1235, 1251, 1760, 1784, 2288, 2303
\__enumext_tmp:nnn 488, 504, 505, 506, 507, 508, 524,
    525
\__enumext_tmp:nnnnn 591, 616, 619, 622, 624, 626,
    629, 632
\__enumext_tmp:w ..... 3797, 3800
\l__enumext_tmpa_vii_int ..... 3037, 3040
\l__enumext_tmpa_viii_int ..... 3448, 3451
\l__enumext_tmpa_X_int ..... 158
\l__enumext_topsep_v_skip 888, 892, 1042, 1055,
    1063, 1068, 1088, 1092, 2970, 3001
\l__enumext_topsep_vii_skip .. 1119, 1128, 1132
\l__enumext_topsep_viii_skip . 1141, 1163, 1167
\__enumext_use_key_ref: .... 33, 440, 440, 2529
\__enumext_use_key_ref_h: .. 33, 472, 472, 2557
\l__enumext_vspace_a_star_v_bool ..... 1284
\l__enumext_vspace_a_star_vii_bool ... 1306
\l__enumext_vspace_a_star_viii_bool ... 1317
\l__enumext_vspace_a_star_X_bool ..... 73
\__enumext_vspace_above: .. 49, 1252, 1252, 2696
\__enumext_vspace_above_v: . 50, 1280, 1280, 2841
\l__enumext_vspace_above_v_skip .. 1282, 1286,
    1288
\__enumext_vspace_above_vii: .. 50, 1302, 1302,
    3230
\l__enumext_vspace_above_vii_skip 1304, 1308,
    1310
\__enumext_vspace_above_viii: . 50, 1302, 1313,
    3598
\l__enumext_vspace_above_viii_skip 1315, 1319,
    1321
\l__enumext_vspace_b_star_v_bool ..... 1295
\l__enumext_vspace_b_star_vii_bool ... 1328
\l__enumext_vspace_b_star_viii_bool ... 1339
\l__enumext_vspace_b_star_X_bool ..... 73
\__enumext_vspace_below: .. 50, 1266, 1266, 2781
\__enumext_vspace_below_v: . 50, 1291, 1291, 2909
\l__enumext_vspace_below_v_skip .. 1293, 1297,
    1299
\__enumext_vspace_below_vii: .. 51, 1324, 1324,
    3239
\l__enumext_vspace_below_vii_skip 1326, 1330,
    1332
\__enumext_vspace_below_viii: . 51, 1324, 1335,
    3606
\l__enumext_vspace_below_viii_skip 1337, 1341,
    1343
\__enumext_widest_from:nnnn .. 35, 555, 555, 570,
    582
\g__enumext_widest_label_tl 23, 31, 43, 344, 348,
    352
\l__enumext_wrap_label_opt_v_bool .... 2397
\l__enumext_wrap_label_opt_vii_bool 91, 3291
\l__enumext_wrap_label_opt_viii_bool 98, 3635
\l__enumext_wrap_label_opt_X_bool ..... 73
\l__enumext_wrap_label_v_bool 2393, 2397, 2405,
    2460
\l__enumext_wrap_label_vii_bool 91, 3290, 3295,
    3303, 3371
\l__enumext_wrap_label_viii_bool .. 98, 3634,
    3639, 3647, 3725
\l__enumext_wrap_label_X_bool ..... 73
\__enumext_wrapper_label_v:n ..... 2462, 2949
\__enumext_wrapper_label_vii:n ..... 3374
\__enumext_wrapper_label_viii:n ..... 3728
\__enumext_zero_count_level: ..... 207, 207
\__enumext_zero_parsep: ..... 44, 939, 994, 994
enumext* ..... 5, 3143
enumXi ..... 316
enumXii ..... 316
enumXiii ..... 316
enumXiv ..... 316
enumXv ..... 316
enumXvi ..... 316
enumXvii ..... 316
enumXviii ..... 316
Environments provide by enumext:
enumext* 22, 23, 25–27, 30, 32–34, 37, 38, 40, 47, 50–54,
    57–64, 66, 69, 75, 76, 78, 89, 90, 92, 94, 95, 97, 99, 102,
    105, 106
enumext 22, 23, 25, 27, 30, 31, 33–46, 48–54, 57–62, 64, 66,
    69–78, 80, 81, 85–87, 90, 102, 105
keyans* 22–24, 26, 27, 30, 32–34, 37, 38, 40, 47, 50, 51, 56,
    57, 59, 66, 69, 75, 96, 97, 105, 106
keyanspic 22–25, 30, 31, 34, 48, 51, 56, 57, 59, 60, 65–68,
    83–85, 106
keyans 22–25, 27, 30, 31, 34–40, 42, 45, 46, 48–51, 56, 57,
    59, 60, 65–68, 72–75, 80, 81, 83–85, 87, 97, 105, 106
Environments:
list ..... 27, 28, 74–76
lrbox ..... 86, 93, 100

```

minipage 27, 28, 40, 43, 83–86, 93, 100
 multicol 41–43, 48, 79, 80, 82, 83
 exp commands:
 \exp_after:wN 3800
 \exp_args:Ne 2620, 3788
 \exp_not:N 155, 347, 436, 457, 467, 668, 682, 683, 694,
 695, 706, 707, 1977, 2083, 2084, 2196, 2259, 2260,
 2272, 2273, 3687, 3688, 3797
 \exp_not:n 436, 437, 457, 458, 467, 468, 669, 1443, 1454,
 1768, 1775, 1942, 1953, 1963, 1977, 1978, 2055, 2163,
 2198, 2200, 2640, 2653, 3210, 3223

F

\fbox 1711
 file commands:
 \file_input_stop: 4106
 first 720
 font 356
 \footnote 69
 \footnote 69, 2312
 \footnotemark 2322
 \footnotesize 2084, 2260, 2273, 3688
 \footnotetext 2306

G

\getkeyans 13, 101, 3786
 group commands:
 \group_begin: .. 1904, 2082, 2258, 2271, 3350, 3369,
 3686, 3713, 3723, 3808, 3842
 \group_end: 1913, 2089, 2265, 2278, 3379, 3391, 3693,
 3733, 3752, 3810, 3849

H

\hbadness 3398, 3759
 hbox commands:
 \hbox_set:Nn 336
 \hfill 386, 390, 395, 396, 1206, 1224, 1977, 2196, 3119, 3530
 hook commands:
 \hook_gput_code:nnn 9, 185, 189, 254
 \hook_gset_rule:nnnn 255
 \hspace 3409, 3770
 \hyperlink 63, 67
 \hyperlink 1977, 2196
 \hypertarget 29
 \hypertarget 284

I

\IfHyperBoolean 262
 \IfPackageLoadedTF 11, 258, 272
 \ignorespaces 671
 \inputlineno 226, 237
 int commands:
 \int_add:Nn 3079, 3490
 \int_case:nn 845, 996, 1652, 1675
 \int_compare:nNnTF 450, 474, 921, 1040, 1185, 1189,
 1193, 1695, 1896, 1900, 2097, 2136, 2141, 2146, 2171,
 2254, 2595, 2614, 2667, 2680, 2719, 2735, 2749, 2770,
 2787, 2826, 2830, 2859, 2884, 2897, 2917, 2921, 2976,
 3049, 3059, 3075, 3170, 3248, 3258, 3404, 3413, 3427,
 3460, 3470, 3486, 3578, 3584, 3765, 3774, 3909
 \int_compare_p:nNn 223, 234, 1658, 1998, 2008, 2020,
 2021, 2036, 2038, 2601, 2602, 3178, 3179
 \int_decr:N 3078, 3489
 \int_eval:n 1790, 2050, 2084, 2154, 2260, 2273, 2515,
 2556, 3067, 3478, 3688
 \int_from_alpha:n 549, 563

\int_from_roman:n 551, 565
 \int_gadd:Nn 3080, 3491
 \int_gdecr:N 1661, 1666, 1669, 1672, 1680
 \int_gincr:N 1517, 1522, 1909, 2207, 2343, 2373, 2709,
 2850, 3268, 3346, 3612
 \int_gset:Nn 226, 237, 2320
 \int_gset_eq:NN 1587, 1591, 1597, 1603, 1611, 1615,
 1621, 1627, 2317
 \int_gzero:N 211, 1214, 1231, 1703, 1704, 2775, 2902,
 3422, 3783
 \int_if_exist:NTF 1410, 1466, 1585, 1601, 1609, 1625
 \int_incr:N 2594, 2822, 2975, 3169, 3267, 3577, 3611
 \int_mod:nn 3415, 3776
 \int_new:N 20, 21, 22, 23, 24, 37, 38, 61, 77, 89, 98, 106,
 119, 120, 127, 128, 129, 130, 131, 132, 133, 144, 161,
 162, 163, 164, 165, 1412, 1468
 \int_set:Nn 545, 549, 551, 1530, 1537, 1547, 1556, 1765,
 1937, 3014, 3015, 3037, 3048, 3054, 3070, 3398, 3448,
 3459, 3465, 3481, 3759, 3905
 \int_set_eq:NN . 1518, 1523, 2635, 3077, 3205, 3488
 \int_step_function:nnN 2014, 2028, 2043
 \int_step_inline:nnn 3016, 3932
 \int_to_roman:n 193, 1994, 2032
 \int_use:N .. 922, 1532, 1539, 1549, 1558, 2515, 2533,
 2556, 2621, 2720, 2729, 2744, 2750, 3052, 3053, 3065,
 3463, 3464, 3476
 \int_zero:N 3407, 3768
 \c_one_int . 3037, 3056, 3062, 3068, 3072, 3075, 3448,
 3467, 3473, 3479, 3483, 3486
 \c_zero_int .. 223, 234, 1998, 2008, 2020, 2021, 2036,
 2038, 3248, 3258, 3418, 3779

\item 28, 39, 40, 60, 70, 83, 85, 86, 88, 96
 \item 70, 72, 91, 92, 97, 99, 247, 1808, 1814, 1839, 1847, 1934,
 2173, 2176, 2380, 2414, 3155, 3157, 3564, 3566, 3676
 \item* 6, 12, 2412
 item-pos* 2288
 item-sym* 2288
 \itemindent 23, 74
 \itemindent 73
 itemindent 635
 \itemsep 84, 85
 \itemsep 2990, 2996
 \itemwidth 3044, 3088, 3092, 3455, 3499, 3503

K

keyans 11, 2798
 keyans* 11, 3553
 keyanspic 12, 2951

Keys for environments provide by **enumext**:

above* 24, 49, 50
 above 24, 49, 50, 78, 81, 90, 97
 after 38–40, 80, 83, 90, 97
 align 24, 31, 32, 72, 93
 before* 38, 39, 78, 90, 97
 before 38–40, 81
 below* 24, 49–51
 below 24, 49–51, 80, 83, 90, 97
 check-ans 24, 25, 27, 57, 62, 67, 68, 70, 71, 78, 80, 94, 105
 columns-sep* 25, 59, 77, 89
 columns-sep 40, 60, 77, 79, 82, 89
 columns* 25, 59, 77, 89
 columns 23, 40, 43, 49, 60, 77, 79, 82, 89
 first 38–40, 93
 font 31, 72, 93
 item-pos* 62, 63, 69

item-sym*	23, 62, 63, 69, 71
item*-sep	71
itemindent	24, 37, 72, 93
itemsep	36, 75
labelsep	31, 71, 74, 93
labelwidth	30, 31, 34, 35, 74
label	23, 30, 31, 34, 35, 86
lisparindent	75
list-indent	23, 37, 85
list-offset	37
listparindent	37, 93
mark-ans	25, 58, 65
mark-pos	58, 59
mark-ref	25, 58, 63
mini-env	24, 40, 43, 48, 49, 69, 78, 81, 87, 90, 95, 97
mini-sep	24, 40, 78, 81
miniright*	24, 40
miniright	24, 40, 47, 88
minirigth*	27
minirigth	27
no-store	25, 57, 58
noitemsep	36, 44
nosep	36, 44
parindent	75
parsep	36, 75, 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, 59, 60, 62, 66, 67, 72, 80, 81, 83, 84, 90, 97, 99, 101, 102, 105
save-key	25, 53
save-ref	25, 29, 58, 62, 63, 66, 67, 72, 99
save-sep	58
series	52-54
show-ans	25, 58, 59, 61, 62, 65, 72, 98, 99
show-length	27, 38, 75, 105
show-pos	25, 58, 59, 61, 62, 65, 68, 72, 98, 99
start	24, 27, 35, 75
store-brk	62
topsep	36
widest	23, 27, 35
wrap-ans	58, 61, 65
wrap-label*	31, 70, 72, 91, 93, 98
wrap-label	31, 72, 91, 93, 98
wrap-opt	58
keys commands:	
\keys_define:nn	358, 380, 409, 490, 510, 526, 573, 593, 637, 656, 713, 722, 801, 818, 1237, 1346, 1351, 1417, 1634, 1708, 1744, 1762, 1880, 2290, 3812, 3875
\l_keys_key_str	3966
\keys_set:nn	372, 825, 1242, 1247, 1499, 1504, 1569, 1577, 1926, 2616, 2620, 2837, 3191, 3593, 3877, 3878, 3879, 3880, 3881, 3882, 3883, 3884, 3885, 3886, 3887, 3888, 3926
keyval commands:	
\keyval_parse:NNn	1431
L	
label	488, 508, 526
Labels provide by enumext:	
\Alph*	30, 31
\Roman*	30, 31
\alph*	30, 31
\arabic*	30-32
\roman*	30, 31
\labelsep	85
\labelsep	2991, 2994
labelsep	356
\labelwidth	31, 85
\labelwidth	2991, 2992
labelwidth	356
\leftmargin	23, 74
\leftmargin	73, 2991
legacy commands:	
\legacy_if:nTF	3334, 3337, 3701, 3704
\legacy_if_gset_false:n	915
\legacy_if_set_false:n	3336, 3703
\legacy_if_set_true:n	3296, 3321, 3328, 3341, 3640, 3668, 3708
\linewidth	78, 81
\linewidth	2704, 2847, 3013, 3040, 3101, 3451, 3512
\list	28
\list	245
list-indent	635
list-offset	635
\listparindent	2993
listparindent	635
\lrbox	3351, 3714
M	
\makebox	86
\makebox	1870, 1872, 2434, 3365, 3373, 3377, 3727, 3731
\makelabel	70, 72, 73, 86
\makelabel	72, 73, 2440, 2456
\makesavenoteenv	278
mark-ans	1706
mark-pos	1706, 1742
mark-ref	1706
mini-env	799
mini-sep	799
\minipage	28
\minipage	251
\miniright	10, 48, 1183, 2773, 2900
\miniright*	10
mode commands:	
\mode_if_vertical:TF	870, 898, 1021, 1100
\mode_leave_vertical:	668, 682, 694, 706, 1839, 1847, 1868, 2432, 3363
msg commands:	
\msg_error:nn	2828, 2832, 2919, 2978, 3172, 3580, 3586, 3889
\msg_error:nnn	1187, 1191, 1216, 1233, 1361, 1372, 1508, 3802, 3807, 3872, 3942
\msg_error:nnnn	1894, 1898, 1902, 2820, 2915, 2923
\msg_fatal:nn	2596
\msg_fatal:nnn	310
\msg_info:nnn	13, 16, 260, 274
\msg_line_context:	3970, 3975, 3980, 3995, 4010, 4014, 4026, 4030, 4034, 4039, 4044, 4049, 4054, 4058, 4063, 4068, 4072, 4077, 4081, 4086, 4091, 4096, 4100, 4104
\msg_new:nnn	3943, 3947, 3951, 3955, 3960, 3964, 3968, 3973, 3978, 3993, 4008, 4012, 4016, 4023, 4028, 4032, 4037, 4042, 4047, 4052, 4056, 4061, 4066, 4070, 4075, 4079, 4084, 4089, 4094, 4098, 4102
\msg_term:nnn	1698
\msg_term:nnnn	2523, 2533, 2562, 2567
\msg_warning:nn	2772, 2899
\msg_warning:nnn	1385, 1395, 1701

\msg_warning:nnnn	2214, 2472, 2477, 3051, 3064, 3462, 3475
\multicolsep	79, 82
\multicolsep	2734, 2872
N	
\NeedsTeXFormat	3
\newcounter	313
\NewDocumentCommand	1183, 1890, 2911, 3786, 3840, 3896
\NewDocumentEnvironment	2572, 2798, 2951, 3143, 3553
\newlabel	29
\newlabel	296
no-store	1632
\noindent	88, 96
\noindent	2711, 2852, 3110, 3156, 3406, 3521, 3565, 3767
\nointerlineskip	2711, 2852, 3110, 3521
noitemsep	591
\nopagebreak	881, 909, 1032, 1111, 1174, 1180
\normalfont	2083, 2259, 2272, 3687
nosep	591
P	
Packages:	
enumext	22, 51, 74, 83, 104
enumitem	30
expl3	86
footnotehyper	29
hyperref	25, 27, 29, 33, 63, 67, 92, 104
lua-visual-debug	43
multicol	22, 104
shortlst	86
\par	881, 909, 1032, 1111, 1174, 1180, 1209, 1226, 2062, 2755, 2777, 2889, 2904, 3025, 3128, 3135, 3406, 3420, 3539, 3546, 3767, 3781
\parindent	3383, 3737
\parsep	41, 44, 84, 85
\parsep	1840, 1848, 2553, 2990, 2997, 3002
parsep	591
\parskip	3384, 3738
\partopsep	85
\partopsep	2554, 2995
partopsep	591
peek commands:	
\peek_meaning:N	3273, 3287, 3304, 3315, 3617, 3631, 3648
\peek_meaning_remove:N	3280, 3624
\peek_remove_spaces:n	2418
\phantomsection	29
\phantomsection	285
prg commands:	
\prg_do_nothing:	289
\prg_new_protected_conditional:Npnn	195
\prg_replicate:nn	204, 4021
\prg_return_false:	199
\prg_return_true:	198
\printkeyans	13, 102, 3840
prop commands:	
\prop_count:N	1790, 2050, 2086, 2154, 2262, 2275, 3690
\prop_gput_if_not_in:Nnn	1785, 1788
\prop_if_exist:N	1400, 3806
\prop_item:Nn	3809
\prop_new:N	1402
\ProvidesExplPackage	4
R	
\raggedcolumns	2743, 2878

\ref	63, 66
ref	488, 508
\refstepcounter	3343, 3710
regex commands:	
\regex_match:nnTF	197, 548, 550, 562, 564, 2633, 2646, 3203, 3216
\regex_replace_once:nnN	424
\renewcommand	436, 457, 467
\RenewDocumentCommand	2312, 2380, 2414, 2440, 2456
\RequirePackage	17
resume	1415
resume*	1415
rightmargin	635
\Roman	31, 35
\Roman	332
\roman	31, 35
\roman	333, 506, 3828
S	
save-ans	1346
save-ref	1706
save-sep	1706
scan commands:	
\scan_stop:	85, 3004, 3155, 3564, 3797, 3800
seq commands:	
\seq_clear:N	3903
\seq_const_from_clist:Nn	3891
\seq_count:N	2964, 3907
\seq_gclear:N	2310, 2311
\seq_gput_right:Nn	1797, 2323, 2324
\seq_if_empty:N	2329, 3855, 3921
\seq_if_exist:N	1404, 3853
\seq_item:Nn	3022
\seq_map_function:NN	3912
\seq_map_inline:Nn	3861, 3866, 3900, 3922, 3923
\seq_map_pairwise_function:NNN	2331
\seq_new:N	99, 100, 117, 145, 146, 1406
\seq_pop_left:NN	3911
\seq_put_right:Nn	2925, 3919, 3936
\seq_set_from_clist:Nn	3904
\seq_set_map_e:NNn	3913
\seq_show:N	3857
series	1415
\setcounter	559, 563, 565, 2515, 2556, 2969
\setenumext	6-9, 103, 3816, 3821, 3826, 3831, 3836, 3896
\setlength	1841, 1849
show-ans	1706, 1742
show-length	711
skip commands:	
\skip_add:Nn	850, 856, 862, 872, 876, 900, 904, 1001, 1007, 1013, 1023, 1027, 1049, 1102, 1106, 2990
\skip_eval:n	1840, 1848
\skip_gset:Nn	1122, 1126, 1130
\skip_gzero_new:N	1117, 1118
\skip_horizontal:N	683, 695, 707, 3366, 3380, 3734
\skip_horizontal:n	669, 1869, 1877, 2433, 2435, 3364, 3742
\skip_if_eq:nnTF	848, 854, 860, 924, 958, 999, 1005, 1011, 1042, 1047, 1068, 1119, 1141, 1254, 1268, 1282, 1293, 1304, 1315, 1326, 1337
\skip_new:N	57, 58, 62, 63, 64, 65, 66, 121, 175
\skip_set:Nn	833, 837, 886, 890, 927, 931, 935, 942, 946, 950, 961, 966, 970, 976, 981, 986, 1044, 1045, 1046, 1053, 1057, 1061, 1070, 1075, 1079, 1082, 1086,

1090, 1121, 1125, 1143, 1147, 1151, 1157, 1161, 1165, 2984, 2998	
\skip_set_eq:NN	2513, 2552, 2553, 3383, 3384, 3737, 3738
\skip_use:N	835, 839, 874, 878, 882, 902, 906, 925, 944, 953, 959, 964, 968, 979, 983, 984, 989, 1025, 1029, 1055, 1255, 1259, 1262, 1269, 1273, 1276, 2755
\skip_zero:N	2554, 2734, 2872, 2995, 2996
\skip_zero_new:N	1037, 1038, 1039, 1116, 1138, 1139, 1140
\c_zero_skip	848, 854, 860, 925, 959, 999, 1005, 1011, 1042, 1047, 1068, 1119, 1141, 1255, 1269, 1282, 1293, 1304, 1315, 1326, 1337
\small	3818, 3823, 3828, 3833, 3838
\star	2294
start	571
\stepcounter	2316, 2932
str commands:	
\c_backslash_str	4030, 4039, 4040, 4044, 4045, 4049, 4050, 4081, 4082, 4086, 4091, 4092
\c_colon_str	2049, 2153, 3797
\str_case:nn	217
\str_case:nnTF	1438, 1447
\str_clear:N	2613, 3190
\str_count:n	204, 4021
\str_if_empty:NTF	1458, 1595, 1619
\str_if_eq:nnTF	2516, 2558
\str_if_in:nnTF	3793
\str_new:N	116, 170
\str_set:Nn	412, 413, 414, 1723, 1724, 1747, 1748
\string	278
\strutbox	929, 933, 937, 948, 952, 963, 972, 978, 988, 1001, 1007, 1013, 1044, 1045, 1046, 1049, 1059, 1063, 1072, 1079, 1084, 1092, 1121, 1122, 1125, 1132, 1145, 1153, 1159, 1167, 3000

T

TeX and \TeX 2_ε commands:

\@auxout	294
\@currentenv	217
\protected@write	294

text commands:

\text_expand:n	3789
\textasteriskcentered	1720, 1734
\thepage	300

tl commands:

\c_space_tl	2241, 3980, 3995
\tl_clear:N	385, 391, 1923, 2096, 2170, 3654
\tl_clear_new:N	342
\tl_const:Nn	147, 326
\tl_gclear:N	1477, 1483, 2216, 2451, 2795, 3139, 3367, 3435, 3550
\tl_gclear_new:N	1463
\tl_gput_right:Nn	327
\tl_greplace_all:Nnn	348
\tl_gset:Nn	1464, 1478, 1484, 2204, 2763, 3310
\tl_gset_eq:NN	344, 2361, 3360
\tl_if_blank:nTF	3358
\tl_if_empty:NTF	442, 476, 482, 1359, 1370, 1566, 1574, 1589, 1593, 1613, 1617, 1804, 1835, 1949, 2107, 2180, 2212, 2235, 2430, 3657, 3934
\tl_if_empty:nTF	1489, 1513

\tl_if_exist:NTF	1494
\tl_if_nvalue:nTF	1924, 1935, 2104, 2178, 2220, 2314, 2339, 2357, 2362, 2391, 2611, 2962, 3188, 3591, 3655, 3898
\tl_map_inline:Nn	345, 422
\tl_new:N	34, 39, 40, 41, 42, 44, 45, 78, 79, 80, 86, 87, 88, 90, 91, 92, 93, 94, 96, 97, 103, 104, 114, 115, 126, 136, 137, 138, 141, 149, 150, 153, 154, 169, 172
\tl_put_left:Nn	1812, 1845, 1932, 2247, 2281, 3673, 3676
\tl_put_right:Nn	343, 434, 455, 465, 1766, 1773, 1816, 1851, 1934, 1940, 1948, 1951, 1961, 1966, 1969, 1975, 2001, 2011, 2025, 2041, 2047, 2052, 2099, 2102, 2109, 2111, 2138, 2143, 2148, 2151, 2160, 2173, 2176, 2182, 2184, 2194, 2638, 2651, 3208, 3221, 3659, 3660, 3814, 3819, 3824, 3829, 3834
\tl_remove_all:Nn	3933
\tl_remove_once:Nn	1989, 2123
\tl_replace_all:Nnn	347
\tl_reverse:N	1988, 1990, 2122, 2124
\tl_set:Nn	155, 312, 386, 390, 395, 396, 430, 449, 666, 680, 692, 704, 1358, 1369, 1527, 2080, 2222, 2256, 2269, 2359, 3662, 3684, 3931
\tl_set_eq:NN	353, 431, 433, 452, 454, 462, 464, 1987, 2121, 2134, 2403, 2407, 2943, 2945
\tl_to_str:n	1494, 1500, 1505, 3789
\tl_trim_spaces:n	343, 3919, 3931, 3937
\tl_use:N	349, 352, 444, 478, 484, 737, 741, 745, 749, 753, 757, 761, 765, 769, 773, 777, 781, 785, 789, 793, 797, 1874, 1994, 2002, 2013, 2027, 2032, 2044, 2346, 2352, 2376, 2394, 2398, 2406, 2442, 2443, 2450, 2458, 2459, 2465, 2579, 2804, 2948, 3133, 3370, 3381, 3385, 3544, 3724, 3735, 3741, 3745, 3843, 3844, 3845, 3846, 3847, 3915

token commands:

\token_to_str:N	296
\topsep	1841, 1849
topsep	591
\typeout	227, 238, 264, 267, 277, 278, 1476, 1482, 1662, 1681, 2605, 3182

U

\u	425
----	-----

use commands:

\use:N	205, 2447, 2581
\use:n	1429, 3795
\use_none:nn	288
\usecounter	2514, 2555

V

\value	1587, 1591, 1597, 1603, 1611, 1615, 1621, 1627
\vspace	916, 1259, 1262, 1273, 1276, 1286, 1288, 1297, 1299, 1308, 1310, 1319, 1321, 1330, 1332, 1341, 1343, 1840, 1848, 2959, 2970, 3421, 3782

W

widest	571
wrap-ans	1706
wrap-label	356
wrap-label*	356
wrap-opt	1706