

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-17^{*}

©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 keyans	10
1.3.3	Support for minipage	4	4.5.1	The <code>\item*</code> in keyans	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment keyanspic	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	108
3.5	Keys for multicol	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 multicol, vertical alignment

^{*}This file describes a documentation for v1.0, last revised 2024-05-17.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

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

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

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

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

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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

(A) value

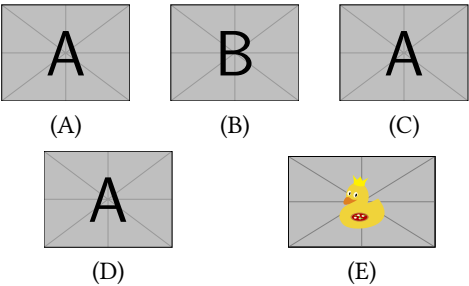
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
-

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

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

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

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

1.1 Description and usage

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

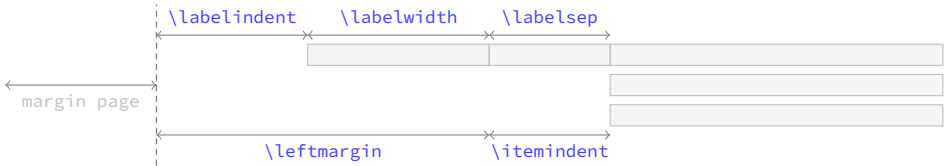


Figure 1: Representation of horizontal lengths in `enumitem`.

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

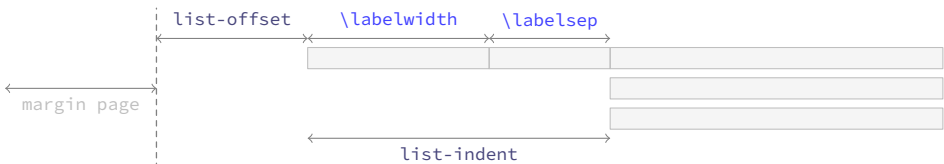


Figure 2: Representation of horizontal lengths concept in `enumext`.

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



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

1.3 User interface

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

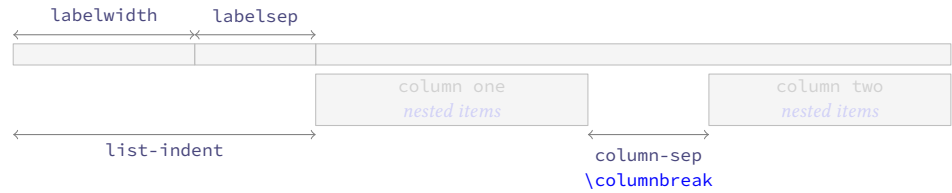


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 \TeX `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§4.2) when the key `save-ans`(§4.1) is active.

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

1.3.5 Support for \footnote

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

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

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

2 The environment enumext

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

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

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

Example

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

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

2.1 The `\item*` in `enumext`

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

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

- The following `⟨keys⟩` should be used with “*caution*”, they are intended to inject `{⟨code⟩}` into different parts of the defined environments. We must keep in mind that the defined environments are based on the `list` base environment provided by `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

- | | |
|---|--|
| <p>★ 1. Text containing our instructions or questions.
 * first answer</p> <p>2. Text containing our instructions or questions.
 (a) Question.
 * second answer</p> | <p>3. Text containing our instructions or questions.
 * third answer</p> <p>4. Text containing our instructions or questions.
 * fourth answer</p> |
|---|--|

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

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

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

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

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

4.5.1 The `\item*` in `keyans`

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

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

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

Example

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

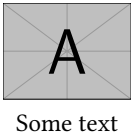
  \item Text containing a question and image.
  \begin{keyans}[nosep,mini-env={0.4\linewidth}]
    \item Choice
    \item Choice
    \item Choice
    \item Choice
    \item* [<note>] Correct choice
    \miniright
    \includegraphics[scale=0.25]{example-image-a}

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

1. Text containing a question.

(A) Choice
* (B) Correct choice
(C) Choice
(D) Choice
2. Text containing a question and image.

(A) Choice
(B) Choice
(C) Choice
(D) Choice
* (E) [note] Correct choice



Some text

4.6 The environment `keyanspic`

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

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

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

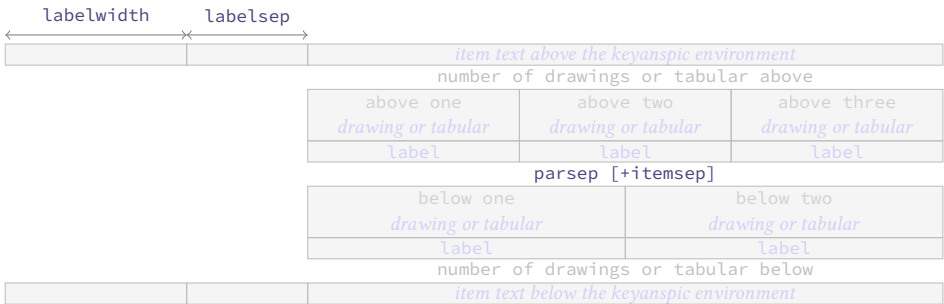


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

4.6.1 The command `\anspic`

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

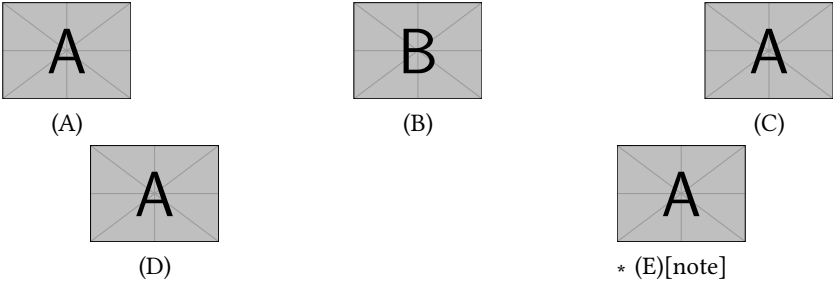
The `\anspic` command take three arguments, the *starred version* ‘`*`’ store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command `\getkeyans`

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

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

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

4.7.2 The command `\printkeyans`

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

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

(a) \LaTeXe is cool?

[2] Very True!
3. Related to Linux

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

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

1. $3(x + y + z)$ *
2. (a) Very True! *
3. (a) Yes *
- (b) i. very good *
- ii. obsolete *


5 Full examples

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

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

and then you can use the excellent `arara`¹ tool to compile them.

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

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

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

B

C

D

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


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

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

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

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

Example 2

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

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

B

C

D

36 km/h.

360 km/h.

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

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

B

C

D

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

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

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

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

B

C

D

36 km/h.

360 km/h.

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

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

B

C

D

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

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

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

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

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

*

*

*

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

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

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

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 .

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

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 2024-05-17 – First public release.

9 Index of Documentation

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

C

Document class:

article 2

book 2

exam 3

letter 2

report 2

\columnbreak 5

\columnsep 9

Commands provide by enumext:

\anskey 4, 10, 11

\anspic* 4, 10–13

\anspic 10, 12, 13

\getkeyans 4, 11, 13

\item* 4–7, 10–12

\item 6, 7, 9–11

\miniright 4, 5, 10

\printkeyans 4, 6, 11, 13

\setenumext 4, 6, 7, 10, 12, 14

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

enumext* 4, 5, 10

enumext 4–6, 9–14, 17

keyans* 4, 5, 10

keyanspic 4, 7, 10–13, 17

keyans 4–12, 17

Environments:

enumerate 1, 3, 4, 6, 19

list 4, 9, 19

minipage 3–5, 9, 10, 19

multicols 3, 5, 9

I

\item 4, 5

\itemsep 8

K

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 18

before* 9

before 9

below* 8

below 8

check-ans 11

columns-sep 5, 9

columns 5, 8, 9

first 9

font 7

item-pos* 6

item-sym* 6

itemindent 8

itemsep 8, 12

labelsep 4, 6–10, 18

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

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

list-indent 4, 8, 9

list-offset 4, 8, 18

listparindent 8

mark-ans 10

mark-pos 10

mark-ref 11

mini-env 5, 8, 9

mini-sep 5, 9

miniright* 10

miniright 10

no-store 11

noitemsep 8

nosep 8, 17

parsep 8, 12

partopsep 8

ref 5, 7

resume 9

rightmargin 8

save-ans 5, 9–13

save-ref 5, 7, 11, 13

save-sep 10

show-ans 10, 11

show-length 7

show-pos 10, 11, 13

start 9

topsep 8

widest 7

wrap-ans 10

wrap-label* 7, 18

wrap-label 7

wrap-opt 10

L

\label 5

Labels provide by enumext:

\Alph* 7, 12

\Roman* 7

\alph* 7

\arabic* 7

\roman* 7

\labelsep 4, 7

\labelwidth 4, 7

\linewidth 9

\listparindent 8

P

Packages:

enumerate 18

enumext 1–4, 13, 18, 19

enumitem 4, 5, 9, 18, 19

footnotehyper 5

hyperref 5, 11, 19

l3prop 1, 19

l3seq 1, 19

©2024 by Pablo González L

20 / 120

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   {2024-05-17}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the `multicol` package is loaded, if not we load it.

```
9 \hook_gput_code:nnn {begindocument} {enumext}
10 {
11   \IfPackageLoadedTF { multicol }
12   {
13     \msg_info:nnn { enumext } { package-load } { multicol }
14   }
15   {
16     \msg_info:nnn { enumext } { package-not-load } { multicol }
17     \RequirePackage{multicol}[2023-03-30]
18   }
19 }
```

10.4 Definition of variables

Variables that do not appear in this section are created by means of `\keys_define:nn` or some function described below.

Integer variables will control the nesting levels of the environments and boolean variables will be used to determine if they are present (nested) in each other. The boolean variables `\g__enumext_starred_bool` and `\g__enumext_standar_bool` will be set to “true” when the `enumext` and `enumext*` environments are not nested with each other.

```
20 \int_new:N \__enumext_level_int
21 \int_new:N \__enumext_level_h_int
22 \int_new:N \__enumext_keyans_level_int
23 \int_new:N \__enumext_keyans_level_h_int
24 \int_new:N \__enumext_keyans_pic_level_int
25 \bool_new:N \__enumext_starred_bool
26 \bool_new:N \g__enumext_starred_bool
```

```

27 \bool_new:N \l__enumext_starred_level_one_bool
28 \bool_new:N \l__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_level_one_bool
31 \bool_new:N \l__enumext_keyans_env_bool

```

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

```

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

```

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

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

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

```

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

```

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

```

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

```

The boolean variable `\l__enumext_resume_bool` is used by `resume` key, the value from which the environment’s will start is stored in the integer variable `\g__enumext_resume_int` (§?). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.27).

```

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

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

44 \dim_new:N \l__enumext_current_widest_dim
45 \tl_new:N \g__enumext_counter_styles_tl
46 \tl_new:N \g__enumext_widest_label_tl
47 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

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

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§10.31) which determines the internal values for `\leftmargin` and `\itemindent`.

```

48 \cs_set_protected:Npn \__enumext_tmp:n #1
49 {
50   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
51   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
52   \dim_new:c { l__enumext_leftmargin_#1_dim }
53   \dim_new:c { l__enumext_itemindent_#1_dim }
54 }
55 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

```

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

```

56 \cs_set_protected:Npn \__enumext_tmp:n #1
57 {
58   \skip_new:c { l__enumext_multicols_above_#1_skip }
59   \skip_new:c { l__enumext_multicols_below_#1_skip }
60 }
61 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

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

```

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

```

62 \int_new:N \g__enumext_minipage_stat_int
63 \skip_new:N \l__enumext_minipage_left_skip
64 \skip_new:N \l__enumext_minipage_right_skip
65 \skip_new:N \l__enumext_minipage_after_skip
66 \skip_new:N \g__enumext_minipage_right_skip
67 \skip_new:N \g__enumext_minipage_after_skip
68 \cs_set_protected:Npn \__enumext_tmp:n #1
69 {
70   \dim_new:c { \l__enumext_minipage_left_#1_dim }
71   \bool_new:c { \l__enumext_minipage_active_#1_bool }
72 }
73 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.25) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.29.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§10.35.1) for the `keyanspic` environment.

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

```

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

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

Internal variables used by the command `\setenumext` (§10.40).

```

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

```

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

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

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

```
102 \cs_set_protected:Npn \l__enumext_tmp:n #1
103 {
104   \tl_new:c { \l__enumext_store_opt_#1_tl }
105   \tl_new:c { \l__enumext_print_keyans_#1_tl }
106   \bool_new:c { \l__enumext_store_columns_#1_bool }
107   \int_new:c { \l__enumext_store_columns_#1_int }
108   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
109   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
110   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
111 }
112 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

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

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

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

```
113 \bool_new:N \l__enumext_show_answer_bool
114 \bool_new:N \l__enumext_show_position_bool
115 \tl_new:N \l__enumext_mark_ref_sym_tl
116 \tl_new:N \l__enumext_mark_answer_sym_tl
117 \str_new:N \l__enumext_mark_position_str
```

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

```
\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip
```

Internal variables used by `keyanspic` environment (§10.35.2).

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

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

```
\l__enumext_store_ans_bool
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int
```

Internal variables used by “check answer” mechanism (§10.23) controlled by the `check-ans` and `no-store` keys.

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

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§10.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with `key hyperfootnotes=true`.

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

(End of definition for `\l__enumext_hyperref_bool` and `\l__enumext_footnotes_key_bool`.)

```
\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_store_write_aux_file_tl
\l__enumext_label_copy_X_tl
```

Internal variables are used when executing the `save-ref` key. The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the labels defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` and the variable `\l__enumext_store_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

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

```

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

```

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

`\g__enumext_footnote_int`

Internal variables used for redefinition of `\footnote`.

`\g__enumext_footnote_arg_seq`

```

145 \int_new:N \g__enumext_footnote_int

```

`\g__enumext_footnote_int_seq`

```

146 \seq_new:N \g__enumext_footnote_arg_seq

```

```

147 \seq_new:N \g__enumext_footnote_int_seq

```

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

`\c__enumext_counter_style_tl`

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

`\l__enumext_ref_key_arg_tl`

```

148 \tl_const:Nn \c__enumext_counter_style_tl

```

`\l__enumext_ref_aux_tl`

```

149 { { arabic } { roman } { Roman } { alph } { Alph } }

```

`\l__enumext_the_counter_X_tl`

```

150 \tl_new:N \l__enumext_ref_key_arg_tl

```

`\l__enumext_counter_style_for_ref_X_tl`

```

151 \tl_new:N \l__enumext_ref_aux_tl

```

```

152 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

153 {

```

```

154   \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }

```

```

155   \tl_new:c { l__enumext_the_counter_#1_tl }

```

```

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

```

```

157 }

```

```

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

```

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

`\l__enumext_item_starred_X_bool`

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

`\l__enumext_item_column_pos_X_int`

```

159 \cs_set_protected:Npn \__enumext_tmp:n #1

```

`\g__enumext_item_count_all_X_int`

```

160 {

```

`\l__enumext_joined_item_X_int`

```

161   \bool_new:c { l__enumext_item_starred_#1_bool }

```

`\l__enumext_joined_item_aux_X_int`

```

162   \int_new:c { l__enumext_item_column_pos_#1_int }

```

`\l__enumext_tmpa_X_int`

```

163   \int_new:c { g__enumext_item_count_all_#1_int }

```

`\l__enumext_item_text_X_box`

```

164   \int_new:c { l__enumext_joined_item_#1_int }

```

`\l__enumext_joined_width_X_dim`

```

165   \int_new:c { l__enumext_joined_item_aux_#1_int }

```

`\l__enumext_item_width_X_dim`

```

166   \int_new:c { l__enumext_tmpa_#1_int }

```

`\g__enumext_item_symbol_aux_X_tl`

```

167   \box_new:c { l__enumext_item_text_#1_box }

```

`\l__enumext_align_label_X_str`

```

168   \dim_new:c { l__enumext_joined_width_#1_dim }

```

`\g__enumext_minipage_active_X_bool`

```

169   \dim_new:c { l__enumext_item_width_#1_dim }

```

`\g__enumext_miniright_code_X_tl`

```

170   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }

```

`\g__enumext_minipage_center_X_bool`

```

171   \str_new:c { l__enumext_align_label_#1_str }

```

`\g__enumext_minipage_right_X_dim`

```

172   \bool_new:c { g__enumext_minipage_active_#1_bool }

```

`\g__enumext_minipage_right_X_skip`

```

173   \tl_new:c { g__enumext_miniright_code_#1_tl }

```

```

174   \bool_new:c { g__enumext_minipage_center_#1_bool }

```

```

175   \dim_new:c { g__enumext_minipage_right_#1_dim }

```

```

176   \skip_new:c { g__enumext_minipage_right_#1_skip }

```

```

177 }

```

```

178 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

`\c__enumext_all_envs_clist`

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

```

179 \clist_const:Nn \c__enumext_all_envs_clist

```

```

180 {

```

```

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

```

```

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

```

```

183 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 Some utility functions

`__enumext_at_begin_document:n`

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

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

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

`__enumext_after_env:nn`

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

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

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

`__enumext_level:`

Function for check current level in `enumext`.

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

(End of definition for `__enumext_level:.`)

`__enumext_if_is_int:nT`

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

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

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

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

`__enumext_show_length:nnn`

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

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

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

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

(End of definition for `__enumext_zero_count_level:.`)

`__enumext_current_env_set_bool:`

The function `__enumext_current_env_set_bool:` will set the global variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` with which we will distinguish whether the environments `enumext` and `enumext*` are nested in each other. This function is passed to the `__enumext_safe_exec:` function in the definition of the `enumext` environment (pag 76) and to the `__enumext_safe_exec_vii:` function in the definition of the `enumext*` environment (pag 89).

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

```

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

```

(End of definition for `__enumext_current_env_set_bool:.`)

10.6 Copying list and minipage environments

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

(End of definition for `__enumext_minipage:w` and `__enumext_endminipage:.`)

10.7 Compatibility with hyperref and footnotehyper

First we define the necessary rules using “hooks” to determine if the `hyperref` package is loaded.

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

```
\__enumext_after_hyperref:
\__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”.

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

If the state of the variable `\l__enumext_footnotes_key_bool` is true we will check if the package `footnotehyper` is loaded, in case it is not present, we will set the value of `\l__enumext_footnotes_key_bool` to false and we will redefine `\footnote`.

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

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

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

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

```
\__enumext_newlabel:nn
```

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

#1: `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

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

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

```

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

```

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

10.8 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

#2: The counter’s name.

```

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

```

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

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

```

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

```

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

10.9 Definition of labels

This part of the code is inspired by the `enumitem` package. The idea is to be able to access the counters using `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

```
\__enumext_register_counter_style:Nn
```

These `⟨counters⟩` will be used as default `⟨labels⟩` if the `label` key is not used for the different levels of the `enumext` environment and the `keyans` environment, so it is necessary to get a default value for `labelwidth` from these `⟨labels⟩` at the same time.

```

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

```

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

```

\__enumext_label_width_by_box:Nn
\__enumext_label_width_by_box:cv

```

The function `__enumext_label_width_by_box:Nn` set the default `\labelwidth` using a box width if no `labelwidth` key is passed.

```

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

```

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

```

\__enumext_label_style:Nnn
\__enumext_label_style:cvn

```

The function `__enumext_label_style:Nnn` is used by the `label` key to creates the variables containing the `<label style>` and will allow to use `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` as arguments. It loops through the defined counter styles in `\g__enumext_counter_styles_tl` (`\arabic`, `\alph`, `\Alph`, `\roman`, and `\Roman`) for example, looking for `\roman*` and replacing that by `\roman{<counter>}`, and doing the same for the `\g__enumext_widest_label_tl` to keep both in sync.

```

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

```

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

10.10 Setting keys associated with label

```

font
labelsep
labelwidth
wrap-label
wrap-label*

```

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

```

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

```

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

🔗 In this point, the following are set `__enumext_wrapper_label_X:n` which will be used by `__enumext_make_label`: for the different levels of the `enumext` environment and is set to `__enumext_wrapper_label_v:n` which will be used by `__enumext_keyans_make_label`: for `keyans` and `keyanspic` environments.

`align`

The `align` key is implemented differently for “starred” and “non starred” environments.

```

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

```

```

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

```

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

```

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

```

(End of definition for `align`.)

10.11 Setting label and ref keys

`__enumext_regex_label_ref_key:`

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

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

```

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

```

(End of definition for `__enumext_regex_label_ref_key:`.)

`__enumext_set_label_ref:n`

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

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

```

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

```



```

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

```

(End of definition for `__enumext_set_label_ref:n`)

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

```

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

```

(End of definition for `__enumext_use_key_ref:`)

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

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

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

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

```

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

```

(End of definition for `__enumext_set_label_ref_h:n`)

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

```

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

```

```

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

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

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

```

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

```

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

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

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

```

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

```

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

10.11.3 Define and set label key for keyans and keyanspic environment

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

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

```

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

```

```

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

```

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

10.12 Setting start and widest keys

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

```

#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3: ⟨integer or string⟩

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

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

```

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

```

```

574 \keys_define:nn { enumext / #1 }
575 {
576   start .code:n = {
577     \__enumext_start_from:ccn
578     { l__enumext_label_#2_tl }
579     { l__enumext_start_#2_int } {##1}
580   },
581   start .initial:n = 1,
582   widest .code:n = {
583     \__enumext_widest_from:nccn {#2}
584     { l__enumext_label_#2_tl }
585     { l__enumext_labelwidth_#2_dim } {##1}
586   },
587   widest .value_required:n = true,
588   start .value_required:n = true,
589 }
590 }
591 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.13 Setting keys for vertical spaces

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

```

topsep 592 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
partopsep 593 {
parsep 594 \keys_define:nn { enumext / #1 }
noitemsep 595 {
nosep 596   topsep .skip_set:c = { l__enumext_topsep_#2_skip },
597   topsep .initial:n = {#3},
598   topsep .value_required:n = true,
599   partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
600   partopsep .initial:n = {#4},
601   partopsep .value_required:n = true,
602   parsep .skip_set:c = { l__enumext_parsep_#2_skip },
603   parsep .initial:n = {#5},
604   parsep .value_required:n = true,
605   itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
606   itemsep .initial:n = {#6},
607   itemsep .value_required:n = true,
608   noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
609   noitemsep .value_forbidden:n = true,
610   nosep .meta:n = {
611     itemsep = 0pt, parsep = 0pt,
612     topsep = 0pt, partopsep = 0pt,
613   },
614   nosep .value_forbidden:n = true,
615 }
616 }

```

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

```

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

```

(End of definition for topsep and others.)

10.14 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```

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

```

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

For `enumext*` and `keyans*` environments the situation is a bit different, the `list-indent` key behaves like the `list-offset` key.

```

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

```

10.14.1 Functions for setting the fake itemindent

__enumext_fake_item:
__enumext_keyans_fake_item:
__enumext_fake_item_vii:
__enumext_fake_item_viii:

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `0pt`. Here I will need to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

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

```

```

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

```

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

10.15 Setting show-length key

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

```

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

```

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

10.16 Setting before, after and first keys

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

```

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

```

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

10.16.1 Functions for before, after and first keys in enumext

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

```

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

```

The function `__enumext_before_keys_exec`: executes the $\{\langle code \rangle\}$ 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 $\langle keys \rangle$.

```

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

```

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

```

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

```

The function `__enumext_after_args_exec`: executes the $\{\langle code \rangle\}$ 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`.

```

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

```

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

10.16.2 Functions for before, after and first keys in keyans

`__enumext_before_args_exec_v`: The function `__enumext_before_args_exec_v`: executes the $\{\langle code \rangle\}$ 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.

```

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

```

The function `__enumext_before_keys_exec_v`: executes the $\{\langle code \rangle\}$ 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 $\langle keys \rangle$.

```

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

```

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

```

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

```

The function `__enumext_after_args_exec_v`: executes the $\{\langle code \rangle\}$ 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`.

```

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

```

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

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

`__enumext_before_args_exec_vii`: The function `__enumext_before_args_exec_v`: executes the $\{\langle code \rangle\}$ 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.

```

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

```


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

```

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

```

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

```

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

```

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

```

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

```

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

10.17 Setting keys for multicol and minipage

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

```

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

```

For `enumext*` and `keyans*` environments the situation is a bit different, the default value for `columns` key are 2 and the command `\miniright` is not available, so we will add the keys `miniright` and `miniright*` to implement support for `minipage`.

```

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

```

```

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

```

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

10.18 Adjustment of vertical spaces for multicol

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

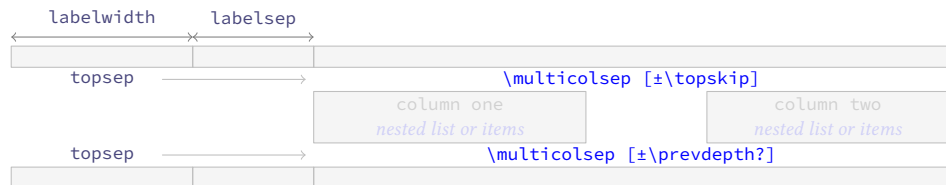


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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

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

10.18.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

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

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

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

```

```

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

```

(End of definition for `__enumext_add_pre_parsep:`)

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

```

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

```

(End of definition for `__enumext_multi_addvspace:`)

10.18.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

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

```

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

10.19 Adjustment of vertical spaces for minipage

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

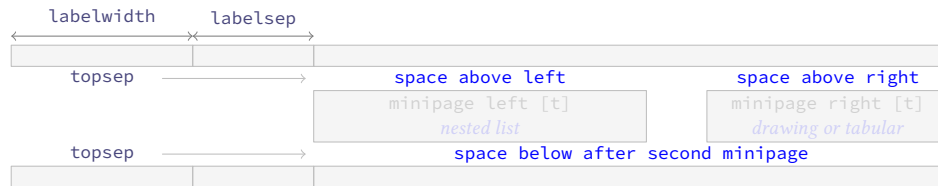


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

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

```

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

(End of definition for `__enumext_mini_env*`.)

10.19.1 Adjustment of vertical spaces for minipage in enumext

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

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

First determine if the `multicols` environment is active by comparing the value of the `\l__enumext_columns_X_int` variable handled by the `columns` key, according to this comparison we set the adjusted values for `\l__enumext_minipage_left_skip`, `\l__enumext_minipage_right_skip` and `\l__enumext_minipage_after_skip`.

```

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

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

```

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

```

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

```

If only `enumext` environment is nested in `__enumext_mini_env*` environment, we will apply a correction factor to the *vertical spaces* taking into account the value of `\topsep`, if this is zero we will use `\strutbox` as the basis for the calculations.

```

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

```

(End of definition for `__enumext_mini_set_vskip:`)

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

```

995 \cs_new_protected:Nn \__enumext_zero_parsep:
996 {

```

```

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

```

(End of definition for `__enumext_zero_parsep:`)

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

```

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

```

(End of definition for `__enumext_mini_addvspace:`)

10.19.2 Adjustment of vertical spaces for minipage in keyans

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

```

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

```

```

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

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

The function `__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `__enumext_mini_env*` environment in `keyans`, taking into account whether \TeX is in $\langle horizontal\ mode\rangle$ 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`.

```

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

```

(End of definition for `__enumext_keyans_mini_addvspace:`)

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

`__enumext_mini_set_vskip_vii:`
`__enumext_mini_set_vskip_viii:`

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

```

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

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

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

```

1172 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:

```

```

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

```

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

10.19.4 The command \miniright

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

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

```

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

```

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

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

```

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

```

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

`__enumext_keyans_mini_right_cmd:n`

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

```

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

```

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

10.20 Setting above and below keys

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

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

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

```

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

```

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

10.20.1 Functions for above and below keys in enumext

`__enumext_vspace_above:`

The function `__enumext_vspace_above:` apply the *vertical space above* the `enumext` environment set by the `above*` and `above` keys.

```

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

```

(End of definition for `__enumext_vspace_above:`.)

`__enumext_vspace_below:` The function `__enumext_vspace_below:` apply the *vertical space below* the `enumext` environment set by the `below*` and `below` keys.

```

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

```

(End of definition for `__enumext_vspace_below:`.)

10.20.2 Functions for above and below keys in keyans

`__enumext_vspace_above_v:` The function `__enumext_vspace_above_v:` apply the *vertical space above* the `keyans` environment set by the `above` and `above*` keys.

```

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

```

(End of definition for `__enumext_vspace_above_v:`.)

`__enumext_vspace_below_v:` The function `__enumext_vspace_below_v:` apply the *vertical space below* the `keyans` environment set by the `below*` and `below` keys.

```

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

```

(End of definition for `__enumext_vspace_below_v:`.)

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

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

`__enumext_vspace_above_viii:`

```

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

```

```

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

```

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

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

```

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

```

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

10.21 Setting series, resume and resume* keys

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

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

```

series
resume
resume*
1347 \cs_set_protected:Npn \__enumext_tmp:n #1
1348 {
1349     \keys_define:nn { enumext / #1 }
1350     {
1351         series .str_set:N = \l__enumext_series_str,
1352         series .value_required:n = true,
1353         resume .code:n = \__enumext_resume_series:n {##1},
1354         resume* .code:n = \__enumext_resume_starred:,
1355         resume* .value_forbidden:n = true,
1356     }
1357 }
1358 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.21.1 Internal functions for series key

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

```

1359 \cs_new:Npn \__enumext_filter_series:n #1
1360 {
1361     \use:e
1362     {
1363         \keyval_parse:NNn
1364         \__enumext_filter_series_key:n
1365         \__enumext_filter_series_pair:nn {#1}
1366     }
1367 }

```

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.

```

1368 \cs_new:Npn \__enumext_filter_series_key:n #1
1369 {
1370   \str_case:nnF {#1}
1371   {
1372     { resume } {}
1373     { resume* } {}
1374   }
1375   { , { \exp_not:n {#1} } }
1376 }

```

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.

```

1377 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1378 {
1379   \str_case:nnF {#1}
1380   {
1381     { series } {}
1382     { resume } {}
1383     { save-ans } {}
1384     { save-key } {}
1385   }
1386   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1387 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§10.33) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§10.36).

```

1388 \cs_new_protected:Npn \__enumext_parse_series:n #1
1389 {
1390   \str_if_empty:NTF \l__enumext_series_str
1391   {
1392     \bool_if:NF \l__enumext_resume_active_bool
1393     {
1394       \__enumext_resume_last:n {#1}
1395     }
1396   }
1397   {
1398     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1399     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1400     { \__enumext_filter_series:n {#1} }
1401     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1402     {
1403       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1404     }
1405   }
1406 }

```

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.

```

1407 \cs_new_protected:Npn \__enumext_resume_last:n #1
1408 {
1409   \bool_if:NT \l__enumext_standar_level_one_bool
1410   {
1411     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT]]}
1412     \tl_gclear:N \g__enumext_standar_series_tl
1413     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }

```

```

1414     }
1415     \bool_if:NT \l__enumext_starred_level_one_bool
1416     {
1417         %%\typeout{[[ON-LEVEL-ONE-ENUMEXT*]]}
1418         \tl_gclear:N \g__enumext_starred_series_tl
1419         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1420     }
1421 }

```

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

10.21.2 Internal function to save counter value

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

• The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments. The integer variable with `\l__enumext_series_str` sets the value when we execute the `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when we use the `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§10.33) and the `enumext*` environment definition (§10.36).

```

1422 \cs_new_protected:Nn \__enumext_resume_save_counter:
1423 {
1424     \bool_if:NT \g__enumext_standar_bool
1425     {
1426         \tl_if_empty:NF \l__enumext_series_str
1427         {
1428             \int_gset_eq:cN
1429             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1430         }
1431         \tl_if_empty:NTF \l__enumext_resume_name_tl
1432         {
1433             \str_if_empty:NT \l__enumext_series_str
1434             {
1435                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1436             }
1437         }
1438         {
1439             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1440             {
1441                 \int_gset_eq:cN
1442                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1443             }
1444         }
1445         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1446         {
1447             \int_gset_eq:cN
1448             { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1449         }
1450     }
1451     \bool_if:NT \g__enumext_starred_bool
1452     {
1453         \tl_if_empty:NF \l__enumext_series_str
1454         {
1455             \int_gset_eq:cN
1456             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1457         }
1458         \tl_if_empty:NTF \l__enumext_resume_name_tl
1459         {
1460             \str_if_empty:NT \l__enumext_series_str
1461             {
1462                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1463             }
1464         }
1465         {
1466             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1467             {
1468                 \int_gset_eq:cN
1469                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}

```



```

1470         }
1471     }
1472     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1473     {
1474         \int_gset_eq:cN
1475         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1476     }
1477 }
1478 }

```

(End of definition for __enumext_resume_save_counter:.)

10.21.3 Internal functions for resume key

__enumext_resume_series:n

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

```

1479 \cs_new_protected:Npn \__enumext_resume_series:n #1
1480 {
1481     \tl_if_empty:NTF {#1}
1482     {
1483         \__enumext_resume_counter:n { }
1484     }
1485     {
1486         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1487         {
1488             \__enumext_resume_counter:n {#1}
1489             \bool_if:NT \g__enumext_standar_bool
1490             {
1491                 \keys_set:nv { enumext / level-1 }
1492                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1493             }
1494             \bool_if:NT \g__enumext_starred_bool
1495             {
1496                 \keys_set:nv { enumext / enumext* }
1497                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1498             }
1499         }
1500         {
1501             \bool_if:NT \g__enumext_standar_bool
1502             {
1503                 \msg_error:nnn { enumext } { unknown-series } {#1}
1504             }
1505             \bool_if:NT \g__enumext_starred_bool
1506             {
1507                 \msg_error:nnn { enumext } { unknown-series } {#1}
1508             }
1509         }
1510     }
1511 }

```

(End of definition for __enumext_resume_series:n.)

__enumext_resume_counter:n

__enumext_resume_counter:

__enumext_resume_counter_series:

__enumext_resume_counter_save_ans:

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

```

1512 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1513 {
1514     \bool_set_true:N \l__enumext_resume_active_bool
1515     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1516     \tl_if_empty:NTF \l__enumext_resume_name_tl
1517     {
1518         \__enumext_resume_counter:

```

```

1519     }
1520     {
1521         \__enumext_resume_counter_series:
1522     }
1523     \__enumext_resume_counter_save_ans:
1524 }

```

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

```

1525 \cs_new_protected:Nn \__enumext_resume_counter:
1526 {
1527     \bool_if:NT \g__enumext_standar_bool
1528     {
1529         \int_gincr:N \g__enumext_resume_int
1530         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1531     }
1532     \bool_if:NT \g__enumext_starred_bool
1533     {
1534         \int_gincr:N \g__enumext_resume_vii_int
1535         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1536     }
1537 }

```

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

```

1538 \cs_new_protected:Nn \__enumext_resume_counter_series:
1539 {
1540     \bool_if:NT \g__enumext_standar_bool
1541     {
1542         \int_set:Nn \l__enumext_start_i_int
1543         {
1544             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1545         }
1546     }
1547     \bool_if:NT \g__enumext_starred_bool
1548     {
1549         \int_set:Nn \l__enumext_start_vii_int
1550         {
1551             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1552         }
1553     }
1554 }

```

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

```

1555 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1556 {
1557     \bool_lazy_and:nnT
1558     { \bool_if_p:N \l__enumext_standar_level_one_bool }
1559     { \bool_if_p:N \l__enumext_store_active_bool }
1560     {
1561         \int_set:Nn \l__enumext_start_i_int
1562         {
1563             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1564         }
1565     }
1566     \bool_lazy_and:nnT
1567     { \bool_if_p:N \l__enumext_starred_level_one_bool }
1568     { \bool_if_p:N \l__enumext_store_active_bool }
1569     {
1570         \int_set:Nn \l__enumext_start_vii_int
1571         {
1572             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1573         }
1574     }
1575 }

```

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

10.21.4 Internal function for resume* key

\\enumext_resume_starred:

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

```

1576 \cs_new_protected:Nn \\enumext_resume_starred:
1577 {
1578   \bool_if:NT \g__enumext_standar_bool
1579   {
1580     \tl_if_empty:NF \g__enumext_standar_series_tl
1581     {
1582       \\enumext_resume_counter:n { }
1583       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1584     }
1585   }
1586   \bool_if:NT \g__enumext_starred_bool
1587   {
1588     \tl_if_empty:NF \g__enumext_starred_series_tl
1589     {
1590       \\enumext_resume_counter:n { }
1591       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1592     }
1593   }
1594 }

```

(End of definition for \\enumext_resume_starred:.)

10.22 Setting save-ans key

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

save-ans

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

```

1595 \keys_define:nn { enumext / level-1 }
1596 {
1597   save-ans .code:n = \\enumext_storing_set:n {#1},
1598   save-ans .value_required:n = true,
1599 }
1600 \keys_define:nn { enumext / enumext* }
1601 {
1602   save-ans .code:n = \\enumext_storing_set_vii:n {#1},
1603   save-ans .value_required:n = true,
1604 }

```

(End of definition for save-ans.)

10.22.1 Internal functions for save-ans key

\\enumext_storing_set:n
 \\enumext_storing_set_vii:n
 \\enumext_storing_standar:
 \\enumext_storing_starred:
 \\enumext_storing_exec:

The functions \\enumext_storing_set:n and \\enumext_storing_set_vii:n first pass the value of the *save-ans* key to the variable \\l__enumext_store_name_tl which will contain the “store name” of the *(sequence)* and *(prop list)* we will use. If \\l__enumext_store_name_tl is empty we return an error message, otherwise we proceed to execute the functions \\enumext_storing_standar: for the *enumext* environment and \\enumext_storing_starred: for the *enumext** environment.

```

1605 \cs_new_protected:Npn \\enumext_storing_set:n #1
1606 {
1607   \tl_set:Ne \\l__enumext_store_name_tl {#1}
1608   \tl_if_empty:NTF \\l__enumext_store_name_tl
1609   {
1610     \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1611   }
1612   {
1613     \\enumext_storing_standar:
1614   }
1615 }
1616 \cs_new_protected:Npn \\enumext_storing_set_vii:n #1
1617 {
1618   \tl_set:Ne \\l__enumext_store_name_tl {#1}
1619   \tl_if_empty:NTF \\l__enumext_store_name_tl
1620   {
1621     \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1622   }

```

```

1623     {
1624         \__enumext_storing_starred:
1625     }
1626 }

```

The functions `__enumext_storing_standar:` and `__enumext_storing_starred:` will verify the state of the variables `\g__enumext_standar_level_one_bool` and `\g__enumext_starred_level_one_bool`, if this is true we execute the function `__enumext_storing_exec:` otherwise we return an warning message.

```

1627 \cs_new_protected:Nn \__enumext_storing_standar:
1628 {
1629     \bool_if:NTF \l__enumext_standar_level_one_bool
1630     {
1631         \__enumext_storing_exec:
1632     }
1633     {
1634         \msg_warning:nnn { enumext } { save-ans-nested } { enumext }
1635     }
1636 }
1637 \cs_new_protected:Nn \__enumext_storing_starred:
1638 {
1639     \bool_if:NTF \l__enumext_starred_level_one_bool
1640     {
1641         \__enumext_storing_exec:
1642     }
1643     {
1644         \msg_warning:nnn { enumext } { save-ans-nested } { enumext* }
1645     }
1646 }

```

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

```

1647 \cs_new_protected:Nn \__enumext_storing_exec:
1648 {
1649     \bool_set_true:N \l__enumext_store_active_bool
1650     \bool_set_true:N \l__enumext_store_ans_bool
1651     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1652     {
1653         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1654     }
1655     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1656     {
1657         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1658     }
1659     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1660     {
1661         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1662     }
1663 }

```

(End of definition for `__enumext_storing_set: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
1664 \cs_set_protected:Npn \__enumext_tmp:n #1
1665 {
1666   \keys_define:nn { enumext / #1 }
1667   {
1668     check-ans .bool_set:N = \__enumext_check_ans_bool,
1669     check-ans .initial:n = false,
1670     no-store .code:n = {
1671       \bool_set_false:N \__enumext_store_ans_bool
1672       \bool_set_false:N \__enumext_check_ans_bool
1673     },
1674     no-store .value_forbidden:n = true,
1675   }
1676 }
1677 \clist_map_inline:nn
1678 {
1679   level-1, level-2, level-3, level-4, enumext*
1680 }
1681 { \__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.

```
1682 \cs_new_protected:Nn \__enumext_check_ans_set:
1683 {
1684   \int_case:nn { \__enumext_level_int }
1685   {
1686     { 1 }{
1687       \bool_lazy_all:nT
1688       {
1689         { \bool_if_p:N \g__enumext_starred_bool }
1690         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1691       }
1692       {
1693         \int_gdecr:N \g__enumext_count_item_number_int
1694         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1695       }
1696     }
1697     { 2 }{
1698       \int_gdecr:N \g__enumext_count_item_number_int
1699     }
1700     { 3 }{
1701       \int_gdecr:N \g__enumext_count_item_number_int
1702     }
1703     { 4 }{
1704       \int_gdecr:N \g__enumext_count_item_number_int
1705     }
1706   }
1707   \int_case:nn { \__enumext_level_h_int }
```

```

1708     {
1709         { 1 }{
1710             \bool_if:NT \g__enumext_standar_bool
1711             {
1712                 \int_gdecr:N \g__enumext_count_item_number_int
1713                 \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1714             }
1715         }
1716     }
1717 }

```

(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.

```

1718 \cs_new_protected:Nn \__enumext_check_ans_exec:
1719 {
1720     \bool_if:NT \l__enumext_check_ans_bool
1721     {
1722         \__enumext_check_ans_set:
1723     }
1724 }

```

(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.

```

1725 \cs_new_protected:Nn \__enumext_check_ans_show:
1726 {
1727     \int_compare:nNnTF
1728     { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1729     {
1730         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1731     }
1732     {
1733         \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1734     }
1735     \int_gzero:N \g__enumext_count_item_number_int
1736     \int_gzero:N \g__enumext_count_item_anskey_int
1737 }

```

(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*`.

```

1738 \cs_set_protected:Npn \__enumext_tmp:n #1
1739 {
1740     \keys_define:nn { enumext / #1 }
1741     {
1742         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1743         wrap-ans .initial:n = \fbox{##1},
1744         wrap-ans .value_required:n = true,
1745         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1746         wrap-opt .initial:n = [{##1}],
1747         wrap-opt .value_required:n = true,
1748         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1749         save-sep .initial:n = {, ~ },
1750         save-sep .value_required:n = true,
1751         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1752         mark-ans .initial:n = \textasteriskcentered,
1753         mark-ans .value_required:n = true,
1754         mark-pos .choice:,
1755         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1756         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1757         mark-pos .initial:n = right,
1758         mark-pos .value_required:n = true,

```

```

1759     show-ans .bool_set:N = \__enumext_show_answer_bool,
1760     show-ans .initial:n = false,
1761     show-ans .value_required:n = true,
1762     show-pos .bool_set:N = \__enumext_show_position_bool,
1763     show-pos .initial:n = false,
1764     show-pos .value_required:n = true,
1765     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
1766     mark-ref .initial:n = \textasteriskcentered,
1767     mark-ref .value_required:n = true,
1768     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
1769     save-ref .initial:n = false,
1770     save-ref .value_required:n = true,
1771   }
1772 }
1773 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1774 \cs_set_protected:Npn \__enumext_tmp:n #1
1775 {
1776   \keys_define:nn { enumext / #1 }
1777   {
1778     mark-pos .choice:,
1779     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1780     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1781     mark-pos .initial:n = right,
1782     mark-pos .value_required:n = true,
1783     show-ans .bool_set:N = \__enumext_show_answer_bool,
1784     show-ans .initial:n = false,
1785     show-ans .value_required:n = true,
1786     show-pos .bool_set:N = \__enumext_show_position_bool,
1787     show-pos .initial:n = false,
1788     show-pos .value_required:n = true,
1789   }
1790 }
1791 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1792 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1793 {
1794   \keys_define:nn { enumext / #1 }
1795   {
1796     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1797               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1798               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1799               {
1800                 columns = \exp_not:v { l__enumext_store_columns_#2_int },
1801               },
1802     columns* .value_required:n = true,
1803     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1804               \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1805               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1806               {
1807                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1808               },
1809     columns-sep* .value_required:n = true,
1810   }
1811 }
1812 \clist_map_inline:nn
1813 {
1814   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1815 }
1816 { \__enumext_tmp:nn #1 }

```

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

10.24.1 Function for storing content in prop list

```
\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V
```

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

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

```
1817 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1818 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1819 {
1820   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1821   {
1822     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1823   }
1824   { #1 }
1825 }
1826 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

10.24.2 Function for storing content in sequence

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

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

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

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

```
1827 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1828 {
1829   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1830 }
1831 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

10.24.3 Functions for storing the list structure in the sequence

```
\__enumext_store_level_open:
\__enumext_store_level_close:
```

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

```
1832 \cs_new_protected:Nn \__enumext_store_level_open:
1833 {
1834   \bool_if:NT \l__enumext_store_ans_bool
1835   {
1836     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1837     {
1838       \__enumext_store_addto_seq:n
1839       {
1840         \item \begin{enumext}
1841       }
1842     }
1843     {
1844       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1845       {
1846         \item \begin{enumext} [
1847       }
1848       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1849       {
1850         ]
1851       }
1852       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1853     }
1854   }
1855 }
1856 \cs_new_protected:Nn \__enumext_store_level_close:
1857 {
1858   \bool_if:NT \l__enumext_store_ans_bool
1859   {
```

```

1860         \\\enumext_store_addto_seq:n { \end{enumext} }
1861     }
1862 }

```

(End of definition for \\\enumext_store_level_open: and \\\enumext_store_level_close:.)

```

\\enumext_store_level_open_vii:
\\enumext_store_level_close_vii:

```

When nesting the `enumext*` environment in `enumext` starting right after `\item` (without material between them) there is a problem with the alignment of the labels with the baseline between the two environments. One way to get around this problem is to place `\mode_leave_vertical:` and then apply `\vspace` taking into account `\baselineskip`, the value of `\parsep` of the current level of `enumext` and the value of `\topsep` of the `enumext*` environment.

```

1863 \cs_new_protected:Nn \\\enumext_store_level_open_vii:
1864 {
1865     \bool_if:NT \l__enumext_store_ans_bool
1866     {
1867         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1868         {
1869             \\\enumext_store_addto_seq:n
1870             {
1871                 \item \mode_leave_vertical:
1872                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1873                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1874             }
1875         }
1876         {
1877             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1878             {
1879                 \item \mode_leave_vertical:
1880                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1881                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1882                 ]
1883             }
1884             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1885             {
1886             }
1887             \\\enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1888         }
1889     }
1890 }
1891 \cs_new_protected:Nn \\\enumext_store_level_close_vii:
1892 {
1893     \bool_if:NT \l__enumext_store_ans_bool
1894     {
1895         \\\enumext_store_addto_seq:n { \end{enumext*} }
1896     }
1897 }

```

(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`

```

1898 \cs_new_protected:Nn \\\enumext_print_keyans_box:NN
1899 {
1900     \mode_leave_vertical:
1901     \skip_horizontal:n { -\dim_use:N #2 }
1902     \makebox[\opt][ r ]
1903     {
1904         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1905         {
1906             \tl_use:N \l__enumext_mark_answer_sym_tl
1907         }
1908     }
1909     \skip_horizontal:n { \dim_use:N #2 }
1910 }
1911 \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 `enumxt*` environments.

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

```

1912 \keys_define:nn { enumext / anskey }
1913 {
1914   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1915   item-sym* .value_required:n = true,
1916   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1917   item-pos* .value_required:n = true,
1918   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1919   store-brk .default:n = true,
1920   store-brk .value_forbidden:n = true,
1921 }

```

This command `\anskey` will only be present when using the `save-ans` key in `enumext` and `enumxt*` 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`.

```

1922 \NewDocumentCommand \anskey { d() s o +m }
1923 {
1924   \bool_if:NF \l__enumext_store_active_bool
1925   {
1926     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1927   }
1928   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1929   {
1930     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1931   }
1932   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1933   {
1934     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1935   }
1936   \group_begin:
1937     \bool_if:NT \l__enumext_store_ans_bool
1938     {
1939       \bool_if:NT \l__enumext_check_ans_bool
1940       {
1941         \int_gincr:N \g__enumext_count_item_anskey_int
1942       }
1943       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1944     }
1945   \group_end:
1946 }

```

(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.

```

1947 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1948 {
1949   \__enumext_store_addto_prop:n {#4}
1950   \bool_if:NT \l__enumext_store_ref_key_bool
1951   {
1952     \__enumext_store_internal_ref:
1953   }
1954   \__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 `[\langle key = val \rangle]`, 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`.

```

1955 \tl_clear:N \l__enumext_store_anskey_arg_tl
1956 \tl_if_novalue:nF {#3}
1957 {
1958   \keys_set:nn { enumext / anskey } {#3}
1959 }
1960 \bool_lazy_and:nnT
1961 { \bool_if_p:N \l__enumext_store_columns_break_bool }
1962 { \bool_not_p:n { \l__enumext_starred_bool } }
1963 {
1964   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1965 }
1966 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

1967 \tl_if_novalue:nF {#1}
1968 {
1969   \int_set:Nn \l__enumext_store_columns_join_int {#1}
1970   \bool_if:NT \l__enumext_starred_bool
1971   {
1972     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1973     {
1974       ( \exp_not:V \l__enumext_store_columns_join_int )
1975     }
1976   }
1977 }

```

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`.

```

1978 \bool_if:nTF {#2}
1979 {
1980   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1981   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1982   {
1983     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1984     {
1985       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1986     }
1987   }
1988   \dim_compare:nT
1989   {
1990     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1991   }
1992   {
1993     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1994     {
1995       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1996     }
1997   }
1998   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1999 }
2000 {
2001   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2002 }

```

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

```

2003 \bool_lazy_and:nnT
2004 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2005 { \bool_if_p:N \l__enumext_hyperref_bool }
2006 {
2007   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2008   {
2009     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2010     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2011   }
2012 }
2013 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2014 }

```

(End of definition for `\l__enumext_store_anskey_code:nnnn`.)

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

First we will remove the dots “.” from the current `⟨labels⟩`, we do not want to get double dots in our references, then we will place this in the variable `__enumext_newlabel_arg_two_tl`.

```

2015 \cs_new_protected:Nn \__enumext_store_internal_ref:
2016 {
2017   \cs_set_protected:Npn \__enumext_tmp:n ##1
2018   {
2019     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2020     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2021     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2022     \tl_reverse:c { \__enumext_label_copy_##1_tl }
2023   }
2024   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2025   \cs_set:Npn \__enumext_tmp:n ##1
2026   { . \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.

```

2027   \bool_lazy_all:nT
2028   {
2029     { \bool_if_p:N \g__enumext_starred_bool }
2030     { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
2031   }
2032   {
2033     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2034     { \tl_use:N \__enumext_label_copy_vii_tl }
2035   }
2036   \bool_lazy_all:nT
2037   {
2038     { \bool_if_p:N \l__enumext_standar_bool }
2039     { \bool_if_p:N \g__enumext_starred_bool }
2040     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2041   }
2042   {
2043     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2044     {
2045       \tl_use:N \__enumext_label_copy_vii_tl
2046       \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2047     }
2048   }

```

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.

```

2049   \bool_lazy_all:nT
2050   {
2051     { \bool_if_p:N \l__enumext_standar_bool }
2052     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2053     { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_zero_int } }
2054     { \bool_not_p:n { \__enumext_starred_bool } }
2055   }
2056   {
2057     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2058     {
2059       \tl_use:N \__enumext_label_copy_i_tl
2060       \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
2061     }
2062   }
2063   \cs_set:Npn \__enumext_tmp:n ##1
2064   { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2065   \bool_lazy_all:nT
2066   {
2067     { \bool_if_p:N \l__enumext_standar_bool }
2068     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
2069     { \bool_not_p:n { \g__enumext_starred_bool } }
2070     { \int_compare_p:nNn { \__enumext_level_h_int } > { \c_zero_int } }
2071   }
2072   {
2073     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2074     {

```

```

2075         \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2076         . \tl_use:N \l__enumext_label_copy_vii_tl
2077     }
2078 }

```

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

```

2079     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2080     {
2081         \l__enumext_store_name_tl \c_colon_str
2082         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2083     }

```

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

```

2084     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2085     {
2086         \l__enumext_newlabel:nn
2087         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2088         { \l__enumext_newlabel_arg_two_tl }
2089     }
2090     \l__enumext_store_write_aux_file_tl
2091 }

```

(End of definition for `\l__enumext_store_internal_ref:`)

`\l__enumext_store_anskey_show_wrap:n`

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

```

2092 \cs_new_protected:Npn \l__enumext_store_anskey_show_wrap:n #1
2093 {
2094     \par
2095     \bool_if:NT \l__enumext_starred_bool
2096     {
2097         \cs_set:Nn \l__enumext_level: { vii }
2098     }
2099     \l__enumext_print_keyans_box:cc
2100     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2101     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2102     \l__enumext_anskey_wrapper:n { #1 }
2103 }

```

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

`\l__enumext_store_anskey_show_left:n`

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

```

2104 \cs_new_protected:Npn \l__enumext_store_anskey_show_left:n #1
2105 {
2106     \bool_if:NT \l__enumext_show_answer_bool
2107     {
2108         \l__enumext_store_anskey_show_wrap:n { #1 }
2109     }
2110     \bool_if:NT \l__enumext_show_position_bool
2111     {
2112         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2113         {
2114             \group_begin:
2115             \exp_not:N \normalfont
2116             \exp_not:N \footnotesize [ \int_eval:n
2117             {
2118                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2119             }
2120             ]
2121             \group_end:
2122         }
2123         \l__enumext_store_anskey_show_wrap:n { #1 }
2124     }
2125 }

```

(End of definition for `\l__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 $\langle label \rangle$ `__enumext_label_v_tl` for the `keyans` environment and the current $\langle label \rangle$ `__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 `__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`.

```

2126 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2127 {
2128   \tl_clear:N \__enumext_store_keyans_label_tl
2129   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2130   {
2131     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_vi_tl }
2132   }
2133   {
2134     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_v_tl }
2135   }
2136   \tl_if_novalue:nF { #1 }
2137   {
2138     % Set save-sep
2139     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
2140     {
2141       \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
2142     }
2143     \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
2144   }
2145   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
2146 }

```

(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.

```

2147 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2148 {
2149   \bool_if:NT \__enumext_store_ref_key_bool
2150   {
2151     \cs_set_protected:Npn \__enumext_tmp:n ##1
2152     {
2153       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2154       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2155       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2156       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2157     }
2158     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2159     \__enumext_keyans_store_ref_aux_i:
2160   }
2161 }

```

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

```

2162 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2163 {
2164   \bool_if:NT \g__enumext_starred_bool
2165   {
2166     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2167   }
2168   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2169   {
2170     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl

```



```

2171         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2172     }
2173     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2174     {
2175         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2176         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2177     }
2178     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2179     {
2180         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2181         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2182     }
2183     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2184     {
2185         \l__enumext_store_name_tl \c_colon_str
2186         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2187     }
2188     \__enumext_keyans_store_ref_aux_ii:
2189 }

```

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.

```

2190 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2191 {
2192     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2193     {
2194         \__enumext_newlabel:nn
2195         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2196         { \l__enumext_newlabel_arg_two_tl }
2197     }
2198     \l__enumext_store_write_aux_file_tl
2199 }

```

(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.

```

2200 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2201 {
2202     \tl_clear:N \l__enumext_store_keyans_label_tl
2203     \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
2204     {
2205         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2206     }
2207     {
2208         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2209     }
2210     \tl_if_novalue:nF { #1 }
2211     {
2212         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2213         {
2214             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2215         }
2216         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2217     }
2218     \__enumext_keyans_addto_seq_link:
2219 }

```

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.

```

2220 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2221 {

```

```

2222 \bool_lazy_and:nnT
2223 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2224 { \bool_if_p:N \l__enumext_hyperref_bool }
2225 {
2226   \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2227   {
2228     \hfill \exp_not:N \hyperlink
2229     {
2230       \exp_not:V \l__enumext_newlabel_arg_one_tl
2231     }
2232     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2233   }
2234 }
2235 \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2236 \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2237 \bool_if:NT \l__enumext_check_ans_bool
2238 {
2239   \int_gincr:N \g__enumext_count_item_anskey_int
2240 }
2241 }

```

(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.

```

2242 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2243 {
2244   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2245   {
2246     \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2247   }
2248   { \tl_gclear:N \g__enumext_check_ans_item_tl }
2249 }

```

(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.

```

2250 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2251 {
2252   \tl_if_novalue:nF { #1 }
2253   {
2254     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2255   }
2256   \bool_if:NT \l__enumext_show_answer_bool
2257   {
2258     \__enumext_keyans_show_ans:
2259   }
2260   \bool_if:NT \l__enumext_show_position_bool
2261   {
2262     \__enumext_keyans_show_pos:
2263   }
2264 }
2265 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2266 {
2267   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2268   {
2269     \bool_lazy_or:nnT
2270     { \bool_if_p:N \l__enumext_show_answer_bool }
2271     { \bool_if_p:N \l__enumext_show_position_bool }
2272     {
2273       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2274     }
2275   }

```

```

2276   }
2277   \cs_new_protected:Nn \__enumext_keyans_show_ans:
2278   {
2279     \tl_put_left:Nn \l__enumext_label_v_tl
2280     {
2281       \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2282     }
2283   }
2284   \cs_new_protected:Nn \__enumext_keyans_show_pos:
2285   {
2286     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2287     {
2288       \tl_set:Ne \l__enumext_mark_answer_sym_tl
2289       {
2290         \group_begin:
2291         \exp_not:N \normalfont
2292         \exp_not:N \footnotesize [ \int_eval:n
2293         {
2294           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2295         }
2296         ]
2297         \group_end:
2298       }
2299     }
2300     {
2301       \tl_set:Ne \l__enumext_mark_answer_sym_tl
2302       {
2303         \group_begin:
2304         \exp_not:N \normalfont
2305         \exp_not:N \footnotesize [ \int_eval:n
2306         {
2307           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2308         }
2309         ]
2310         \group_end:
2311       }
2312     }
2313     \tl_put_left:Nn \l__enumext_label_v_tl
2314     {
2315       \__enumext_print_keyans_box:NN
2316       \l__enumext_labelwidth_i_dim
2317       \l__enumext_labelsep_i_dim
2318     }
2319   }

```

(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*
2320 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2321 {
2322   \keys_define:nn { enumext / #1 }
2323   {
2324     item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2325     item-sym* .value_required:n = true,
2326     item-sym* .initial:n = { $\star$ },
2327     item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2328     item-pos* .value_required:n = true,
2329   }
2330 }
2331 \clist_map_inline:nn
2332 {
2333   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2334 }
2335 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

2336 \cs_new_protected:Nn \__enumext_footnotetext:nn
2337 {
2338   \footnotetext[#1]{#2}
2339 }
2340 \cs_new_protected:Nn \__enumext_renew_footnote:
2341 {
2342   \seq_gclear:N \g__enumext_footnote_arg_seq
2343   \seq_gclear:N \g__enumext_footnote_int_seq
2344   \RenewDocumentCommand \footnote { o +m }
2345   {
2346     \tl_if_novalue:nTF {##1}
2347     {
2348       \stepcounter{footnote}
2349       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2350     }
2351     {
2352       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2353     }
2354     \footnotemark [ \g__enumext_footnote_int ]
2355     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2356     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2357   }
2358 }
2359 \cs_new_protected:Nn \__enumext_print_footnote:
2360 {
2361   \seq_if_empty:NF \g__enumext_footnote_int_seq
2362   {
2363     \seq_map_pairwise_function:NNN
2364     \g__enumext_footnote_int_seq
2365     \g__enumext_footnote_arg_seq
2366     \__enumext_footnotetext:nn
2367   }
2368 }

```

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

10.29 Redefining `\item` command

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

10.29.1 The `\item` command in `enumext`

```

\__enumext_default_item:n

```

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

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

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

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

```

2369 \cs_new_protected:Npn \__enumext_default_item:n #1
2370 {
2371   \tl_if_novalue:nTF {#1}
2372   {
2373     \bool_if:NT \l__enumext_check_ans_bool
2374     {
2375       \int_gincr:N \g__enumext_count_item_number_int
2376     }
2377     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2378     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2379   }
2380   {
2381     \bool_set_eq:cc
2382     { l__enumext_wrap_label_ \__enumext_level: _bool }

```

```

2383         { \__enumext_wrap_label_opt_ \__enumext_level: _bool }
2384         \__enumext_item_std:w [#1] \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl
2385     }
2386 }

```

(End of definition for __enumext_default_item:n.)

__enumext_starred_item:nn

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

#1: \l__enumext_item_symbol_X_tl

#2: \l__enumext_item_symbol_sep_X_dim

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

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

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

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

```

2387 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2388 {
2389     \tl_if_novalue:nF {#1}
2390     {
2391         \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2392     }
2393     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2394     \tl_if_novalue:nTF {#2}
2395     {
2396         \dim_set_eq:cc
2397         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2398         { \__enumext_labelsep_ \__enumext_level: _dim }
2399     }
2400     {
2401         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2402     }
2403     \bool_if:NT \l__enumext_check_ans_bool
2404     {
2405         \int_gincr:N \g__enumext_count_item_number_int
2406     }
2407     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2408     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2409 }

```

(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).

```

2410 \cs_new_protected:Npn \__enumext_redefine_item:
2411 {
2412     \RenewDocumentCommand \item { s o o }
2413     {
2414         \bool_if:nTF {##1}
2415         {
2416             \__enumext_starred_item:nn {##2} {##3}
2417         }
2418         { \__enumext_default_item:n {##2} }
2419     }
2420 }

```

(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`.

```

2421 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2422 {
2423   \tl_if_novalue:nTF { #1 }
2424   {
2425     \bool_set_true:N \__enumext_wrap_label_v_bool
2426     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2427   }
2428   {
2429     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2430     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2431   }
2432 }

```

(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 $\langle label \rangle$.

```

2433 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2434 {
2435   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2436   \__enumext_keyans_show_left:n { #1 }
2437   \bool_set_true:N \__enumext_wrap_label_v_bool
2438   \__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 $\langle sequence \rangle$.

```

2439   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2440   \__enumext_keyans_addto_prop:n { #1 }
2441   \__enumext_keyans_store_ref:
2442   \__enumext_keyans_addto_seq:n { #1 }
2443 }

```

(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).

```

2444 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2445 {
2446   \RenewDocumentCommand \item { s o }
2447   {
2448     \bool_if:nTF {##1}
2449     {
2450       \peek_remove_spaces:n
2451       {
2452         \__enumext_keyans_starred_item:n {##2}
2453       }
2454     }
2455     {
2456       \__enumext_keyans_default_item:n {##2}
2457     }
2458   }
2459 }

```

(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.

```
2460 \cs_new_protected:Nn \__enumext_item_starred:
2461 {
2462   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2463   {
2464     \mode_leave_vertical:
2465     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2466     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2467     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2468   }
2469 }
```

(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).

```
2470 \cs_new_protected:Nn \__enumext_make_label:
2471 {
2472   \RenewDocumentCommand \makelabel { m }
2473   {
2474     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2475     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2476     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2477     {
2478       \__enumext_item_starred:
2479       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2480     }
2481     { ##1 }
2482     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2483     \tl_gclear:N \g__enumext_item_symbol_tl
2484   }
2485 }
```

(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).

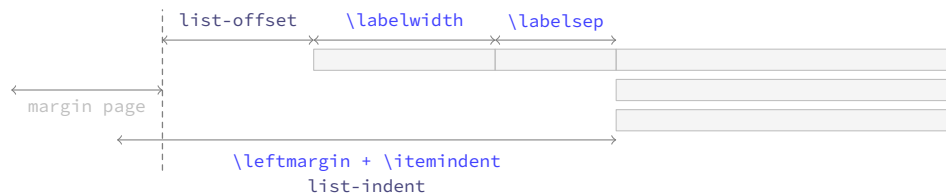
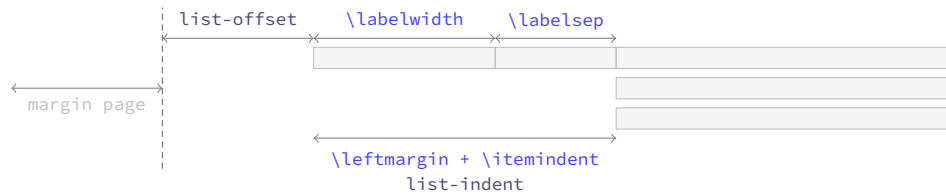
```
2486 \cs_new_protected:Nn \__enumext_keyans_make_label:
2487 {
2488   \RenewDocumentCommand \makelabel { m }
2489   {
2490     \tl_use:N \l__enumext_label_fill_left_v_tl
2491     \tl_use:N \l__enumext_label_font_style_v_tl
2492     \bool_if:NTF \l__enumext_wrap_label_v_bool
2493     {
2494       \__enumext_wrapper_label_v:n { ##1 }
2495     }
2496     { ##1 }
2497     \tl_use:N \l__enumext_label_fill_right_v_tl
2498   }
2499 }
```

(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).

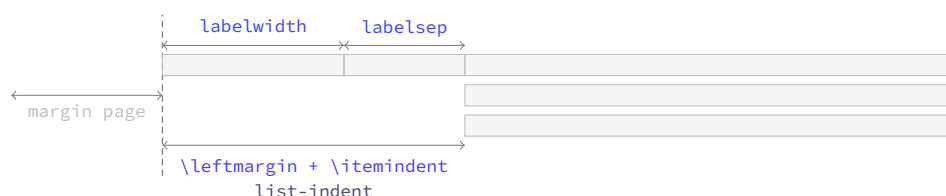
```
2500 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
2501 {
2502   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2503   {
2504     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2505     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2506   }
2507   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2508   {
2509     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2510     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2511   }
2512 }
```

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

```
2512 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2513 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2514 {
2515   \dim_set:Nn #6 { #1 + #2 - #4 }
2516   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2517 }
2518 {
2519   \dim_compare:nNnT { #4 } = { #1 + #2 }
2520   { \dim_set:Nn #6 { \c_zero_dim } }
2521   \dim_compare:nNnT { #4 } < { #1 + #2 }
2522   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2523   \dim_compare:nNnT { #4 } > { #1 + #2 }
2524   {
2525     \dim_set:Nn #6 { -#1 - #2 + #4 }
2526     \dim_set:Nn #6 { #6*-1 }
2527   }
2528 }
```

Figure 11: Default horizontal lengths in `enumext`.

```

2528         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2529     }
2530 }
2531 \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.

```

2532 \cs_set_protected:Npn \__enumext_tmp:n #1
2533 {
2534     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2535     {
2536         \__enumext_calc_hspace:ccccc
2537         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2538         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2539         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2540         { \__enumext_leftmargin_tmp_#1_bool }
2541         \clist_map_inline:nn
2542         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2543         { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2544         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2545         { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2546         \usecounter { enumX#1 }
2547         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2548         \str_if_eq:nnTF {#1} { v }
2549         {
2550             \__enumext_keyans_redefine_item:
2551             \__enumext_keyans_make_label:
2552             \__enumext_keyans_fake_item:
2553             \bool_if:cT { \__enumext_show_length_#1_bool }
2554             {
2555                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2556             }
2557         }
2558         {
2559             \__enumext_redefine_item:
2560             \__enumext_make_label:
2561             \__enumext_use_key_ref:
2562             \__enumext_fake_item:
2563             \bool_if:cT { \__enumext_show_length_#1_bool }
2564             {
2565                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2566             }
2567         }
2568     }
2569 }
2570 \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.

```

2571 \cs_set_protected:Npn \__enumext_tmp:n #1
2572 {
2573     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2574     {
2575         \__enumext_calc_hspace:ccccc
2576         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2577         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }

```

```

2578     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2579     { \__enumext_leftmargin_tmp_#1_bool }
2580 \clist_map_inline:nn
2581     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2582     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2583 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2584     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2585 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2586 \skip_zero:N \partopsep
2587 \usecounter { enumX#1 }
2588 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2589 \__enumext_use_key_ref_h:
2590 \str_if_eq:nnTF {#1} { vii }
2591     {
2592         \__enumext_fake_item_vii:
2593         \bool_if:cT { \__enumext_show_length_vii_bool }
2594             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2595     }
2596     {
2597         \__enumext_fake_item_viii:
2598         \bool_if:cT { \__enumext_show_length_#1_bool }
2599             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2600     }
2601 }
2602 }
2603 \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.

```

2604 \NewDocumentEnvironment{enumext}{0}{ }
2605 {
2606     \__enumext_safe_exec:
2607     \__enumext_parse_keys:n {#1}
2608     \__enumext_before_list:
2609     \__enumext_start_store_level:
2610     \__enumext_start_list:nn
2611     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2612     {
2613         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2614         \__enumext_before_keys_exec:
2615     }
2616     \__enumext_after_args_exec:
2617 }
2618 {
2619     \__enumext_stop_list:
2620     \__enumext_stop_store_level:
2621     \__enumext_after_list:
2622 }

```

(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.

```

2623 \cs_new_protected:Nn \__enumext_safe_exec:
2624 {
2625     \__enumext_current_env_set_bool:
2626     \int_incr:N \__enumext_level_int
2627     \int_compare:nNnT { \__enumext_level_int } > { 4 }
2628         { \msg_fatal:nn { enumext } { list-too-deep } }
2629     \bool_set_true:N \l__enumext_standar_bool
2630     \bool_lazy_all:nT
2631     {
2632         { \bool_if_p:N \g__enumext_standar_bool }
2633         { \int_compare_p:nNn { \__enumext_level_int } = { 1 } }
2634         { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2635     }
2636     {
2637         \typeout{[[ON-FIRST-LEVEL-ENUMEXT-NOT-NESTED]]}

```

```

2638         \bool_set_true:N \l__enumext_standar_level_one_bool
2639     }
2640 }

```

(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.

```

2641 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2642 {
2643     \tl_if_novalue:nF {#1}
2644     {
2645         \str_clear:N \l__enumext_series_str
2646         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2647         {
2648             \keys_set:nn { enumext / level-1 } {#1}
2649             \__enumext_parse_series:n {#1}
2650         }
2651         {
2652             \exp_args:Ne \keys_set:nn
2653             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2654         }
2655         \bool_if:NT \l__enumext_store_active_bool
2656         {
2657             \__enumext_parse_store_keys:n {#1}
2658         }
2659     }
2660 }

```

(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:`.

```

2661 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2662 {
2663     \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2664     {
2665         \regex_match:nnT { \b columns\b } {#1}
2666         {
2667             \int_set_eq:cc
2668             { \l__enumext_store_columns_ \__enumext_level: _int }
2669             { \l__enumext_columns_ \__enumext_level: _int }
2670             \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2671             {
2672                 columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2673             }
2674         }
2675     }
2676     \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2677     {
2678         \regex_match:nnT { \b columns-sep\b } {#1}
2679         {
2680             \dim_set_eq:cc
2681             { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2682             { \l__enumext_columns_sep_ \__enumext_level: _dim }
2683             \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2684             {
2685                 columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2686             }
2687         }
2688     }
2689 }

```

(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.

```

2690 \cs_new_protected:Nn \__enumext_start_store_level:
2691 {
2692   \bool_lazy_all:nT
2693   {
2694     { \bool_if_p:N \l__enumext_store_active_bool }
2695     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2696     { \bool_not_p:n { \g__enumext_starred_bool } }
2697   }
2698   {
2699     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2700     {
2701       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2702       \__enumext_store_level_open:
2703     }
2704   }
2705   \bool_lazy_all:nT
2706   {
2707     { \bool_if_p:N \l__enumext_store_active_bool }
2708     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2709     { \bool_if_p:N \g__enumext_starred_bool }
2710   }
2711   {
2712     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2713     {
2714       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2715       \__enumext_store_level_open:
2716     }
2717   }
2718 }
2719 \cs_new_protected:Nn \__enumext_stop_store_level:
2720 {
2721   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2722   {
2723     \__enumext_store_level_close:
2724   }
2725 }

```

(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.

```

2726 \cs_new_protected:Nn \__enumext_before_list:
2727 {
2728   \__enumext_vspace_above:
2729   \__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.

```

2730   \__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.

```

2731   \dim_compare:nNnT
2732   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2733   {
2734     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2735     {
2736       \linewidth
2737       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2738       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2739     }

```

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*.

```

2740         \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
2741         \int_gincr:N \g__enumext_minipage_stat_int
2742         \__enumext_mini_addvspace:
2743         \nointerlineskip\noindent
2744         \begin{\__enumext_mini_env*}
2745         { \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
2746     }
2747     \__enumext_multicols_start:
2748 }

```

(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.

```

2749 \cs_new_protected:Nn \__enumext_multicols_start:
2750 {
2751     \int_compare:nNt
2752     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2753     {
2754         \dim_compare:nNt
2755         { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2756         {
2757             \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
2758             {
2759                 ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
2760                 + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
2761                 ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
2762                 - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
2763             }
2764         }
2765         \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
2766         \skip_zero:N \multicolsep
2767         \int_compare:nNt { \__enumext_level_int } > { 1 }
2768         {
2769             \dim_zero:N \columnseprule
2770         }
2771     }

```

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.

```

2771         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2772         {
2773             \__enumext_multi_addvspace:
2774         }
2775         \raggedcolumns
2776         \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2777     }
2778 }

```

(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.

```

2779 \cs_new_protected:Nn \__enumext_multicols_stop:
2780 {
2781     \int_compare:nNt
2782     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2783     {
2784         \end{multicols}
2785         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2786         {
2787             \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2788         }
2789     }

```

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.

```

2790   \bool_lazy_and:nnT
2791   { \bool_if_p:N \l__enumext_check_ans_bool }
2792   { \bool_not_p:n { \g__enumext_starred_bool } }
2793   {
2794     \bool_gset_true:N \g__enumext_check_ans_show_bool
2795     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2796   }
2797 }

```

(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.

```

2798 \cs_new_protected:Nn \__enumext_after_list:
2799 {
2800   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2801   {
2802     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2803     {
2804       \msg_warning:nn { enumext } { missing-miniright }
2805       \miniright
2806     }
2807     \int_gzero:N \g__enumext_minipage_stat_int
2808     \end{__enumext_mini_env*}
2809     \par\addvspace { \l__enumext_minipage_after_skip }
2810   }
2811   { \__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.

```

2812   \__enumext_after_stop_list:
2813   \__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.

```

2814   \bool_set_false:N \l__enumext_standar_bool
2815   \__enumext_resume_save_counter:
2816 }

```

(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`.

```

2817 \__enumext_after_env:nn {enumext}
2818 {
2819   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2820   {
2821     \bool_if:NT \g__enumext_check_ans_show_bool
2822     {
2823       \__enumext_check_ans_show:
2824     }
2825     \bool_gset_false:N \g__enumext_standar_bool
2826     \bool_gset_false:N \g__enumext_check_ans_show_bool
2827     \tl_gclear:N \g__enumext_store_name_tl
2828   }
2829 }

```

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.

```

2830 \NewDocumentEnvironment{keyans}{ 0{ } }
2831 {
2832   \__enumext_keyans_safe_exec:
2833   \__enumext_keyans_parse_keys:n {#1}
2834   \__enumext_before_list_v:
2835   \__enumext_start_list:nn
2836   { \tl_use:N \l__enumext_label_v_tl }
2837   {
2838     \__enumext_list_arg_two_v:
2839     \__enumext_before_keys_exec_v:
2840   }
2841   \__enumext_after_args_exec_v:
2842 }
2843 {
2844   \__enumext_keyans_check_ans:nn { item }{ keyans }
2845   \__enumext_stop_list:
2846   \__enumext_after_list_v:
2847 }

```

(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.

```

2848 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2849 {
2850   \bool_if:NF \l__enumext_store_active_bool
2851   {
2852     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2853   }
2854   \int_incr:N \l__enumext_keyans_level_int
2855   \bool_set_true:N \l__enumext_keyans_env_bool
2856   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2857   \bool_set_false:N \l__enumext_store_active_bool
2858   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2859   {
2860     \msg_error:nn { enumext } { keyans-nested }
2861   }
2862   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2863   {
2864     \msg_error:nn { enumext } { keyans-wrong-level }
2865   }
2866 }

```

(End of definition for *__enumext_keyans_safe_exec:*.)

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

```

2867 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2868 {
2869   \keys_set:nn { enumext / keyans } {#1}
2870 }

```

(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.

```

2871 \cs_new_protected:Nn \__enumext_before_list_v:
2872 {
2873   \__enumext_vspace_above_v:
2874   \__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.

```

2875   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2876   {
2877     \dim_set:Nn \l__enumext_minipage_left_v_dim
2878     {

```

```

2879         \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2880     }

```

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*.

```

2881     \bool_set_true:N \l__enumext_minipage_active_v_bool
2882     \int_gincr:N \g__enumext_minipage_stat_int
2883     \__enumext_keyans_mini_addvspace:
2884     \nointerlineskip\noindent
2885     \begin{\__enumext_minienv*}{ \l__enumext_minipage_left_v_dim }
2886 }

```

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

```

2887 \__enumext_keyans_multicols_start:
2888 }

```

(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.

```

2889 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2890 {
2891     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2892     {

```

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

```

2893         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2894         {
2895             \dim_set:Nn \l__enumext_columns_sep_v_dim
2896             {
2897                 (
2898                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2899                 ) / \l__enumext_columns_v_int
2900                 - \l__enumext_listoffset_v_dim
2901             }
2902         }
2903         \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).

```

2904         \skip_zero:N \multicolsep
2905         \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.

```

2906     \bool_if:NF \l__enumext_minipage_active_v_bool
2907     {
2908         \__enumext_keyans_multi_addvspace:
2909     }
2910     \raggedcolumns
2911     \begin{multicols}{ \l__enumext_columns_v_int }
2912 }
2913 }

```

(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.

```

2914 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2915 {
2916     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2917     {
2918         \end{multicols}
2919         \bool_if:NF \l__enumext_minipage_active_v_bool
2920         {

```

```

2921         \par\addvspace{ \l__enumext_multicols_below_v_skip }
2922     }
2923 }
2924 }

```

(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.

```

2925 \cs_new_protected:Nn \__enumext_after_list_v:
2926 {
2927     \bool_if:NTF \l__enumext_minipage_active_v_bool
2928     {
2929         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2930         {
2931             \msg_warning:nn { enumext } { missing-miniright }
2932             \miniright
2933         }
2934         \int_gzero:N \g__enumext_minipage_stat_int
2935         \end{\__enumext_mini_env*}
2936         \par\addvspace{ \l__enumext_minipage_after_skip }
2937     }
2938     { \__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.

```

2939     \bool_set_false:N \l__enumext_keyans_env_bool
2940     \__enumext_after_stop_list_v:
2941     \__enumext_vspace_below_v:
2942 }

```

(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.

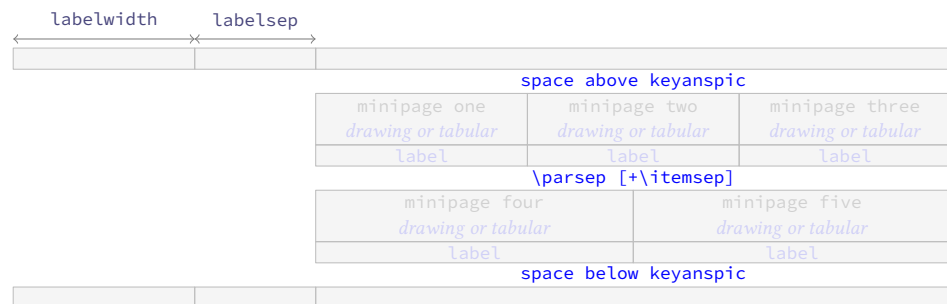


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

```

2943 \NewDocumentCommand \anspic { s o +m }
2944 {

```

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

```

2945 \bool_if:NF \__enumext_store_active_bool
2946 {
2947   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2948 }
2949 \int_compare:nNt { \__enumext_level_int } > { 1 }
2950 {
2951   \msg_error:nn { enumext } { keyanspic-wrong-level }
2952 }
2953 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2954 {
2955   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2956 }

```

The three arguments are handled by the function `__enumext_keyans_anspic_code:nnn` and stored in the sequence `__enumext_keyans_pic_body_seq` which is processed by the `keyanspic` environment.

```

2957 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
2958 {
2959   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2960 }
2961 }

```

(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.

```

2962 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2963 {
2964   \stepcounter { enumXvi }
2965   #3 \\\
2966   \bool_if:nT { #1 }
2967   {
2968     \__enumext_keyans_addto_prop:n { #2 }
2969     \__enumext_keyans_store_ref:
2970     \__enumext_keyans_addto_seq:n { #2 }
2971     \bool_lazy_or:nnT
2972     { \bool_if_p:N \__enumext_show_answer_bool }
2973     { \bool_if_p:N \__enumext_show_position_bool }
2974     {
2975       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
2976       \__enumext_keyans_show_left:n { #2 }
2977       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
2978     }
2979   }
2980   \tl_use:N \__enumext_label_font_style_v_tl
2981   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2982 }

```

(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.

```

2983 \NewDocumentEnvironment{keyanspic}{ o }
2984 {
2985   \__enumext_keyans_pic_safe_exec:
2986   \__enumext_start_list:nn
2987   { }
2988   {
2989     \__enumext_keyans_pic_arg_two:
2990   }

```

We apply the “adjusted” vertical spacing above the environment

```

2991 \vspace { \__enumext_keyans_pic_above_skip }
2992 }

```

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.

```

2993 {
2994   \tl_if_novalue:nTF { #1 }
2995   {
2996     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2997   }
2998   { \__enumext_keyans_pic_do:n { #1 } }
2999   \__enumext_stop_list:
3000   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
3001   \setcounter { enumXvi } { 0 }
3002   \vspace { \l__enumext_topsep_v_skip }
3003   %\bool_set_false:N \l__enumext_store_active_bool
3004 }

```

(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.

```

3005 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3006 {
3007   \int_incr:N \l__enumext_keyans_pic_level_int
3008   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3009   {
3010     \msg_error:nn { enumext } { keyanspic-nested }
3011   }
3012 }

```

(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`.

```

3013 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3014 {
3015   \dim_compare:nNnT { #1 } < { 0pt }
3016   { \skip_set:Nn #1 { -#1 } }
3017 }

```

(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.

```

3018 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3019 {

```

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.

```

3020   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3021   \__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.

```

3022   \skip_add:Nn \parsep { \itemsep }
3023   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3024   \dim_zero:N \labelwidth
3025   \dim_zero:N \listparindent
3026   \dim_zero:N \labelsep
3027   \skip_zero:N \partopsep
3028   \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.

```

3029   \__enumext_keyans_pic_skip_abs:N \parsep
3030   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3031   {
3032     \box_dp:N \strutbox

```

```

3033         + \l__enumext_topsep_v_skip
3034         - \parsep
3035     }
3036     \__enumext_item_std:w \scan_stop:
3037 }

```

(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.

```

3038 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3039 {
3040     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3041 }
3042 \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 *⟨stored⟩* by `\anspic*` in the `\l__enumext_keyans_pic_body_seq` sequence inside them.

```

3043 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3044 {
3045     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3046     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3047     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3048     \int_step_inline:nnn
3049     { \l__enumext_keyans_pic_above_int + 1 }
3050     { \l__enumext_keyans_pic_below_int }
3051     {
3052         \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3053         \centering
3054         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3055         \__enumext_endminipage:
3056     }
3057     \par
3058 }

```

(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 (*⟨number⟩*) 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 (*⟨number⟩*).

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.

```

3059 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3060 {
3061     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3062     {
3063         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3064         {
3065             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3066             / \l__enumext_columns_vii_int
3067         }
3068     }
3069     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3070     \dim_set:Nn \l__enumext_item_width_vii_dim
3071     {
3072         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3073         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim

```

```

3074         - \l__enumext_labelsep_vii_dim
3075     }
3076     \dim_zero_new:N \itemwidth
3077 }

```

(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.

```

3078 \cs_new_protected:Npn \l__enumext_starred_joined_item_vii:n #1
3079 {
3080     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3081     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3082     {
3083         \msg_warning:nnee { enumext } { item-joined }
3084         { \int_use:N \l__enumext_joined_item_vii_int }
3085         { \int_use:N \l__enumext_columns_vii_int }
3086         \int_set:Nn \l__enumext_joined_item_vii_int
3087         {
3088             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3089         }
3090     }
3091     \int_compare:nNnT
3092     { \l__enumext_joined_item_vii_int }
3093     >
3094     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3095     {
3096         \msg_warning:nnee { enumext } { item-joined-columns }
3097         { \int_use:N \l__enumext_joined_item_vii_int }
3098         {
3099             \int_eval:n
3100             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3101         }
3102         \int_set:Nn \l__enumext_joined_item_vii_int
3103         {
3104             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3105         }
3106     }
}

```

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

```

3107     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3108     {
3109         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3110         \int_decr:N \l__enumext_joined_item_aux_vii_int
3111         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3112         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3113         \dim_set:Nn \l__enumext_joined_width_vii_dim
3114         {
3115             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3116             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3117                 + \l__enumext_columns_sep_vii_dim
3118                 ) * \l__enumext_joined_item_aux_vii_int
3119         }
3120         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3121     }
3122     {
3123         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3124         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3125     }
3126 }

```

(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.

```

3127 \cs_new_protected:Nn \l__enumext_start_mini_vii:
3128 {
3129     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3130     {

```



```

3131         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3132         {
3133             \linewidth
3134             - \l__enumext_minipage_right_vii_dim
3135             - \l__enumext_minipage_hsep_vii_dim
3136         }
3137     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3138     \dim_gset_eq:NN
3139         \g__enumext_minipage_right_vii_dim
3140         \l__enumext_minipage_right_vii_dim
3141     \__enumext_mini_addvspace_vii:
3142     \nointerlineskip\noindent
3143     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3144 }
3145 }

```

(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”.

```

3146 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3147 {
3148     \bool_if:NT \l__enumext_minipage_active_vii_bool
3149     {
3150         \end{__enumext_mini_env*}
3151         \hfill
3152         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3153     }
3154 }

```

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”.

```

3155 \__enumext_after_env:nn {enumext*}
3156 {
3157     \bool_if:NT \g__enumext_minipage_active_vii_bool
3158     {
3159         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3160         \par\addvspace { \g__enumext_minipage_right_skip }
3161         \bool_if:NF \g__enumext_minipage_center_vii_bool
3162         {
3163             \centering
3164         }
3165         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3166         \end{__enumext_mini_env*}
3167         \par\addvspace{ \g__enumext_minipage_after_skip }
3168     }
3169     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3170     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3171     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3172     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3173     \bool_gset_false:N \g__enumext_starred_bool
3174 }

```

(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.

```

3175 \NewDocumentEnvironment{enumext*}{ o }
3176 {
3177     \__enumext_safe_exec_vii:
3178     \__enumext_parse_keys_vii:n {#1}
3179     \__enumext_before_list_vii:
3180     \__enumext_start_store_level_vii:
3181     \__enumext_start_list:nn { }
3182     {
3183         \__enumext_list_arg_two_vii:
3184         \__enumext_before_keys_exec_vii:
3185     }

```

```

3186     \__enumext_starred_columns_set_vii:
3187     \item[] \scan_stop:
3188     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3189     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3190   }
3191   {
3192     \__enumext_stop_item_tmp_vii:
3193     \__enumext_remove_extra_parsep_vii:
3194     \__enumext_stop_list:
3195     \__enumext_stop_store_level_vii:
3196     \__enumext_after_list_vii:
3197   }

```

(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`.

```

3198 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3199 {
3200   \__enumext_current_env_set_bool:
3201   \int_incr:N \l__enumext_level_h_int
3202   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3203   {
3204     \msg_error:nn { enumext } { nested }
3205   }
3206   \bool_set_true:N \l__enumext_starred_bool
3207   \bool_lazy_all:nT
3208   {
3209     { \bool_if_p:N \g__enumext_starred_bool }
3210     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3211     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
3212   }
3213   {
3214     \typeout{[[ON-FIRST-LEVEL-ENUMEXT*-NOT-NESTED]]}
3215     \bool_set_true:N \l__enumext_starred_level_one_bool
3216   }
3217 }

```

(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.

```

3218 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3219 {
3220   \tl_if_novalue:nF {#1}
3221   {
3222     \str_clear:N \l__enumext_series_str
3223     \keys_set:nn { enumext / enumext* } {#1}
3224     \__enumext_parse_series:n {#1}
3225     \bool_if:NT \l__enumext_store_active_bool
3226     {
3227       \__enumext_parse_store_keys_vii:n {#1}
3228     }
3229   }
3230 }

```

(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:`.

```

3231 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3232 {
3233   \bool_if:NF \l__enumext_store_columns_vii_bool
3234   {
3235     \regex_match:nnT { \b columns\b } {#1}
3236     {

```

```

3237         \int_set_eq:NN
3238         \l__enumext_store_columns_vii_int
3239         \l__enumext_columns_vii_int
3240         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3241         {
3242             columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3243         }
3244     }
3245 }
3246 \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3247 {
3248     \regex_match:nnT { \b columns-sep \b} {#1}
3249     {
3250         \dim_set_eq:NN
3251         \l__enumext_store_columns_sep_vii_dim
3252         \l__enumext_columns_sep_vii_dim
3253         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3254         {
3255             columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3256         }
3257     }
3258 }
3259 }

```

(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`.

```

3260 \cs_new_protected:Nn \__enumext_before_list_vii:
3261 {
3262     \__enumext_vspace_above_vii:
3263     \__enumext_check_ans_exec: % need by chek-ans
3264     \__enumext_before_args_exec_vii:
3265     \__enumext_start_mini_vii:
3266 }

```

(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.

```

3267 \cs_new_protected:Nn \__enumext_after_list_vii:
3268 {
3269     \__enumext_stop_mini_vii:
3270     \__enumext_after_stop_list_vii:
3271     \__enumext_vspace_below_vii:
3272     \bool_set_false:N \l__enumext_starred_bool
3273     \__enumext_resume_save_counter:
3274 }

```

(End of definition for `__enumext_after_list_vii:.`)

`__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`.

```

3275 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3276 {
3277     \bool_if:NT \l__enumext_store_active_bool
3278     {
3279         \int_compare:nNNT { \l__enumext_level_int } > { \c_zero_int }
3280         {
3281             \__enumext_store_level_open_vii:
3282         }
3283     }
3284 }
3285 \cs_new_protected:Nn \__enumext_stop_store_level_vii:

```

```

3286 {
3287     \bool_if:NT \l__enumext_store_active_bool
3288     {
3289         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3290         {
3291             \__enumext_store_level_close_vii:
3292         }
3293     }
3294 }

```

(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.

```

3295 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3296 {
3297     \__enumext_stop_item_tmp_vii:
3298     \int_incr:N \l__enumext_item_column_pos_vii_int
3299     \int_gincr:N \g__enumext_item_count_all_vii_int
3300     \__enumext_item_peek_args_vii:
3301 }

```

(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).

```

3302 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3303 {
3304     \peek_meaning:NTF (
3305     { \__enumext_joined_item_vii:w }
3306     { \__enumext_joined_item_vii:w (1) }
3307 }

```

(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.

```

3308 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3309 {
3310     \__enumext_starred_joined_item_vii:n {#1}
3311     \peek_meaning_remove:NTF *
3312     { \__enumext_starred_item_vii:w }
3313     { \__enumext_standard_item_vii:w }
3314 }

```

(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].

```

3315 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3316 {
3317     \bool_set_false:N \l__enumext_item_starred_vii_bool
3318     \peek_meaning:NTF [
3319     {
3320         \bool_set_eq:NN
3321         \l__enumext_wrap_label_vii_bool

```

```

3322         \l__enumext_wrap_label_opt_vii_bool
3323     \__enumext_start_item_vii:w
3324 }
3325 {
3326     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3327     \legacy_if_set_true:n { @noitemarg }
3328     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3329 }
3330 }

```

(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*[\symbol] and \item*[\symbol][\offset].

```

\__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
3331 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3332 {
3333     \bool_set_true:N \l__enumext_item_starred_vii_bool
3334     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3335     \peek_meaning:NTF [
3336         { \__enumext_starred_item_vii_aux_i:w }
3337         { \__enumext_starred_item_vii_aux_ii:w }
3338     ]
3339     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3340     {
3341         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3342         \__enumext_starred_item_vii_aux_ii:w
3343     }
3344     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3345     {
3346         \peek_meaning:NTF [
3347             { \__enumext_starred_item_vii_aux_iii:w }
3348             {
3349                 \dim_set_eq:NN
3350                 \l__enumext_item_symbol_sep_vii_dim
3351                 \l__enumext_labelsep_vii_dim
3352                 \legacy_if_set_true:n { @noitemarg }
3353                 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3354             }
3355         ]
3356     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3357     {
3358         \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3359         \legacy_if_set_true:n { @noitemarg }
3360         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3361     }

```

(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.

```

3362 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3363 {
3364     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3365     \legacy_if:nT { @noitemarg }
3366     {
3367         \legacy_if_set_false:n { @noitemarg }
3368         \legacy_if:nT { @nmbrrlist }
3369         {
3370             \bool_if:NT \l__enumext_hyperref_bool
3371             {
3372                 \legacy_if_set_true:n { @hyper@item }
3373             }
3374             \refstepcounter{enumXvii}
3375             \bool_if:NT \l__enumext_check_ans_bool

```

```

3376         {
3377         \int_gincr:N \g__enumext_count_item_number_int
3378         }
3379     }
3380 }

```

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.

```

3381 \group_begin:
3382 \lrbox{ \l__enumext_item_text_vii_box }
3383 \bool_if:NF \l__enumext_footnotes_key_bool
3384 {
3385     \__enumext_renew_footnote:
3386 }
3387 \bool_if:NT \l__enumext_item_starred_vii_bool
3388 {
3389     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3390     {
3391         \tl_gset_eq:NN
3392         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3393     }
3394     \mode_leave_vertical:
3395     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3396     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3397     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3398     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3399 }
3400 \group_begin:
3401 \tl_use:N \l__enumext_label_font_style_vii_tl
3402 \bool_if:NTF \l__enumext_wrap_label_vii_bool
3403 {
3404     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3405     { \__enumext_wrapper_label_vii:n {#1} }
3406 }
3407 {
3408     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3409 }
3410 \group_end:
3411 \skip_horizontal:N \l__enumext_labelsep_vii_dim
3412 \tl_use:N \l__enumext_after_list_args_vii_tl
3413 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3414 \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3415 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3416 \tl_use:N \l__enumext_fake_item_indent_vii_tl
3417 }

```

(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.

```

3418 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3419 {
3420     \__enumext_endminipage:
3421     \endlrbox
3422     \group_end:
3423     \box_set_wd:Nn \l__enumext_item_text_vii_box
3424     {
3425         \l__enumext_joined_width_vii_dim
3426         + \l__enumext_labelwidth_vii_dim
3427         + \l__enumext_labelsep_vii_dim
3428     }
3429     \int_set:Nn \hbadness { 10000 }
3430     \box_use:N \l__enumext_item_text_vii_box
3431     \bool_if:NF \l__enumext_footnotes_key_bool
3432     {

```

```

3433     \__enumext_print_footnote:
3434   }
3435   \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3436   {
3437     \par\noindent
3438     \int_zero:N \__enumext_item_column_pos_vii_int
3439   }
3440   { \hspace{ \__enumext_columns_sep_vii_dim } }
3441 }

```

(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.

```

3442 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3443 {
3444   \int_compare:nNnT
3445   {
3446     \int_mod:nn { \g__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3447   }
3448   =
3449   { \c_zero_int }
3450   {
3451     \par
3452     \vspace{ -\__enumext_itemsep_vii_skip }
3453     \int_gzero:N \g__enumext_item_count_all_vii_int
3454   }
3455 }

```

(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`.

```

3456 \__enumext_after_env:nn {enumext*}
3457 {
3458   \int_compare:nNnT { \__enumext_level_int } = { 0 }
3459   {
3460     \bool_if:NT \g__enumext_check_ans_show_h_bool
3461     {
3462       \__enumext_check_ans_show:
3463     }
3464     \bool_gset_false:N \g__enumext_starred_bool
3465     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3466     \tl_gclear:N \g__enumext_store_name_tl
3467   }
3468 }

```

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.

```

3469 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3470 {
3471   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3472   {
3473     \dim_set:Nn \__enumext_columns_sep_viii_dim
3474     {
3475       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
3476       / \__enumext_columns_viii_int
3477     }
3478   }
3479   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - \c_one_int }
3480   \dim_set:Nn \__enumext_item_width_viii_dim
3481   {
3482     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
3483     / \__enumext_columns_viii_int - \__enumext_labelwidth_viii_dim
3484     - \__enumext_labelsep_viii_dim
3485   }
3486   \dim_zero_new:N \itemwidth
3487 }

```

(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`.

```

3488 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3489 {
3490   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3491   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3492   {
3493     \msg_warning:nnee { enumext } { item-joined }
3494     { \int_use:N \l__enumext_joined_item_viii_int }
3495     { \int_use:N \l__enumext_columns_viii_int }
3496     \int_set:Nn \l__enumext_joined_item_viii_int
3497     {
3498       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3499     }
3500   }
3501   \int_compare:nNnT
3502   { \l__enumext_joined_item_viii_int }
3503   >
3504   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3505   {
3506     \msg_warning:nnee { enumext } { item-joined-columns }
3507     { \int_use:N \l__enumext_joined_item_viii_int }
3508     {
3509       \int_eval:n
3510       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3511     }
3512     \int_set:Nn \l__enumext_joined_item_viii_int
3513     {
3514       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3515     }
3516   }
3517   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3518   {
3519     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3520     \int_decr:N \l__enumext_joined_item_aux_viii_int
3521     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3522     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3523     \dim_set:Nn \l__enumext_joined_width_viii_dim
3524     {
3525       \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3526       + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3527         + \l__enumext_columns_sep_viii_dim
3528       ) * \l__enumext_joined_item_aux_viii_int
3529     }
3530     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3531   }
3532   {
3533     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3534     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3535   }
3536 }

```

(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:`

```

3537 \cs_new_protected:Npn \__enumext_start_mini_viii:
3538 {
3539   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3540   {
3541     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3542     {
3543       \linewidth
3544       - \l__enumext_minipage_right_viii_dim
3545       - \l__enumext_minipage_hsep_viii_dim
3546     }
3547     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3548     \dim_gset_eq:NN
3549     \g__enumext_minipage_right_viii_dim
3550     \l__enumext_minipage_right_viii_dim

```



```

3551         \__enumext_mini_addvspace_viii:
3552         \nointerlineskip\noindent
3553         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3554     }
3555 }
3556 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3557 {
3558     \bool_if:NT \l__enumext_minipage_active_viii_bool
3559     {
3560         \end{__enumext_mini_env*}
3561         \hfill
3562         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3563     }
3564 }
3565 \__enumext_after_env:nn {keyans*}
3566 {
3567     \bool_if:NT \g__enumext_minipage_active_viii_bool
3568     {
3569         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3570         \par\addvspace { \g__enumext_minipage_right_skip }
3571         \bool_if:NF \g__enumext_minipage_center_viii_bool
3572         {
3573             \centering
3574         }
3575         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3576         \end{__enumext_mini_env*}
3577         \par\addvspace{ \g__enumext_minipage_after_skip }
3578     }
3579     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3580     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3581     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3582     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3583 }

```

(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.

```

3584 \NewDocumentEnvironment{keyans*}{ o }
3585 {
3586     \__enumext_safe_exec_viii:
3587     \__enumext_parse_keys_viii:n {#1}
3588     \__enumext_before_list_viii:
3589     \__enumext_start_list:nn { }
3590     {
3591         \__enumext_list_arg_two_viii:
3592         \__enumext_before_keys_exec_viii:
3593     }
3594     \__enumext_starred_columns_set_viii:
3595     \item[] \scan_stop:
3596     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3597     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3598 }
3599 {
3600     \__enumext_stop_item_tmp_viii:
3601     \__enumext_remove_extra_parsep_viii:
3602     \__enumext_keyans_check_ans:nn { item }{ keyans* }
3603     \__enumext_stop_list:
3604     \__enumext_after_list_viii:
3605 }

```

(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.

```

3606 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3607 {
3608     \int_incr:N \l__enumext_keyans_level_h_int
3609     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3610     {
3611         \msg_error:nn { enumext } { nested }

```

```

3612     }
3613     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3614     \bool_set_false:N \l__enumext_store_active_bool
3615     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
3616     {
3617         \msg_error:nn { enumext } { keyans-wrong-level }
3618     }
3619 }

```

(End of definition for \l__enumext_safe_exec_viii:.)

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

```

3620 \cs_new_protected:Npn \l__enumext_parse_keys_viii:n #1
3621 {
3622     \tl_if_novalue:nF {#1}
3623     {
3624         \keys_set:nn { enumext / keyans* } {#1}
3625     }
3626 }

```

(End of definition for \l__enumext_parse_keys_viii:n.)

\l__enumext_before_list_viii: The function \l__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 \l__enumext_start_mini_viii: handle by *mini-env*.

```

3627 \cs_new_protected:Nn \l__enumext_before_list_viii:
3628 {
3629     \l__enumext_vspace_above_viii:
3630     \l__enumext_before_args_exec_viii:
3631     \l__enumext_start_mini_viii:
3632 }

```

(End of definition for \l__enumext_before_list_viii:.)

\l__enumext_after_list_viii: The function \l__enumext_after_list: first call the function \l__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.

```

3633 \cs_new_protected:Nn \l__enumext_after_list_viii:
3634 {
3635     \l__enumext_stop_mini_viii:
3636     \l__enumext_after_stop_list_viii:
3637     \l__enumext_vspace_below_viii:
3638 }

```

(End of definition for \l__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.

\l__enumext_start_item_tmp_viii: First we will call the function \l__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 \l__enumext_item_peek_args_viii: that will handle the arguments passed to *\item*.

```

3639 \cs_new_protected_nopar:Nn \l__enumext_start_item_tmp_viii:
3640 {
3641     \l__enumext_stop_item_tmp_viii:
3642     \int_incr:N \l__enumext_item_column_pos_viii_int
3643     \int_gincr:N \g__enumext_item_count_all_viii_int
3644     \l__enumext_item_peek_args_viii:
3645 }

```

(End of definition for \l__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).

```

3646 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3647 {
3648   \peek_meaning:NTF (
3649     { \__enumext_joined_item_viii:w }
3650     { \__enumext_joined_item_viii:w (1) }
3651   }

```

(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`.

```

3652 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3653 {
3654   \__enumext_starred_joined_item_viii:n {#1}
3655   \peek_meaning_remove:NTF *
3656   { \__enumext_starred_item_viii:w }
3657   { \__enumext_standard_item_viii:w }
3658 }

```

(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]`.

```

3659 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3660 {
3661   \bool_set_false:N \l__enumext_item_starred_viii_bool
3662   \peek_meaning:NTF [
3663     {
3664       \bool_set_eq:NN
3665         \l__enumext_wrap_label_viii_bool
3666         \l__enumext_wrap_label_opt_viii_bool
3667       \__enumext_start_item_viii:w
3668     }
3669     {
3670       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3671       \legacy_if_set_true:n { @noitemarg }
3672       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3673     }
3674   }

```

(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>]`.

```

3675 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3676 {
3677   \bool_set_true:N \l__enumext_item_starred_viii_bool
3678   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3679   \peek_meaning:NTF [
3680     { \__enumext_starred_item_viii_aux_i:w }
3681     { \__enumext_starred_item_viii_aux_ii:w }
3682   }

```

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)*.

```

3683 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3684 {

```

```

3685 \tl_clear:N \l__enumext_store_keyans_label_tl
3686 \tl_if_novalue:nF { #1 }
3687 {
3688   \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3689   {
3690     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
3691     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3692   }
3693   \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3694 }
3695 \__enumext_starred_item_viii_aux_ii:w
3696 }
3697 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3698 {
3699   \legacy_if_set_true:n { @noitemarg }
3700   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3701 }

```

(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.

```

3702 \cs_new_protected:Nn \__enumext_starred_item_exec:
3703 {
3704   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3705   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3706   \__enumext_keyans_store_ref:
3707   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3708   \__enumext_keyans_addto_seq_link:
3709   \bool_if:NT \l__enumext_show_answer_bool
3710   {
3711     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3712   }
3713   \bool_if:NT \l__enumext_show_position_bool
3714   {
3715     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3716     {
3717       \group_begin:
3718       \exp_not:N \normalfont
3719       \exp_not:N \footnotesize [ \int_eval:n
3720       {
3721         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3722       }
3723       ]
3724       \group_end:
3725     }
3726     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3727   }
3728 }

```

(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.

```

3729 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3730 {
3731   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3732   \legacy_if:nT { @noitemarg }
3733   {
3734     \legacy_if_set_false:n { @noitemarg }
3735     \legacy_if:nT { @nmbrrlist }
3736     {
3737       \bool_if:NT \l__enumext_hyperref_bool
3738       {
3739         \legacy_if_set_true:n { @hyper@item }
3740       }
3741       \refstepcounter{enumXviii}
3742     }
3743   }

```

```
3743     }
```

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

```
3744     \group_begin:
3745     \lrbox{ \l__enumext_item_text_viii_box }
3746     \bool_if:NF \l__enumext_footnotes_key_bool
3747     {
3748         \__enumext_renew_footnote:
3749     }
3750     \bool_if:NT \l__enumext_item_starred_viii_bool
3751     {
3752         \__enumext_starred_item_exec:
3753     }
3754     \group_begin:
3755     \tl_use:N \l__enumext_label_font_style_viii_tl
3756     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3757     {
3758         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3759         { \__enumext_wrapper_label_viii:n {#1} }
3760     }
3761     {
3762         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3763     }
3764     \group_end:
3765     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3766     \tl_use:N \l__enumext_after_list_args_viii_tl
3767     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3768     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3769     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3770     \bool_if:NT \l__enumext_item_starred_viii_bool
3771     {
3772         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3773         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3774     }
3775     {
3776         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3777     }
3778 }
```

(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.

```
3779 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3780 {
3781     \__enumext_endminipage:
3782     \endlrbox
3783     \group_end:
3784     \box_set_wd:Nn \l__enumext_item_text_viii_box
3785     {
3786         \l__enumext_joined_width_viii_dim
3787         + \l__enumext_labelwidth_viii_dim
3788         + \l__enumext_labelsep_viii_dim
3789     }
3790     \int_set:Nn \hbadness { 10000 }
3791     \box_use:N \l__enumext_item_text_viii_box
3792     \bool_if:NF \l__enumext_footnotes_key_bool
3793     {
3794         \__enumext_print_footnote:
3795     }
3796     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3797     {
3798         \par\noindent
3799         \int_zero:N \l__enumext_item_column_pos_viii_int
3800     }
3801     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3802 }
```

(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.

```

3803 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3804 {
3805   \int_compare:nNnT
3806   {
3807     \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3808   }
3809   =
3810   { \c_zero_int }
3811   {
3812     \par
3813     \vspace{ -\l__enumext_itemsep_viii_skip }
3814     \int_gzero:N \g__enumext_item_count_all_viii_int
3815   }
3816 }

```

(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.

```

3817 \NewDocumentCommand \getkeyans { m }
3818 {
3819   \exp_args:Ne \__enumext_getkeyans_aux:n
3820   { \tl_to_str:e { \text_expand:n {#1} } }
3821 }

```

(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.

```

3822 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3823 {
3824   \str_if_in:nnTF {#1} { : }
3825   {
3826     \use:e
3827     {
3828       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3829       { {##1} {##2} }
3830     }
3831     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3832   }
3833   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3834 }

```

(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⟩`.

```

3835 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3836 {
3837   \prop_if_exist:cF { g__enumext_#1_prop }
3838   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3839   \group_begin:
3840     \prop_item:cn { g__enumext_#1_prop }{#2}
3841   \group_end:
3842 }

```

(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`.

```

3843 \keys_define:nn { keyanskey / print }
3844 {
3845   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3846                 {
3847                   \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3848                 },
3849   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3850   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3851                 {
3852                   \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3853                 },
3854   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3855   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3856                 {
3857                   \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3858                 },
3859   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3860   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3861                 {
3862                   \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3863                 },
3864   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3865   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3866                 {
3867                   \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3868                 },
3869   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3870 }

```

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

```

3871 \NewDocumentCommand \printkeyans { s O{ } m }
3872 {
3873   \group_begin:
3874     \tl_use:N \l__enumext_print_keyans_i_tl
3875     \tl_use:N \l__enumext_print_keyans_ii_tl
3876     \tl_use:N \l__enumext_print_keyans_iii_tl
3877     \tl_use:N \l__enumext_print_keyans_iv_tl
3878     \tl_use:N \l__enumext_print_keyans_vii_tl
3879     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3880   \group_end:
3881 }

```

(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

```

3882 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3883 {
3884   \seq_if_exist:cTF { g__enumext_#3_seq }
3885   {
3886     \seq_if_empty:cF { g__enumext_#3_seq }
3887     {
3888       %%\seq_show:c { g__enumext_#3_seq }
3889       \bool_if:nTF {#1}
3890       {
3891         \begin{enumext*}[#2]
3892         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3893         \end{enumext*}
3894       }
3895     }

```

```

3896         \begin{enumext}[#2]
3897         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3898         \end{enumext}
3899     }
3900 }
3901 }
3902 {
3903     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3904 }
3905 }

```

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

10.40 The command `\setenumext`

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

```

3906 \keys_define:nn { enumext / meta-families }
3907 {
3908     level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3909     level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3910     level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3911     level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3912     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3913     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3914     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3915     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3916     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3917     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3918     print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3919     print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3920     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3921 }

```

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

```

3922 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3923 {
3924     level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3925     keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3926 }

```

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

```

3927 \NewDocumentCommand \setenumext { o +m }
3928 {
3929     \tl_if_novalue:nTF {#1}
3930     {
3931         \seq_map_inline:Nn \c__enumext_all_families_seq
3932     }
3933     {
3934         \seq_clear:N \l__enumext_setkey_tmpa_seq
3935         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3936         \int_set:Nn \l__enumext_setkey_tmpa_int
3937         {
3938             \seq_count:N \l__enumext_setkey_tmpb_seq
3939         }
3940         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3941         {
3942             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3943             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3944             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3945             {
3946                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3947             }
3948         }
3949         {
3950             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3951         }
3952         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3953         { \seq_map_inline:Nn \c__enumext_all_families_seq }
3954         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3955     }
3956     {
3957         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3958     }

```



```

3958     }
3959 }

```

(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.

```

3960 \cs_new_protected:Npn \__enumext_set_parse:n #1
3961 {
3962   \tl_set:Nc \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3963   \int_step_inline:nnn { 0 } { 4 } {%<- max level
3964     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3965     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3966       {
3967         \seq_put_right:Nc \l__enumext_setkey_tmpa_seq
3968           { \tl_trim_spaces:n {#1} }
3969       }
3970     { \__enumext_set_error:nn {#1} { } }
3971   }
3972   \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3973   { \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.

```

3974 \msg_new:nnn { enumext } { package-load }
3975 {
3976   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3977 }
3978 \msg_new:nnn { enumext } { package-not-load }
3979 {
3980   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3981 }
3982 \msg_new:nnn { enumext } { package-load-foot }
3983 {
3984   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3985 }

```

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

```

3986 \msg_new:nnn { enumext } { counters }
3987 {
3988   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
3989   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3990 }

```

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

```

3991 \msg_new:nnn { enumext } { invalid-key }
3992 {
3993   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3994 }
3995 \msg_new:nnn { enumext } { unknown-key-family }
3996 {
3997   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3998 }

```

Messages used in length calculation.

```

3999 \msg_new:nnn { enumext } { width-negative }
4000 {
4001   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4002   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4003 }
4004 \msg_new:nnn { enumext } { width-zero }
4005 {
4006   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4007   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4008 }

```

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

```

4009 \msg_new:nnn { enumext } { list-lengths }
4010 {
4011     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4012     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4013     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4014     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4015     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4016     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4017     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4018     \__enumext_show_length:nnn { skip } { topsep } {#1}
4019     \__enumext_show_length:nnn { skip } { parsep } {#1}
4020     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4021     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4022     *****
4023 }

```

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

```

4024 \msg_new:nnn { enumext } { list-lengths-not-nested }
4025 {
4026     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4027     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4028     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4029     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4030     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4031     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4032     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4033     \__enumext_show_length:nnn { skip } { topsep } {#1}
4034     \__enumext_show_length:nnn { skip } { parsep } {#1}
4035     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4036     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4037     *****
4038 }

```

Messages used by `save-ans` key.

```

4039 \msg_new:nnn { enumext } { save-ans-empty }
4040 {
4041     The ~ 'save-ans' ~ key ~ cannot ~ be ~ empty~ in ~ '#1'. ~ \msg_line_context:.
4042 }
4043 \msg_new:nnn { enumext } { save-ans-nested }
4044 {
4045     The ~ 'save-ans' ~ key ~ cannot ~ be ~ used ~ in ~ nested ~ '#1'. ~ \msg_line_context:.
4046 }

```

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

```

4047 \msg_new:nnn { enumext } { items-same-answer }
4048 {
4049     *****~Checking~answers~on~'#1'~OK~*****\\
4050     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4051     *****
4052     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4053 }
4054 \msg_new:nnn { enumext } { item-different-answer }
4055 {
4056     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4057     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4058 }

```

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

```

4059 \msg_new:nnn { enumext } { missing-starred }
4060 {
4061     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4062 }

```

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

```

4063 \msg_new:nnn { enumext } { list-too-deep }
4064 {
4065     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4066     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4067 }

```

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

```

4068 \msg_new:nnn { enumext } { anskey-wrong-place }
4069 {
4070   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4071   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4072 }
4073 \msg_new:nnn { enumext } { anspic-wrong-place }
4074 {
4075   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4076   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4077 }
4078 \msg_new:nnn { enumext } { command-wrong-place }
4079 {
4080   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4081   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4082 }

```

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

```

4083 \msg_new:nnn { enumext } { keyans-nested }
4084 {
4085   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4086 }
4087 \msg_new:nnn { enumext } { keyans-wrong-level }
4088 {
4089   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4090   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4091 }
4092 \msg_new:nnn { enumext } { wrong-place }
4093 {
4094   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4095   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4096 }
4097 \msg_new:nnn { enumext } { keyanspic-nested }
4098 {
4099   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4100 }
4101 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4102 {
4103   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4104   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4105 }

```

Messages used by `\getkeyans` command.

```

4106 \msg_new:nnn { enumext } { undefined-storage-anskey }
4107 {
4108   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4109 }

```

Messages used by `\miniright` command.

```

4110 \msg_new:nnn { enumext } { missing-miniright }
4111 {
4112   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
4113   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4114 }
4115 \msg_new:nnn { enumext } { wrong-miniright-place }
4116 {
4117   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4118   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4119 }
4120 \msg_new:nnn { enumext } { wrong-miniright-use }
4121 {
4122   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4123   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4124 }

```

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

```

4125 \msg_new:nnn { enumext } { nested }
4126 {
4127   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4128 }
4129 \msg_new:nnn { enumext } { item-joined }
4130 {
4131   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.

```

```
4132     }  
4133 \msg_new:nnn { enumext } { item-joined-columns }  
4134 {  
4135     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4136 }
```

10.42 Finish package

Finish package implementation.

```
4137 \file_input_stop:  
4138 </package>
```

11 Index of Implementation

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

Symbols	
<code>*</code>	425
<code>\+</code>	198
<code>\-</code>	198
<code>\\</code> 206, 2965, 3988, 4001, 4006, 4011, 4026, 4049, 4050, 4065, 4070, 4075, 4080, 4089, 4094, 4103, 4112, 4117, 4122	
A	
<code>above</code>	<u>1236</u>
<code>above*</code>	<u>1236</u>
<code>\addvspace</code> .. 883, 911, 1034, 1113, 1176, 1182, 1210, 1227, 2787, 2809, 2921, 2936, 3160, 3167, 3570, 3577	
<code>after</code>	<u>721</u>
<code>align</code>	<u>379</u>
<code>\Alph</code>	31, 35
<code>\Alph</code>	331, 508, 526, 539, 3864
<code>\alph</code>	31, 35
<code>\alph</code>	332, 506, 3854
<code>\anskey</code>	11, 63, <u>1912</u>
<code>\anspic</code>	13, 84, <u>2943</u>
<code>\arabic</code>	31, 32
<code>\arabic</code>	330, 505, 525, 3849, 3869
B	
<code>\b</code>	2665, 2678, 3235, 3248
<code>\baselineskip</code>	43
<code>\baselineskip</code>	1872, 1880
<code>before</code>	<u>721</u>
<code>before*</code>	<u>721</u>
<code>below</code>	<u>1236</u>
<code>below*</code>	<u>1236</u>
bool commands:	
<code>\bool_gset_false:N</code> .. 2825, 2826, 3169, 3173, 3464, 3465, 3579	
<code>\bool_gset_true:N</code> 226, 237, 825, 2794, 3152, 3170, 3562, 3580	
<code>\bool_if:NTF</code> . 271, 283, 300, 1258, 1272, 1285, 1296, 1307, 1318, 1329, 1340, 1392, 1409, 1415, 1424, 1451, 1489, 1494, 1501, 1505, 1527, 1532, 1540, 1547, 1578, 1586, 1629, 1639, 1710, 1720, 1834, 1858, 1865, 1893, 1924, 1937, 1939, 1950, 1970, 2095, 2106, 2110, 2149, 2164, 2237, 2256, 2260, 2373, 2403, 2476, 2492, 2553, 2563, 2593, 2598, 2655, 2663, 2676, 2721, 2771, 2785, 2800, 2821, 2850, 2906, 2919, 2927, 2945, 3148, 3157, 3161, 3225, 3233, 3246, 3277, 3287, 3370, 3375, 3383, 3387, 3402, 3431, 3460, 3558, 3567, 3571, 3709, 3713, 3737, 3746, 3750, 3756, 3770, 3792	
<code>\bool_if:nTF</code> 1211, 1228, 1978, 2414, 2448, 2512, 2966, 3889	
<code>\bool_if_p:N</code> 1558, 1559, 1567, 1568, 1689, 1961, 2004, 2005, 2029, 2038, 2039, 2051, 2067, 2223, 2224, 2270, 2271, 2632, 2694, 2707, 2709, 2791, 2972, 2973, 3209	
<code>\bool_lazy_all:nTF</code> .. 1687, 2027, 2036, 2049, 2065, 2630, 2692, 2705, 3207	
<code>\bool_lazy_and:nnTF</code> .. 222, 233, 1557, 1566, 1960, 2003, 2222, 2790	
<code>\bool_lazy_or:nnTF</code>	2269, 2971
<code>\bool_new:N</code> 25, 26, 27, 28, 29, 30, 31, 40, 50, 71, 76, 77, 82, 83, 86, 106, 108, 110, 113, 114, 123, 124, 125, 126, 135, 136, 161, 172, 174	
<code>\bool_not_p:n</code> 223, 234, 1962, 2054, 2069, 2695, 2696, 2708, 2792	
<code>\bool_set_eq:NN</code>	2381, 2429, 3320, 3664
<code>\bool_set_false:N</code> 280, 1671, 1672, 2814, 2857, 2939, 3003, 3020, 3272, 3317, 3614, 3661	
<code>\bool_set_true:N</code> 262, 266, 372, 649, 1242, 1247, 1514, 1649, 1650, 1796, 1803, 2377, 2407, 2425, 2437, 2629, 2638, 2701, 2714, 2740, 2855, 2881, 3137, 3206, 3215, 3326, 3333, 3334, 3547, 3670, 3677, 3678	
box commands:	
<code>\box_dp:N</code> . 930, 934, 938, 949, 953, 964, 973, 979, 989, 1002, 1008, 1014, 1045, 1046, 1047, 1050, 1060, 1064, 1073, 1080, 1085, 1093, 1122, 1123, 1126, 1133, 1146, 1154, 1160, 1168, 3032	
<code>\box_new:N</code>	47, 167
<code>\box_set_wd:Nn</code>	3423, 3784
<code>\box_use:N</code>	3430, 3791
<code>\box_wd:N</code>	338
C	
<code>\c</code>	425, 426, 549, 551, 563, 565
<code>\cB</code>	426
<code>\cE</code>	426
<code>\centering</code>	1213, 1230, 3053, 3163, 3573
<code>check-ans</code>	<u>1664</u>
Document class:	
<code>article</code>	36
clist commands:	
<code>\clist_const:Nn</code>	179
<code>\clist_map_function:nN</code>	3040
<code>\clist_map_inline:Nn</code> . 378, 591, 654, 720, 735, 816, 1252	
<code>\clist_map_inline:nn</code> . 36, 55, 61, 73, 85, 112, 144, 158, 178, 214, 403, 420, 659, 831, 1358, 1677, 1773, 1791, 1812, 2024, 2158, 2331, 2541, 2544, 2570, 2580, 2583, 2603	
<code>\columnbreak</code>	63
<code>\columnbreak</code>	1964
<code>columns</code>	<u>800</u>
<code>columns*</code>	<u>1792</u>
<code>columns-sep</code>	<u>800</u>
<code>columns-sep*</code>	<u>1792</u>
<code>\columnsep</code>	80, 83
<code>\columnsep</code>	2765, 2903
<code>\columnseprule</code>	80, 83
<code>\columnseprule</code>	2769, 2905
Commands provide by enumext :	
<code>\anskey</code> 24, 25, 57, 58, 61, 63, 65–67, 69, 78, 91, 102, 103, 107	
<code>\anspic*</code>	24, 67–69, 84, 86, 87, 102, 103
<code>\anspic</code>	61, 84, 86, 107
<code>\getkeyans</code>	61, 102, 107
<code>\item*</code> 24, 61, 67–69, 72, 73, 93, 99, 100, 102, 103	
<code>\itemwidth</code>	87, 88, 95, 96
<code>\item</code>	71, 73, 88, 92, 93, 96, 98, 99
<code>\miniright</code>	24, 40, 48, 49, 79, 81, 83, 84, 107
<code>\printkeyans</code>	25, 61, 103
<code>\setenumext</code>	24, 104, 105
Counters defined by enumext :	
<code>enumXiii</code>	23, 30

enumXii 23, 30
 enumXiv 23, 30
 enumXi 23, 30
 enumXviii 23, 30
 enumXvii 23, 30, 93
 enumXvi 23, 30
 enumXv 23, 30

cs commands:

\cs_generate_variant:Nn 340, 356, 555, 571, 1817,
 1826, 1831, 1911, 2531, 3042
 \cs_if_exist:NTF 310
 \cs_new:Nn 192
 \cs_new:Npn 202, 1359, 1368, 1377
 \cs_new_eq:NN 246, 247, 248, 252, 253, 285, 286, 289,
 290
 \cs_new_protected:Nn . 216, 257, 421, 441, 473, 736,
 740, 744, 748, 752, 756, 760, 764, 768, 772, 776, 780,
 784, 788, 792, 796, 832, 844, 868, 885, 896, 920, 995,
 1019, 1036, 1098, 1115, 1137, 1172, 1178, 1253, 1267,
 1281, 1292, 1303, 1314, 1325, 1336, 1422, 1525, 1538,
 1555, 1576, 1627, 1637, 1647, 1682, 1718, 1725, 1832,
 1856, 1863, 1891, 1898, 2015, 2147, 2162, 2190, 2220,
 2265, 2277, 2284, 2336, 2340, 2359, 2410, 2444, 2460,
 2470, 2486, 2623, 2690, 2719, 2726, 2749, 2779, 2798,
 2848, 2871, 2889, 2914, 2925, 2962, 3005, 3018, 3038,
 3043, 3059, 3127, 3146, 3198, 3260, 3267, 3275, 3285,
 3302, 3442, 3469, 3537, 3556, 3606, 3627, 3633, 3646,
 3702, 3803
 \cs_new_protected:Npn 184, 188, 293, 308, 325, 335,
 341, 429, 448, 542, 556, 1200, 1219, 1388, 1407, 1479,
 1512, 1605, 1616, 1818, 1827, 1947, 2092, 2104, 2126,
 2200, 2242, 2250, 2369, 2387, 2421, 2433, 2500, 2534,
 2573, 2641, 2661, 2867, 3013, 3078, 3218, 3231, 3308,
 3315, 3331, 3339, 3344, 3356, 3488, 3620, 3652, 3659,
 3675, 3683, 3697, 3822, 3835, 3882, 3960, 3972
 \cs_new_protected_nopar:Nn ... 3295, 3418, 3639,
 3779
 \cs_new_protected_nopar:Npn 3362, 3729
 \cs_set:Nn 2097
 \cs_set:Npn 2025, 2063, 3828
 \cs_set_eq:NN .. 3188, 3189, 3364, 3596, 3597, 3731
 \cs_set_protected:Nn 208, 660, 676, 688, 700
 \cs_set_protected:Npn 32, 48, 56, 68, 74, 102, 140,
 152, 159, 210, 357, 379, 408, 489, 509, 572, 592, 636,
 655, 712, 721, 800, 817, 1236, 1347, 1664, 1738, 1774,
 1792, 2017, 2151, 2320, 2532, 2571
 \cs_to_str:N 327, 350

D

\d 198
 \DeclareDocumentEnvironment 913

dim commands:

\dim_abs:n 2505, 2510
 \dim_add:Nn 3023
 \dim_compare:nNnTF . 662, 678, 690, 702, 1202, 1221,
 2502, 2507, 2513, 2519, 2521, 2523, 2731, 2754, 2875,
 2893, 3015, 3061, 3129, 3471, 3539
 \dim_compare:nTF 1988
 \dim_gset_eq:NN 3138, 3548
 \dim_gzero:N 3172, 3582
 \dim_new:N 44, 51, 52, 53, 70, 96, 109, 119, 168, 169, 175
 \dim_set:Nn .. 338, 650, 1804, 2401, 2505, 2510, 2512,
 2515, 2516, 2520, 2522, 2525, 2526, 2528, 2734, 2757,
 2877, 2895, 3045, 3063, 3070, 3113, 3131, 3358, 3473,
 3480, 3523, 3541

\dim_set_eq:NN 496, 516, 532, 536, 2396, 2543, 2582,
 2680, 2765, 2903, 3120, 3123, 3124, 3250, 3349, 3530,
 3533, 3534
 \dim_use:N 663, 671, 1203, 1209, 1901, 1904, 1909, 2465,
 2467, 2732, 2737, 2738, 2745, 2755, 2759, 2760, 2762
 \dim_zero:N 2769, 2905, 3024, 3025, 3026
 \dim_zero_new:N 3076, 3486
 \c_zero_dim 665, 679, 691, 703, 1203, 1221, 1990, 2502,
 2507, 2513, 2520, 2732, 2755, 2875, 2893, 3061, 3129,
 3471, 3539

E

\end .. 1206, 1224, 1860, 1895, 2784, 2808, 2918, 2935, 3150,
 3166, 3560, 3576, 3893, 3898
 \endlist 28
 \endlist 247
 \endlrbox 3421, 3782
 \endminipage 28
 \endminipage 253
 enumext 5, 2604
 enumext internal commands:

\g__enumext_ \t__enumext_store_name_tl
 _prop 69
 \l__enumext__resume_name_tl 53
 __enumext_add_pre_parsep: ... 41, 842, 844, 844
 __enumext_after_args_exec: . 39, 736, 748, 2616
 __enumext_after_args_exec_v: . 39, 40, 752, 764,
 2841
 __enumext_after_args_exec_vii: ... 768, 792
 __enumext_after_args_exec_viii: 796
 __enumext_after_env:n 81
 __enumext_after_env:nn .. 81, 95, 188, 188, 2817,
 3155, 3456, 3565
 __enumext_after_hyperref: ... 29, 255, 257, 257
 __enumext_after_list: 81, 91, 98, 2621, 2798, 2798
 \l__enumext_after_list_args_v_tl 766
 \l__enumext_after_list_args_vii_tl 794, 3412
 \l__enumext_after_list_args_viii_tl 798, 3766
 __enumext_after_list_v: .. 84, 2846, 2925, 2925
 __enumext_after_list_vii: ... 3196, 3267, 3267
 __enumext_after_list_viii: .. 3604, 3633, 3633
 __enumext_after_star_env:nn 89
 __enumext_after_stop_list: ... 39, 40, 736, 744,
 2812
 __enumext_after_stop_list_v: 39, 752, 760, 2940
 \l__enumext_after_stop_list_v_tl 762
 __enumext_after_stop_list_vii: 768, 784, 3270
 \l__enumext_after_stop_list_vii_tl ... 786
 __enumext_after_stop_list_viii: . 788, 3636
 \l__enumext_after_stop_list_viii_tl ... 790
 \l__enumext_align_label_vii_str .. 3404, 3408
 \l__enumext_align_label_viii_str . 3758, 3762
 \l__enumext_align_label_X_str 159
 \c__enumext_all_envs_clist .. 179, 378, 591, 654,
 720, 735, 816, 1252
 \c__enumext_all_families_seq . 104, 3922, 3931,
 3953
 __enumext_anskey_wrapper:n 1742, 2102
 __enumext_at_begin_document:n .. 28, 184, 184,
 244, 250
 __enumext_before_args_exec: 38, 736, 736, 2729
 __enumext_before_args_exec_v: .. 39, 752, 752,
 2874
 __enumext_before_args_exec_vii: .. 768, 768,
 3264

```

\__enumext_before_args_exec_viii: 772, 3630
\__enumext_before_keys_exec: 39, 736, 740, 2614
\__enumext_before_keys_exec_v: .. 39, 752, 756,
    2839
\__enumext_before_keys_exec_vii ..... 768
\__enumext_before_keys_exec_vii: 40, 776, 3184
\__enumext_before_keys_exec_viii: .. 40, 780,
    3592
\__enumext_before_list: ... 79, 2608, 2726, 2726
\__enumext_before_list_v: . 82, 2834, 2871, 2871
\__enumext_before_list_vii: 91, 3179, 3260, 3260
\__enumext_before_list_viii: .. 98, 3588, 3627,
    3627
\l__enumext_before_no_starred_key_v_tl 758
\l__enumext_before_no_starred_key_vii_-
    tl ..... 778
\l__enumext_before_no_starred_key_viii_-
    tl ..... 782
\l__enumext_before_starred_key_v_tl ... 754
\l__enumext_before_starred_key_vii_tl . 770
\l__enumext_before_starred_key_viii_tl 774
\__enumext_calc_hspace:NNNNNNN 74, 2500, 2500,
    2531, 2536, 2575
\l__enumext_check_ans_bool ... 71, 72, 123, 1668,
    1672, 1720, 1939, 2237, 2373, 2403, 2791, 3375
\__enumext_check_ans_exec: .. 59, 79, 1718, 1718,
    2730, 3263
\g__enumext_check_ans_item_tl .. 68, 123, 2236,
    2244, 2248
\__enumext_check_ans_set: . 58, 1682, 1682, 1722
\__enumext_check_ans_show: 59, 1725, 1725, 2823,
    3462
\g__enumext_check_ans_show_bool 81, 123, 2794,
    2821, 2826
\g__enumext_check_ans_show_h_bool 123, 3460,
    3465
\l__enumext_columns_sep_v_dim 2893, 2895, 2903
\l__enumext_columns_sep_vii_dim .. 3061, 3063,
    3072, 3117, 3252, 3440
\l__enumext_columns_sep_viii_dim . 3471, 3473,
    3482, 3527, 3801
\l__enumext_columns_v_int 1041, 2891, 2899, 2911,
    2916
\l__enumext_columns_vii_int .. 3066, 3069, 3073,
    3081, 3085, 3088, 3094, 3100, 3104, 3239, 3435, 3446
\l__enumext_columns_viii_int . 3476, 3479, 3483,
    3491, 3495, 3498, 3504, 3510, 3514, 3796, 3807
\g__enumext_count_item_anskey_int .. 68, 123,
    1728, 1736, 1941, 2239
\g__enumext_count_item_number_int 123, 1693,
    1698, 1701, 1704, 1712, 1728, 1735, 2375, 2405, 3377
\g__enumext_count_item_with_ans_int .... 63
\l__enumext_counter_i_tl ..... 32, 317
\l__enumext_counter_ii_tl ..... 32, 318
\l__enumext_counter_iii_tl ..... 32, 319
\l__enumext_counter_iv_tl ..... 32, 320
\l__enumext_counter_style_for_ref_vii_-
    tl ..... 456, 466, 477, 479
\l__enumext_counter_style_for_ref_viii_-
    tl ..... 483, 485
\l__enumext_counter_style_for_ref_X_tl 148
\c__enumext_counter_style_tl .... 32, 148, 423
\g__enumext_counter_styles_tl . 23, 31, 44, 328,
    346
\l__enumext_counter_v_tl ..... 32, 321
\l__enumext_counter_vi_tl ..... 32, 322
\l__enumext_counter_vii_tl ..... 32, 323, 453
\l__enumext_counter_viii_tl ..... 32, 324, 463
\__enumext_current_env_set_bool: 27, 216, 216,
    2625, 3200
\l__enumext_current_widest_dim 23, 44, 352, 497,
    517, 533, 537
\__enumext_default_item:n ... 2369, 2369, 2418
\__enumext_define_counters:Nn 23, 308, 308, 317,
    318, 319, 320, 321, 322, 323, 324
\__enumext_endminipage: . 28, 250, 253, 919, 3055,
    3420, 3781
\__enumext_fake_item: ..... 660, 660, 2562
\l__enumext_fake_item_indent_v_dim 679, 684
\l__enumext_fake_item_indent_v_tl 681, 2426,
    2430, 2438
\l__enumext_fake_item_indent_vii_dim 691, 696
\l__enumext_fake_item_indent_vii_tl 693, 3416
\l__enumext_fake_item_indent_viii_dim . 703,
    708, 3773
\l__enumext_fake_item_indent_viii_tl .. 705,
    3772, 3776
\l__enumext_fake_item_indent_X_tl ..... 74
\__enumext_fake_item_vii: .... 660, 688, 2592
\__enumext_fake_item_viii: .... 660, 700, 2597
\__enumext_filter_series:n 51, 1359, 1359, 1400,
    1413, 1419
\__enumext_filter_series_key:n 52, 1359, 1364,
    1368
\__enumext_filter_series_pair:nn .. 52, 1359,
    1365, 1377
\g__enumext_footnote_arg_seq . 145, 2342, 2355,
    2365
\g__enumext_footnote_int . 145, 2349, 2352, 2354,
    2356
\g__enumext_footnote_int_seq . 145, 2343, 2356,
    2361, 2364
\__enumext_footnotes_key_bool ..... 29
\l__enumext_footnotes_key_bool 25, 29, 94, 135,
    266, 271, 280, 3383, 3431, 3746, 3792
\__enumext_footnotetext:nn ... 2336, 2336, 2366
\__enumext_getkeyans:nn .. 102, 3831, 3835, 3835
\__enumext_getkeyans_aux:n 102, 3819, 3822, 3822
\l__enumext_hyperref_bool 25, 29, 135, 262, 283,
    300, 2005, 2224, 3370, 3737
\__enumext_hypertarget:nn 29, 257, 285, 289, 305
\__enumext_if_is_int:n ..... 196
\__enumext_if_is_int:nTF ..... 196, 544, 558
\l__enumext_item_column_pos_vii_int 92, 3088,
    3094, 3100, 3104, 3111, 3298, 3435, 3438
\l__enumext_item_column_pos_viii_int ... 98,
    3498, 3504, 3510, 3514, 3521, 3642, 3796, 3799
\l__enumext_item_column_pos_X_int ..... 159
\g__enumext_item_count_all_vii_int 92, 3112,
    3299, 3446, 3453
\g__enumext_item_count_all_viii_int 98, 3522,
    3643, 3807, 3814
\g__enumext_item_count_all_X_int ..... 159
\__enumext_item_peek_args_vii: 92, 3300, 3302,
    3302
\__enumext_item_peek_args_viii: . 98, 99, 3644,
    3646, 3646
\__enumext_item_starred: .. 74, 2460, 2460, 2478
\l__enumext_item_starred_vii_bool 3317, 3333,

```


3387
 \l__enumext_item_starred_viii_bool 3661, 3677, 3750, 3770
 \l__enumext_item_starred_X_bool 159
 __enumext_item_std:w 28, 71–73, 86, 244, 248, 2378, 2384, 2408, 2426, 2430, 2438, 3036
 \g__enumext_item_symbol_aux_vii_tl 3341, 3389, 3392, 3396, 3398
 \g__enumext_item_symbol_aux_X_tl 159
 \l__enumext_item_symbol_sep_vii_dim . . 3350, 3358, 3395, 3397
 \g__enumext_item_symbol_tl 23, 72, 37, 2393, 2466, 2483
 \l__enumext_item_symbol_vii_tl 3392
 \l__enumext_item_text_vii_box 3382, 3423, 3430
 \l__enumext_item_text_viii_box 3745, 3784, 3791
 \l__enumext_item_text_X_box 159
 \l__enumext_item_width_vii_dim . . 3070, 3115, 3123, 3124
 \l__enumext_item_width_viii_dim . . 3480, 3525, 3533, 3534
 \l__enumext_item_width_X_dim 159
 \l__enumext_itemindent_X_dim 48
 \l__enumext_itemsep_vii_skip 3452
 \l__enumext_itemsep_viii_skip 3813
 \l__enumext_joined_item_aux_vii_int . . 3109, 3110, 3111, 3112, 3118
 \l__enumext_joined_item_aux_viii_int . 3519, 3520, 3521, 3522, 3528
 \l__enumext_joined_item_aux_X_int . . . 159
 __enumext_joined_item_vii:w . . 92, 3305, 3306, 3308, 3308
 \l__enumext_joined_item_vii_int . . 3080, 3081, 3084, 3086, 3092, 3097, 3102, 3107, 3109, 3115
 __enumext_joined_item_viii:w . 99, 3649, 3650, 3652, 3652
 \l__enumext_joined_item_viii_int . 3490, 3491, 3494, 3496, 3502, 3507, 3512, 3517, 3519, 3525
 \l__enumext_joined_item_X_int 159
 \l__enumext_joined_width_vii_dim . 3113, 3120, 3123, 3413, 3425
 \l__enumext_joined_width_viii_dim 3523, 3530, 3533, 3767, 3786
 \l__enumext_joined_width_X_dim 159
 __enumext_keyans_addto_prop:n 67, 2126, 2126, 2440, 2968
 __enumext_keyans_addto_seq:n . 68, 2200, 2200, 2442, 2970
 __enumext_keyans_addto_seq_link: 2200, 2218, 2220, 3708
 __enumext_keyans_anspic_code:nnn . 85, 2959, 2962, 2962
 __enumext_keyans_check_ans:nn . . 68, 69, 2242, 2242, 2844, 3000, 3602
 __enumext_keyans_default_item:n . . 73, 2421, 2421, 2456
 \l__enumext_keyans_env_bool 20, 2695, 2708, 2855, 2939
 __enumext_keyans_fake_item: . . 660, 676, 2552
 \l__enumext_keyans_item_opt_tl 86, 2254, 2267, 2273, 3693
 \l__enumext_keyans_level_h_int 20, 2178, 3608, 3609
 \l__enumext_keyans_level_int . . 20, 1194, 1928, 2173, 2854, 2858, 2953
 __enumext_keyans_make_label: 31, 74, 2486, 2486, 2551
 __enumext_keyans_mini_addvspace: 46, 83, 1098, 1098, 2883
 __enumext_keyans_mini_right_cmd:n 49, 1196, 1219, 1219
 __enumext_keyans_mini_set_vskip: . 45, 1036, 1036, 1100
 __enumext_keyans_multi_addvspace: . 83, 885, 896, 2908
 __enumext_keyans_multi_set_vskip: . 42, 885, 885, 898
 __enumext_keyans_multicols_start: 83, 2887, 2889, 2889
 __enumext_keyans_multicols_stop: . 83, 1223, 2914, 2914, 2938
 __enumext_keyans_parse_keys:n 2833, 2867, 2867
 \l__enumext_keyans_pic_above_int . 118, 3046, 3047, 3049
 \l__enumext_keyans_pic_above_skip . . 86, 118, 2991, 3030
 __enumext_keyans_pic_arg_two: 86, 2989, 3018, 3018
 \l__enumext_keyans_pic_below_int . 118, 3046, 3047, 3050
 \l__enumext_keyans_pic_body_seq . . 85–87, 118, 2957, 2996, 3054
 __enumext_keyans_pic_do:n 87, 2996, 2998, 3038, 3038, 3042
 \l__enumext_keyans_pic_level_int . . 20, 1186, 1932, 2129, 2168, 2203, 2286, 3007, 3008
 __enumext_keyans_pic_row:n 87, 3040, 3043, 3043
 __enumext_keyans_pic_safe_exec: . . 86, 2985, 3005, 3005
 __enumext_keyans_pic_skip_abs:N . . 86, 3013, 3013, 3029
 \l__enumext_keyans_pic_width_dim . 118, 3045, 3052
 __enumext_keyans_redefine_item: . . 73, 2444, 2444, 2550
 __enumext_keyans_safe_exec: . 2832, 2848, 2848
 __enumext_keyans_show_ans: . . 2250, 2258, 2277
 __enumext_keyans_show_item_opt: . 2250, 2265, 2438, 2981, 3773
 __enumext_keyans_show_left:n . 73, 2250, 2250, 2436, 2976
 __enumext_keyans_show_pos: . . 2250, 2262, 2284
 __enumext_keyans_starred_item:n . . 73, 2433, 2433, 2452
 __enumext_keyans_store_ref: . . 67, 2147, 2147, 2441, 2969, 3706
 __enumext_keyans_store_ref_aux_i: 67, 2147, 2159, 2162
 __enumext_keyans_store_ref_aux_ii: 68, 2147, 2188, 2190
 \l__enumext_keyans_tmpa_dim 86
 \l__enumext_keyans_tmpa_tl 24, 99, 86, 2435, 2439
 \l__enumext_keyans_tmptb_tl 99, 86
 __enumext_keyans_wrapper_opt:n . . 1745, 2273
 \l__enumext_label_copy_i_tl . . 2059, 2166, 2171, 2176, 2181
 \l__enumext_label_copy_v_tl 2176
 \l__enumext_label_copy_vi_tl 2171

<code>\l__enumext_label_copy_vii_tl</code>	2034, 2045, 2076, 2166
<code>\l__enumext_label_copy_viii_tl</code>	2181
<code>\l__enumext_label_copy_X_tl</code>	137
<code>\l__enumext_label_fill_left_v_tl</code>	2490
<code>\l__enumext_label_fill_left_X_tl</code>	74
<code>\l__enumext_label_fill_right_v_tl</code>	2497
<code>\l__enumext_label_fill_right_X_tl</code>	74
<code>\l__enumext_label_font_style_v_tl</code>	2491, 2980
<code>\l__enumext_label_font_style_vii_tl</code>	3401
<code>\l__enumext_label_font_style_viii_tl</code>	3755
<code>\l__enumext_label_i_tl</code>	489
<code>\l__enumext_label_ii_tl</code>	489
<code>\l__enumext_label_iii_tl</code>	489
<code>\l__enumext_label_iv_tl</code>	489
<code>__enumext_label_style:Nnn</code>	23, 31, 341, 341, 356, 494, 514, 530, 534
<code>\l__enumext_label_v_tl</code>	67, 68, 527, 2134, 2208, 2279, 2313, 2435, 2439, 2836, 2975, 2977
<code>\l__enumext_label_vi_tl</code>	67, 68, 527, 2131, 2205, 2975, 2977, 2981
<code>\l__enumext_label_vii_tl</code>	509, 3328, 3353, 3360
<code>\l__enumext_label_viii_tl</code>	509, 3672, 3700, 3704
<code>\l__enumext_label_width_by_box</code>	44, 337, 338
<code>__enumext_label_width_by_box:Nn</code>	31, 335, 335, 340, 352, 568
<code>\l__enumext_labelsep_i_dim</code>	2281, 2317, 3711, 3726
<code>\l__enumext_labelsep_v_dim</code>	2898
<code>\l__enumext_labelsep_vii_dim</code>	3065, 3074, 3116, 3351, 3411, 3427
<code>\l__enumext_labelsep_viii_dim</code>	3475, 3484, 3526, 3765, 3788
<code>\l__enumext_labelwidth_i_dim</code>	2281, 2316, 3711, 3726
<code>\l__enumext_labelwidth_v_dim</code>	2898
<code>\l__enumext_labelwidth_vii_dim</code>	3065, 3073, 3116, 3404, 3408, 3426
<code>\l__enumext_labelwidth_viii_dim</code>	3475, 3483, 3526, 3758, 3762, 3787
<code>\l__enumext_leftmargin_tmp_v_bool</code>	86, 3020
<code>\l__enumext_leftmargin_tmp_X_bool</code>	48
<code>\l__enumext_leftmargin_tmp_X_dim</code>	48
<code>\l__enumext_leftmargin_X_dim</code>	48
<code>__enumext_level:</code>	192, 192, 432, 434, 435, 443, 445, 663, 667, 671, 738, 742, 746, 750, 834, 836, 838, 840, 873, 875, 877, 879, 883, 923, 926, 945, 954, 960, 965, 969, 980, 984, 985, 990, 1026, 1030, 1203, 1209, 1256, 1258, 1260, 1263, 1270, 1272, 1274, 1277, 1836, 1844, 1848, 1852, 2097, 2100, 2101, 2377, 2378, 2382, 2383, 2384, 2391, 2393, 2397, 2398, 2401, 2407, 2408, 2462, 2465, 2467, 2474, 2475, 2476, 2479, 2482, 2611, 2613, 2663, 2668, 2669, 2670, 2672, 2676, 2681, 2682, 2683, 2685, 2701, 2714, 2721, 2732, 2734, 2737, 2738, 2740, 2745, 2752, 2755, 2757, 2759, 2760, 2761, 2762, 2765, 2771, 2776, 2782, 2785, 2787, 2800
<code>\l__enumext_level_h_int</code>	20, 224, 451, 475, 1690, 1707, 2053, 2070, 2634, 3201, 3202, 3210
<code>\l__enumext_level_int</code>	20, 194, 235, 846, 997, 1190, 1684, 2030, 2040, 2046, 2052, 2060, 2068, 2075, 2565, 2626, 2627, 2633, 2646, 2653, 2699, 2712, 2767, 2819, 2862, 2949, 3211, 3279, 3289, 3458, 3615
<code>__enumext_list_arg_two_i:</code>	2532
<code>__enumext_list_arg_two_ii:</code>	2532
<code>__enumext_list_arg_two_iii:</code>	2532
<code>__enumext_list_arg_two_iv:</code>	2532
<code>__enumext_list_arg_two_v:</code>	73, 2532, 2838, 3021
<code>__enumext_list_arg_two_vii:</code>	2571, 3183
<code>__enumext_list_arg_two_viii:</code>	2571, 3591
<code>\l__enumext_listoffset_v_dim</code>	2900
<code>\l__enumext_listparindent_vii_dim</code>	3414
<code>\l__enumext_listparindent_viii_dim</code>	3768
<code>__enumext_make_label:</code>	31, 71, 72, 74, 2470, 2470, 2560
<code>\l__enumext_mark_answer_sym_tl</code>	62, 113, 1751, 1906, 2112, 2288, 2301, 3715
<code>\l__enumext_mark_position_str</code>	113, 1755, 1756, 1779, 1780, 1904
<code>\l__enumext_mark_ref_sym_tl</code>	113, 1765, 2010, 2232
<code>__enumext_mini_addvspace:</code>	45, 79, 1019, 1019, 2742
<code>__enumext_mini_addvspace_vii:</code>	47, 1172, 1172, 3141
<code>__enumext_mini_addvspace_viii:</code>	47, 1172, 1178, 3551
<code>__enumext_mini_env*</code>	913
<code>__enumext_mini_right_cmd:n</code>	48, 49, 1198, 1200, 1200
<code>__enumext_mini_set_vskip:</code>	43, 920, 920, 1021
<code>__enumext_mini_set_vskip_vii:</code>	47, 1115, 1115, 1174
<code>__enumext_mini_set_vskip_viii:</code>	47, 1115, 1137, 1180
<code>__enumext_minipage:w</code>	28, 250, 252, 915, 3052, 3413, 3767
<code>\l__enumext_minipage_active_v_bool</code>	83, 84, 2881, 2906, 2919, 2927
<code>\g__enumext_minipage_active_vii_bool</code>	89, 3152, 3157, 3169
<code>\l__enumext_minipage_active_vii_bool</code>	3137, 3148
<code>\g__enumext_minipage_active_viii_bool</code>	3562, 3567, 3579
<code>\l__enumext_minipage_active_viii_bool</code>	3547, 3558
<code>\g__enumext_minipage_active_X_bool</code>	159
<code>\l__enumext_minipage_active_X_bool</code>	62
<code>\g__enumext_minipage_after_skip</code>	62, 1119, 1131, 3167, 3577
<code>\l__enumext_minipage_after_skip</code>	43, 44, 81, 84, 62, 936, 951, 971, 987, 1002, 1008, 1014, 1028, 1038, 1047, 1050, 1062, 1080, 1091, 1107, 1139, 1152, 1166, 2809, 2936
<code>\g__enumext_minipage_center_vii_bool</code>	3161, 3170
<code>\g__enumext_minipage_center_viii_bool</code>	3571, 3580
<code>\g__enumext_minipage_center_X_bool</code>	159
<code>\l__enumext_minipage_hsep_v_dim</code>	82, 2879
<code>\l__enumext_minipage_hsep_vii_dim</code>	3135
<code>\l__enumext_minipage_hsep_viii_dim</code>	3545
<code>\l__enumext_minipage_left_skip</code>	43, 83, 62, 928, 943, 962, 977, 1024, 1034, 1039, 1045, 1054, 1071, 1083, 1103, 1113, 1117, 1122, 1126, 1140, 1144, 1158, 1176, 1182
<code>\l__enumext_minipage_left_v_dim</code>	82, 2877, 2885
<code>\l__enumext_minipage_left_vii_dim</code>	3131, 3143

```

\l__enumext_minipage_left_viii_dim 3541, 3553
\l__enumext_minipage_left_X_dim ..... 62
\g__enumext_minipage_right_skip 62, 1118, 1123,
    1127, 3160, 3570
\l__enumext_minipage_right_skip .. 43, 62, 932,
    947, 967, 982, 1040, 1046, 1058, 1076, 1087, 1141,
    1148, 1162, 1210, 1227
\l__enumext_minipage_right_v_dim .. 82, 1221,
    1226, 2875, 2879
\g__enumext_minipage_right_vii_dim 88, 3139,
    3159, 3172
\l__enumext_minipage_right_vii_dim 88, 3129,
    3134, 3140
\g__enumext_minipage_right_viii_dim .. 3549,
    3569, 3582
\l__enumext_minipage_right_viii_dim .. 3539,
    3544, 3550
\g__enumext_minipage_right_X_dim ..... 159
\g__enumext_minipage_right_X_skip .... 159
\g__enumext_minipage_stat_int . 79, 83, 62, 1215,
    1232, 2741, 2802, 2807, 2882, 2929, 2934
\g__enumext_miniright_code_vii_tl . 89, 3165,
    3171
\g__enumext_miniright_code_viii_tl 3575, 3581
\g__enumext_miniright_code_X_tl ..... 159
\__enumext_multi_addvspace: ... 42, 80, 868, 868,
    2773
\__enumext_multi_set_vskip: .. 41, 832, 832, 870
\l__enumext_multicols_above_ii_skip ... 851
\l__enumext_multicols_above_iii_skip .. 857
\l__enumext_multicols_above_iv_skip ... 863
\l__enumext_multicols_above_v_skip 887, 901,
    911
\l__enumext_multicols_above_X_skip .... 56
\l__enumext_multicols_below_v_skip 891, 905,
    2921
\l__enumext_multicols_below_X_skip .... 56
\__enumext_multicols_start: 80, 2747, 2749, 2749
\__enumext_multicols_stop: 80, 1205, 2779, 2779,
    2811
\__enumext_newlabel:nn 25, 29, 66, 293, 293, 2086,
    2194
\l__enumext_newlabel_arg_one_tl 25, 29, 66, 67,
    137, 2009, 2079, 2087, 2183, 2195, 2230
\l__enumext_newlabel_arg_two_tl 25, 29, 65, 137,
    2033, 2043, 2057, 2073, 2088, 2170, 2175, 2180, 2196
\__enumext_parse_keys:n ... 52, 2607, 2641, 2641
\__enumext_parse_keys_vii:n 52, 3178, 3218, 3218
\__enumext_parse_keys_viii:n . 3587, 3620, 3620
\__enumext_parse_series:n 52, 1388, 1388, 2649,
    3224
\__enumext_parse_store_keys:n . 78, 2657, 2661,
    2661
\__enumext_parse_store_keys_vii:n . 90, 3227,
    3231, 3231
\l__enumext_parsep_i_skip 849, 851, 1000, 1048
\l__enumext_parsep_ii_skip .... 855, 857, 1006
\l__enumext_parsep_iii_skip ... 861, 863, 1012
\l__enumext_parsep_vii_skip ..... 3415
\l__enumext_parsep_viii_skip ..... 3769
\l__enumext_partopsep_v_skip .. 903, 907, 1074,
    1078, 1085, 1089, 1105, 1109
\l__enumext_partopsep_viii_skip ..... 1150
\__enumext_phantomsection: 29, 257, 286, 290, 306

\__enumext_print_footnote: ... 2336, 2359, 3433,
    3794
\__enumext_print_keyans_box:NN 62, 1898, 1898,
    1911, 2099, 2281, 2315, 3711, 3726
\l__enumext_print_keyans_i_tl .... 3845, 3874
\l__enumext_print_keyans_ii_tl ... 3850, 3875
\l__enumext_print_keyans_iii_tl .. 3855, 3876
\l__enumext_print_keyans_iv_tl ... 3860, 3877
\l__enumext_print_keyans_vii_tl .. 3865, 3878
\l__enumext_print_keyans_X_tl ..... 102
\__enumext_printkeyans:nnn 103, 3879, 3882, 3882
\__enumext_redefine_item: . 72, 2410, 2410, 2559
\l__enumext_ref_aux_tl 148, 432, 434, 437, 453, 455,
    458, 463, 465, 468
\l__enumext_ref_key_arg_tl .. 148, 426, 431, 438,
    450, 459, 469
\__enumext_regex_label_ref_key: .. 32, 33, 421,
    421, 433, 454, 464
\__enumext_register_counter_style:Nn .. 325,
    325, 330, 331, 332, 333, 334
\__enumext_remove_extra_parsep_vii: .. 3193,
    3442, 3442
\__enumext_remove_extra_parsep_viii: . 3601,
    3803, 3803
\__enumext_renew_footnote: ... 2336, 2340, 3385,
    3748
\l__enumext_resume_active_bool 52, 54, 37, 1392,
    1514
\l__enumext_resume_bool ..... 23
\__enumext_resume_counter: .. 54, 55, 1512, 1518,
    1525
\__enumext_resume_counter:n . 52, 54, 1483, 1488,
    1512, 1512, 1582, 1590
\__enumext_resume_counter_save_ans: .. 54, 55,
    1512, 1523, 1555
\__enumext_resume_counter_series: 54, 55, 1512,
    1521, 1538
\g__enumext_resume_int 23, 81, 37, 1435, 1529, 1530
\__enumext_resume_last:n . 52, 1388, 1394, 1407
\l__enumext_resume_name_tl 37, 1431, 1439, 1442,
    1458, 1466, 1469, 1515, 1516, 1544, 1551
\__enumext_resume_save_counter: 53, 1422, 1422,
    2815, 3273
\__enumext_resume_series:n . 54, 1353, 1479, 1479
\__enumext_resume_starred: . 56, 1354, 1576, 1576
\g__enumext_resume_vii_int .. 91, 37, 1462, 1534,
    1535
\__enumext_safe_exec: ..... 27, 2606, 2623, 2623
\__enumext_safe_exec_vii: . 27, 3177, 3198, 3198
\__enumext_safe_exec_viii: ... 3586, 3606, 3606
\l__enumext_series_name_tl ..... 54
\l__enumext_series_str 53, 1351, 1390, 1398, 1399,
    1401, 1403, 1426, 1429, 1433, 1453, 1456, 1460, 2645,
    3222
\__enumext_set_error:nn ..... 3960, 3970, 3972
\__enumext_set_label_ref:n ... 32, 429, 429, 501
\__enumext_set_label_ref_h:n . 33, 448, 448, 521
\__enumext_set_parse:n ..... 3943, 3960, 3960
\l__enumext_setkey_tmpa_int ... 97, 3936, 3940
\l__enumext_setkey_tmpa_seq 97, 3934, 3944, 3950,
    3952, 3954, 3967
\l__enumext_setkey_tmpa_tl .... 97, 3942, 3946
\l__enumext_setkey_tmpb_seq 97, 3935, 3938, 3942,
    3943

```

```

\l__enumext_setkey_tmptb_tl 97, 3962, 3964, 3965
\l__enumext_show_answer_bool . 113, 1759, 1783,
    2106, 2256, 2270, 2972, 3709
\__enumext_show_length:nnn . . 38, 202, 202, 4012,
    4013, 4014, 4015, 4016, 4017, 4018, 4019, 4020, 4021,
    4027, 4028, 4029, 4030, 4031, 4032, 4033, 4034, 4035,
    4036
\l__enumext_show_position_bool 113, 1762, 1786,
    2110, 2260, 2271, 2973, 3713
\g__enumext_standar_bool . 27, 20, 223, 226, 1424,
    1489, 1501, 1527, 1540, 1578, 1710, 2632, 2825
\l__enumext_standar_bool . 20, 2038, 2051, 2067,
    2629, 2814
\g__enumext_standar_keyans_pic_star_env_-
    int . . . . . 134
\g__enumext_standar_keyans_star_env_int 133
\g__enumext_standar_level_one_bool . . . . 57
\l__enumext_standar_level_one_bool 20, 1409,
    1558, 1629, 2638
\g__enumext_standar_series_tl . 37, 1412, 1413,
    1580, 1583
\g__enumext_standar_star_env_int . . 130, 227
\__enumext_standard_item_vii:w 92, 3313, 3315,
    3315
\__enumext_standard_item_viii:w 99, 3657, 3659,
    3659
\g__enumext_starred_bool 27, 90, 91, 20, 234, 237,
    1451, 1494, 1505, 1532, 1547, 1586, 1689, 2029, 2039,
    2069, 2164, 2696, 2709, 2792, 3173, 3209, 3464
\l__enumext_starred_bool . 90, 91, 20, 1962, 1970,
    2054, 2095, 3206, 3272
\__enumext_starred_columns_set_vii: . . 3059,
    3059, 3186
\__enumext_starred_columns_set_viii: . 3469,
    3469, 3594
\__enumext_starred_item:nn . . . 2387, 2387, 2416
\__enumext_starred_item_exec: . 100, 3702, 3702,
    3752
\__enumext_starred_item_vii:w 92, 93, 3312, 3331,
    3331
\__enumext_starred_item_vii_aux_i:w . . 3331,
    3336, 3339
\__enumext_starred_item_vii_aux_ii:w . 3331,
    3337, 3342, 3344
\__enumext_starred_item_vii_aux_iii:w 3331,
    3347, 3356
\__enumext_starred_item_viii:w 99, 3656, 3675,
    3675
\__enumext_starred_item_viii_aux_i:w . 3675,
    3680, 3683
\__enumext_starred_item_viii_aux_ii:w 3675,
    3681, 3695, 3697
\__enumext_starred_joined_item_vii:n . 88, 92,
    3078, 3078, 3310
\__enumext_starred_joined_item_viii:n 96, 99,
    3488, 3488, 3654
\g__enumext_starred_keyans_star_env_int 132
\g__enumext_starred_level_one_bool . . . . 57
\l__enumext_starred_level_one_bool 20, 1415,
    1567, 1639, 3215
\g__enumext_starred_series_tl . 37, 1418, 1419,
    1588, 1591
\g__enumext_starred_star_env_int . . 131, 238
\__enumext_start_from:NNn 35, 542, 542, 555, 577
\l__enumext_start_i_int . . . . . 1530, 1542, 1561
\__enumext_start_item_tmp_vii: 89, 3189, 3295,
    3295
\__enumext_start_item_tmp_viii: 97, 3597, 3639,
    3639
\__enumext_start_item_vii:w . 92, 93, 3323, 3328,
    3353, 3360, 3362, 3362
\__enumext_start_item_viii:w . . 99, 3667, 3672,
    3700, 3729, 3729
\__enumext_start_list:nn 28, 76, 86, 244, 246, 2610,
    2835, 2986, 3181, 3589
\__enumext_start_mini_vii: . 91, 3127, 3127, 3265
\__enumext_start_mini_viii: 98, 3537, 3537, 3631
\__enumext_start_store_level: . 78, 2609, 2690,
    2690
\__enumext_start_store_level_vii: . 91, 3180,
    3275, 3275
\l__enumext_start_vii_int . . . 1535, 1549, 1570
\l__enumext_start_X_int . . . . . 74, 572
\__enumext_stop_item_tmp_vii: . 89, 92, 93, 3188,
    3192, 3297, 3364
\__enumext_stop_item_tmp_viii: . . 97, 98, 3596,
    3600, 3641, 3731
\__enumext_stop_item_vii: 93, 94, 3364, 3418, 3418
\__enumext_stop_item_viii: 101, 3731, 3779, 3779
\__enumext_stop_list: . . 28, 244, 247, 2619, 2845,
    2999, 3194, 3603
\__enumext_stop_mini_vii: 89, 91, 3146, 3146, 3269
\__enumext_stop_mini_viii: . 98, 3537, 3556, 3635
\__enumext_stop_store_level: . . 78, 2620, 2690,
    2719
\__enumext_stop_store_level_vii: . . 91, 3195,
    3275, 3285
\l__enumext_store_active_bool 24, 57, 78, 90, 86,
    1559, 1568, 1649, 1924, 2655, 2694, 2707, 2850, 2857,
    2945, 3003, 3225, 3277, 3287, 3614
\__enumext_store_addto_prop:n 61, 67, 1817, 1818,
    1826, 1949, 2145, 3705
\__enumext_store_addto_seq:n 61, 68, 1827, 1827,
    1831, 1838, 1852, 1860, 1869, 1887, 1895, 2013, 2235
\l__enumext_store_ans_bool . 57, 123, 1650, 1671,
    1834, 1858, 1865, 1893, 1937
\l__enumext_store_anskey_arg_tl 24, 63, 64, 86,
    1955, 1964, 1966, 1972, 1980, 1983, 1993, 1998, 2001,
    2007, 2013
\__enumext_store_anskey_code:nnnn . 63, 1943,
    1947, 1947
\__enumext_store_anskey_show_left:n 66, 1954,
    2104, 2104
\__enumext_store_anskey_show_wrap:n 66, 2092,
    2092, 2108, 2123
\l__enumext_store_columns_break_bool . 1918,
    1961
\l__enumext_store_columns_join_int 86, 1969,
    1974
\l__enumext_store_columns_sep_vii_bool 3246
\l__enumext_store_columns_sep_vii_dim 3251,
    3255
\l__enumext_store_columns_sep_X_bool . . 102
\l__enumext_store_columns_sep_X_dim . . . 102
\l__enumext_store_columns_vii_bool . . . 3233
\l__enumext_store_columns_vii_int 3238, 3242
\l__enumext_store_columns_X_bool . . . . . 102
\l__enumext_store_columns_X_int . . . . . 102
\__enumext_store_internal_ref: . . 63, 65, 1952,

```

[2015](#), [2015](#)
`\l__enumext_store_item_symbol_sep_dim` [1916](#),
[1990](#), [1995](#)
`\l__enumext_store_item_symbol_tl` [1914](#), [1981](#),
[1985](#)
`\l__enumext_store_keyans_item_opt_sep_-`
`tl` [1748](#), [2139](#), [2141](#), [2212](#), [2214](#), [3688](#), [3690](#)
`\l__enumext_store_keyans_item_opt_tl` [86](#)
`\l__enumext_store_keyans_label_tl` [24](#), [67](#), [68](#),
[86](#), [2128](#), [2131](#), [2134](#), [2141](#), [2143](#), [2145](#), [2202](#), [2205](#),
[2208](#), [2214](#), [2216](#), [2226](#), [2235](#), [2236](#), [3685](#), [3690](#), [3691](#),
[3704](#), [3705](#), [3707](#)
`__enumext_store_level_close:` [61](#), [1832](#), [1856](#),
[2723](#)
`__enumext_store_level_close_vii:` [1863](#), [1891](#),
[3291](#)
`__enumext_store_level_open:` [60](#), [61](#), [78](#), [1832](#),
[1832](#), [2702](#), [2715](#)
`__enumext_store_level_open_vii:` [90](#), [1863](#),
[1863](#), [3281](#)
`\g__enumext_store_name_tl` [24](#), [81](#), [86](#), [1730](#), [1733](#),
[2795](#), [2827](#), [3466](#)
`\l__enumext_store_name_tl` [24](#), [56](#), [86](#), [1445](#), [1448](#),
[1472](#), [1475](#), [1563](#), [1572](#), [1607](#), [1608](#), [1618](#), [1619](#), [1651](#),
[1653](#), [1655](#), [1657](#), [1659](#), [1661](#), [1820](#), [1822](#), [1829](#), [2081](#),
[2082](#), [2118](#), [2185](#), [2186](#), [2294](#), [2307](#), [2795](#), [3721](#)
`\l__enumext_store_opt_vii_tl` [1867](#), [1877](#), [1883](#),
[1887](#), [3240](#), [3253](#)
`\l__enumext_store_opt_X_tl` [102](#)
`\l__enumext_store_ref_key_bool` [63](#), [1768](#), [1950](#),
[2004](#), [2149](#), [2223](#)
`\l__enumext_store_upper_level_X_bool` [102](#)
`\l__enumext_store_write_aux_file_tl` [25](#), [66](#), [68](#),
[137](#), [2084](#), [2090](#), [2192](#), [2198](#)
`__enumext_storing_exec:` [57](#), [1605](#), [1631](#), [1641](#),
[1647](#)
`__enumext_storing_set:n` [56](#), [1597](#), [1605](#), [1605](#)
`__enumext_storing_set_vii:n` [56](#), [1602](#), [1605](#),
[1616](#)
`__enumext_storing_standar:` [56](#), [57](#), [1605](#), [1613](#),
[1627](#)
`__enumext_storing_starred:` [56](#), [57](#), [1605](#), [1624](#),
[1637](#)
`\l__enumext_the_counter_vii_tl` [455](#)
`\l__enumext_the_counter_viii_tl` [465](#)
`\l__enumext_the_counter_X_tl` [148](#)
`__enumext_tmp:n` [32](#), [36](#), [48](#), [55](#), [56](#), [61](#), [68](#), [73](#), [74](#), [85](#),
[102](#), [112](#), [140](#), [144](#), [152](#), [158](#), [159](#), [178](#), [210](#), [214](#), [655](#),
[659](#), [1347](#), [1358](#), [1664](#), [1681](#), [1738](#), [1773](#), [1774](#), [1791](#),
[2017](#), [2024](#), [2025](#), [2046](#), [2060](#), [2063](#), [2075](#), [2151](#), [2158](#),
[2532](#), [2570](#), [2571](#), [2603](#)
`__enumext_tmp:nn` [357](#), [378](#), [379](#), [407](#), [408](#), [420](#), [572](#),
[591](#), [636](#), [654](#), [712](#), [720](#), [721](#), [735](#), [800](#), [816](#), [817](#), [831](#),
[1236](#), [1252](#), [1792](#), [1816](#), [2320](#), [2335](#)
`__enumext_tmp:nnn` [489](#), [505](#), [506](#), [507](#), [508](#), [509](#), [525](#),
[526](#)
`__enumext_tmp:nnnnnn` [592](#), [617](#), [620](#), [623](#), [625](#), [627](#),
[630](#), [633](#)
`__enumext_tmp:w` [3828](#), [3831](#)
`\l__enumext_tmpa_vii_int` [3069](#), [3072](#)
`\l__enumext_tmpa_viii_int` [3479](#), [3482](#)
`\l__enumext_tmpa_X_int` [159](#)
`\l__enumext_topsep_v_skip` [889](#), [893](#), [1043](#), [1056](#),
[1064](#), [1069](#), [1089](#), [1093](#), [3002](#), [3033](#)
`\l__enumext_topsep_vii_skip` [1120](#), [1129](#), [1133](#)
`\l__enumext_topsep_viii_skip` [1142](#), [1164](#), [1168](#)
`__enumext_use_key_ref:` [33](#), [441](#), [441](#), [2561](#)
`__enumext_use_key_ref_h:` [33](#), [473](#), [473](#), [2589](#)
`\l__enumext_vspace_a_star_v_bool` [1285](#)
`\l__enumext_vspace_a_star_vii_bool` [1307](#)
`\l__enumext_vspace_a_star_viii_bool` [1318](#)
`\l__enumext_vspace_a_star_X_bool` [74](#)
`__enumext_vspace_above:` [49](#), [1253](#), [1253](#), [2728](#)
`__enumext_vspace_above_v:` [50](#), [1281](#), [1281](#), [2873](#)
`\l__enumext_vspace_above_v_skip` [1283](#), [1287](#),
[1289](#)
`__enumext_vspace_above_vii:` [50](#), [1303](#), [1303](#),
[3262](#)
`\l__enumext_vspace_above_vii_skip` [1305](#), [1309](#),
[1311](#)
`__enumext_vspace_above_viii:` [50](#), [1303](#), [1314](#),
[3629](#)
`\l__enumext_vspace_above_viii_skip` [1316](#), [1320](#),
[1322](#)
`\l__enumext_vspace_b_star_v_bool` [1296](#)
`\l__enumext_vspace_b_star_vii_bool` [1329](#)
`\l__enumext_vspace_b_star_viii_bool` [1340](#)
`\l__enumext_vspace_b_star_X_bool` [74](#)
`__enumext_vspace_below:` [50](#), [1267](#), [1267](#), [2813](#)
`__enumext_vspace_below_v:` [50](#), [1292](#), [1292](#), [2941](#)
`\l__enumext_vspace_below_v_skip` [1294](#), [1298](#),
[1300](#)
`__enumext_vspace_below_vii:` [51](#), [1325](#), [1325](#),
[3271](#)
`\l__enumext_vspace_below_vii_skip` [1327](#), [1331](#),
[1333](#)
`__enumext_vspace_below_viii:` [51](#), [1325](#), [1336](#),
[3637](#)
`\l__enumext_vspace_below_viii_skip` [1338](#), [1342](#),
[1344](#)
`__enumext_widest_from:nnnn` [35](#), [556](#), [556](#), [571](#),
[583](#)
`\g__enumext_widest_label_tl` [23](#), [31](#), [44](#), [345](#), [349](#),
[353](#)
`\l__enumext_wrap_label_opt_v_bool` [2429](#)
`\l__enumext_wrap_label_opt_vii_bool` [92](#), [3322](#)
`\l__enumext_wrap_label_opt_viii_bool` [99](#), [3666](#)
`\l__enumext_wrap_label_opt_X_bool` [74](#)
`\l__enumext_wrap_label_v_bool` [2425](#), [2429](#), [2437](#),
[2492](#)
`\l__enumext_wrap_label_vii_bool` [92](#), [3321](#), [3326](#),
[3334](#), [3402](#)
`\l__enumext_wrap_label_viii_bool` [99](#), [3665](#),
[3670](#), [3678](#), [3756](#)
`\l__enumext_wrap_label_X_bool` [74](#)
`__enumext_wrapper_label_v:n` [2494](#), [2981](#)
`__enumext_wrapper_label_vii:n` [3405](#)
`__enumext_wrapper_label_viii:n` [3759](#)
`__enumext_zero_count_level:` [208](#), [208](#)
`__enumext_zero_parsep:` [44](#), [940](#), [995](#), [995](#)
`enumext*` [5](#), [3175](#)
`enumXi` [317](#)
`enumXii` [317](#)
`enumXiii` [317](#)
`enumXiv` [317](#)
`enumXv` [317](#)
`enumXvi` [317](#)
`enumXvii` [317](#)
`enumXviii` [317](#)

Environments provide by **enumext**:

enumext*	22, 23, 25–27, 30, 32–34, 37, 38, 40, 47, 50–54, 56, 58–65, 67, 70, 71, 76, 77, 79, 90, 91, 93, 95, 96, 98, 100, 103, 106, 107
enumext	22, 23, 25, 27, 30, 31, 33–46, 48–54, 56, 58–63, 65, 67, 70–79, 81, 82, 86–88, 91, 103, 106
keyans*	22–24, 26, 27, 30, 32–34, 37, 38, 40, 47, 50, 51, 57, 58, 60, 61, 67, 71, 76, 97, 98, 106, 107
keyanspic	22–25, 30, 31, 34, 48, 57, 58, 61, 67–69, 84–86, 107
keyans	22–25, 27, 30, 31, 34–40, 42, 45, 46, 48–50, 57, 58, 60, 61, 67–69, 73–76, 81, 82, 84–86, 88, 98, 106, 107

Environments:

list	27, 28, 75–77
lrbox	87, 94, 101
minipage	27, 28, 40, 43, 84–87, 94, 101
multicols	41–43, 48, 80, 81, 83, 84

exp commands:

\exp_after:wN	3831
\exp_args:Ne	2652, 3819
\exp_not:N	156, 348, 437, 458, 468, 669, 683, 684, 695, 696, 707, 708, 2009, 2115, 2116, 2228, 2291, 2292, 2304, 2305, 3718, 3719, 3828
\exp_not:n	437, 438, 458, 459, 468, 469, 670, 1375, 1386, 1800, 1807, 1974, 1985, 1995, 2009, 2010, 2087, 2195, 2230, 2232, 2672, 2685, 3242, 3255

F

\fbox	1743
file commands:	
\file_input_stop:	4137
first	721
font	357
\footnote	71
\footnote	71, 2344
\footnotemark	2354
\footnotesize	2116, 2292, 2305, 3719
\footnotetext	2338

G

\getkeyans	13, 102, 3817
group commands:	
\group_begin:	1936, 2114, 2290, 2303, 3381, 3400, 3717, 3744, 3754, 3839, 3873
\group_end:	1945, 2121, 2297, 2310, 3410, 3422, 3724, 3764, 3783, 3841, 3880

H

\hbadness	3429, 3790
hbox commands:	
\hbox_set:Nn	337
\hfill	387, 391, 396, 397, 1207, 1225, 2009, 2228, 3151, 3561
hook commands:	
\hook_gput_code:nnn	9, 186, 190, 255
\hook_gset_rule:nnnn	256
\hspace	3440, 3801
\hyperlink	64, 68
\hyperlink	2009, 2228
\hypertarget	29
\hypertarget	285

I

\IfHyperBoolean	263
\IfPackageLoadedTF	11, 259, 273
\ignorespaces	672
\inputlineno	227, 238

int commands:

\int_add:Nn	3111, 3521
\int_case:nn	846, 997, 1684, 1707
\int_compare:nNnTF	451, 475, 922, 1041, 1186, 1190, 1194, 1727, 1928, 1932, 2129, 2168, 2173, 2178, 2203, 2286, 2627, 2646, 2699, 2712, 2751, 2767, 2781, 2802, 2819, 2858, 2862, 2891, 2916, 2929, 2949, 2953, 3008, 3081, 3091, 3107, 3202, 3279, 3289, 3435, 3444, 3458, 3491, 3501, 3517, 3609, 3615, 3796, 3805, 3940
\int_compare_p:nNn	224, 235, 1690, 2030, 2040, 2052, 2053, 2068, 2070, 2633, 2634, 3210, 3211
\int_decr:N	3110, 3520
\int_eval:n	1822, 2082, 2116, 2186, 2292, 2305, 2547, 2588, 3099, 3509, 3719
\int_from_alph:n	550, 564
\int_from_roman:n	552, 566
\int_gadd:Nn	3112, 3522
\int_gdecr:N	1693, 1698, 1701, 1704, 1712
\int_gincr:N	1529, 1534, 1941, 2239, 2375, 2405, 2741, 2882, 3299, 3377, 3643
\int_gset:Nn	227, 238, 2352
\int_gset_eq:NN	1428, 1435, 1441, 1447, 1455, 1462, 1468, 1474, 2349
\int_gzero:N	212, 1215, 1232, 1735, 1736, 2807, 2934, 3453, 3814
\int_if_exist:NTF	1401, 1439, 1445, 1466, 1472, 1659
\int_incr:N	2626, 2854, 3007, 3201, 3298, 3608, 3642
\int_mod:nn	3446, 3807
\int_new:N	20, 21, 22, 23, 24, 37, 38, 62, 78, 90, 99, 107, 120, 121, 128, 129, 130, 131, 132, 133, 134, 145, 162, 163, 164, 165, 166, 1403, 1661
\int_set:Nn	546, 550, 552, 1542, 1549, 1561, 1570, 1797, 1969, 3046, 3047, 3069, 3080, 3086, 3102, 3429, 3479, 3490, 3496, 3512, 3790, 3936
\int_set_eq:NN	1530, 1535, 2667, 3109, 3237, 3519
\int_step_function:nnN	2046, 2060, 2075
\int_step_inline:nnn	3048, 3963
\int_to_roman:n	194, 2026, 2064
\int_use:N	923, 1544, 1551, 1563, 1572, 2547, 2565, 2588, 2653, 2752, 2761, 2776, 2782, 3084, 3085, 3097, 3494, 3495, 3507
\int_zero:N	3438, 3799
\c_one_int	3069, 3088, 3094, 3100, 3104, 3107, 3479, 3498, 3504, 3510, 3514, 3517
\c_zero_int	224, 235, 2030, 2040, 2052, 2053, 2068, 2070, 3279, 3289, 3449, 3810
\item	28, 39, 40, 62, 71, 84, 86, 87, 89, 97
\item	71, 73, 92, 93, 98, 100, 248, 1840, 1846, 1871, 1879, 1966, 2205, 2208, 2412, 2446, 3187, 3189, 3595, 3597, 3707
\item*	6, 12, 2444
item-pos*	2320
item-sym*	2320
\itemindent	23, 75
\itemindent	74
itemindent	636
\itemsep	85, 86
\itemsep	3022, 3028
\itemwidth	3076, 3120, 3124, 3486, 3530, 3534

K

keyans	11, 2830
keyans*	11, 3584
keyanspic	12, 2983
Keys for environments provide by enumext :	
above*	24, 49, 50

above	24, 49, 50, 79, 82, 91, 98
after	38–40, 81, 84, 91, 98
align	24, 31, 32, 73, 94
before*	38, 39, 79, 91, 98
before	38–40, 82
below*	24, 49–51
below	24, 49–51, 81, 84, 91, 98
check-ans	24, 25, 27, 57, 58, 63, 68, 69, 71, 72, 79, 81, 95, 106
columns-sep*	25, 60, 78, 90
columns-sep	40, 61, 78, 80, 83, 90
columns*	25, 60, 78, 90
columns	23, 40, 43, 49, 61, 78, 80, 83, 90
first	38–40, 94
font	31, 73, 94
item-pos*	63, 64, 70
item-sym*	23, 63, 64, 70, 72
item*-sep	72
itemindent	24, 37, 73, 94
itemsep	36, 76
labelsep	31, 72, 75, 94
labelwidth	30, 31, 34, 35, 75
label	23, 30, 31, 34, 35, 87
lisparindent	76
list-indent	23, 37, 86
list-offset	37
listparindent	37, 94
mark-ans	25, 59, 66
mark-pos	59, 60
mark-ref	25, 59, 65
mini-env	24, 40, 43, 48, 49, 71, 79, 82, 88, 91, 96, 98
mini-sep	24, 40, 79, 82
miniright*	24, 40
miniright	24, 40, 47, 89
minirigth*	27
minirigth	27
no-store	25, 57–59
noitemsep	36, 44
nosep	36, 44
parindent	76
parsep	36, 76, 94
partopsep	36
ref	26, 32, 33
resume*	51, 52, 56, 57
resume	23, 51–57, 76, 81, 91
rightmargin	37
save-ans	24, 52–56, 61, 63, 67, 68, 73, 81, 82, 84, 85, 91, 98, 100, 102, 103, 106
save-key	25, 52
save-ref	25, 29, 59, 63–65, 67, 68, 73, 100
save-sep	59
series	51–56
show-ans	25, 59, 60, 62, 63, 66, 73, 99, 100
show-length	27, 38, 76, 106
show-pos	25, 59, 60, 62, 63, 66, 69, 73, 99, 100
start	24, 27, 35, 76
store-brk	63
topsep	36
widest	23, 27, 35
wrap-ans	59, 62, 66
wrap-label*	31, 71, 73, 92, 94, 99
wrap-label	31, 73, 92, 94, 99
wrap-opt	59
keys commands:	
\keys_define:nn	359, 381, 410, 491, 511, 527, 574, 594, 638, 657, 714, 723, 802, 819, 1238, 1349, 1595, 1600, 1666, 1740, 1776, 1794, 1912, 2322, 3843, 3906
\l_keys_key_str	3997
\keys_set:nn	373, 826, 1243, 1248, 1491, 1496, 1583, 1591, 1958, 2648, 2652, 2869, 3223, 3624, 3908, 3909, 3910, 3911, 3912, 3913, 3914, 3915, 3916, 3917, 3918, 3919, 3957
keyval commands:	
\keyval_parse:NNn	1363
L	
label	489, 509, 527
Labels provide by enumext:	
\Alph*	30, 31
\Roman*	30, 31
\alph*	30, 31
\arabic*	30–32
\roman*	30, 31
\labelsep	86
\labelsep	3023, 3026
labelsep	357
\labelwidth	31, 86
\labelwidth	3023, 3024
labelwidth	357
\leftmargin	23, 75
\leftmargin	74, 3023
legacy commands:	
\legacy_if:nTF	3365, 3368, 3732, 3735
\legacy_if_gset_false:n	916
\legacy_if_set_false:n	3367, 3734
\legacy_if_set_true:n	3327, 3352, 3359, 3372, 3671, 3699, 3739
\linewidth	79, 82
\linewidth	2736, 2879, 3045, 3072, 3133, 3482, 3543
\list	28
\list	246
list-indent	636
list-offset	636
\listparindent	3025
listparindent	636
\lrbox	3382, 3745
M	
\makebox	87
\makebox	1902, 1904, 2466, 3396, 3404, 3408, 3758, 3762
\makelabel	71, 73, 74, 87
\makelabel	73, 74, 2472, 2488
\makesavenoteenv	279
mark-ans	1738
mark-pos	1738, 1774
mark-ref	1738
mini-env	800
mini-sep	800
\minipage	28
\minipage	252
\miniright	10, 48, 1184, 2805, 2932
\miniright*	10
mode commands:	
\mode_if_vertical:TF	871, 899, 1022, 1101
\mode_leave_vertical:	669, 683, 695, 707, 1871, 1879, 1900, 2464, 3394
msg commands:	
\msg_error:nn	2860, 2864, 2951, 3010, 3204, 3611, 3617, 3920

\msg_error:nnn	1188, 1192, 1217, 1234, 1503, 1507, 1610, 1621, 3833, 3838, 3903, 3973
\msg_error:nnnn	1926, 1930, 1934, 2852, 2947, 2955
\msg_fatal:nn	2628
\msg_fatal:nnn	311
\msg_info:nnn	13, 16, 261, 275
\msg_line_context:	4001, 4006, 4011, 4026, 4041, 4045, 4057, 4061, 4065, 4070, 4075, 4080, 4085, 4089, 4094, 4099, 4103, 4108, 4112, 4117, 4122, 4127, 4131, 4135
\msg_new:nnn	3974, 3978, 3982, 3986, 3991, 3995, 3999, 4004, 4009, 4024, 4039, 4043, 4047, 4054, 4059, 4063, 4068, 4073, 4078, 4083, 4087, 4092, 4097, 4101, 4106, 4110, 4115, 4120, 4125, 4129, 4133
\msg_term:nnn	1730
\msg_term:nnnn	2555, 2565, 2594, 2599
\msg_warning:nn	2804, 2931
\msg_warning:nnn	1634, 1644, 1733
\msg_warning:nnnn	2246, 2504, 2509, 3083, 3096, 3493, 3506
\multicolsep	80, 83
\multicolsep	2766, 2904
N	
\NeedsTeXFormat	3
\newcounter	314
\NewDocumentCommand	1184, 1922, 2943, 3817, 3871, 3927
\NewDocumentEnvironment	2604, 2830, 2983, 3175, 3584
\newlabel	29
\newlabel	297
no-store	1664
\noindent	89, 97
\noindent	2743, 2884, 3142, 3188, 3437, 3552, 3596, 3798
\nointerlineskip	2743, 2884, 3142, 3552
noitemsep	592
\nopagebreak	882, 910, 1033, 1112, 1175, 1181
\normalfont	2115, 2291, 2304, 3718
nosep	592
P	
Packages:	
enumext	22, 56, 75, 84, 105
enumitem	30
expl3	87
footnotehyper	29
hyperref	25, 27, 29, 33, 64, 68, 93, 105
lua-visual-debug	43
multicol	22, 105
shortlst	87
\par	882, 910, 1033, 1112, 1175, 1181, 1210, 1227, 2094, 2787, 2809, 2921, 2936, 3057, 3160, 3167, 3437, 3451, 3570, 3577, 3798, 3812
\parindent	3414, 3768
\parsep	41, 44, 85, 86
\parsep	1872, 1880, 2585, 3022, 3029, 3034
parsep	592
\parskip	3415, 3769
\partopsep	86
\partopsep	2586, 3027
partopsep	592
peek commands:	
\peek_meaning:N	3304, 3318, 3335, 3346, 3648, 3662, 3679
\peek_meaning_remove:N	3311, 3655
\peek_remove_spaces:n	2450
\phantomsection	29
\phantomsection	286
prg commands:	
\prg_do_nothing:	290
\prg_new_protected_conditional:Npnn	196
\prg_replicate:nn	205, 4052
\prg_return_false:	200
\prg_return_true:	199
\printkeyans	13, 103, 3871
prop commands:	
\prop_count:N	1822, 2082, 2118, 2186, 2294, 2307, 3721
\prop_gput_if_not_in:Nnn	1817, 1820
\prop_if_exist:N	1651, 3837
\prop_item:Nn	3840
\prop_new:N	1653
\ProvidesExplPackage	4
R	
\raggedcolumns	2775, 2910
\ref	65, 67
ref	489, 509
\refstepcounter	3374, 3741
regex commands:	
\regex_match:nnTF	198, 549, 551, 563, 565, 2665, 2678, 3235, 3248
\regex_replace_once:nnN	425
\renewcommand	437, 458, 468
\RenewDocumentCommand	2344, 2412, 2446, 2472, 2488
\RequirePackage	17
resume	1347
resume*	1347
rightmargin	636
\Roman	31, 35
\Roman	333
\roman	31, 35
\roman	334, 507, 3859
S	
save-ans	1595
save-ref	1738
save-sep	1738
scan commands:	
\scan_stop:	86, 3036, 3187, 3595, 3828, 3831
seq commands:	
\seq_clear:N	3934
\seq_const_from_clist:Nn	3922
\seq_count:N	2996, 3938
\seq_gclear:N	2342, 2343
\seq_gput_right:Nn	1829, 2355, 2356
\seq_if_empty:N	2361, 3886, 3952
\seq_if_exist:N	1655, 3884
\seq_item:Nn	3054
\seq_map_function:NN	3943
\seq_map_inline:Nn	3892, 3897, 3931, 3953, 3954
\seq_map_pairwise_function:NNN	2363
\seq_new:N	100, 101, 118, 146, 147, 1657
\seq_pop_left:NN	3942
\seq_put_right:Nn	2957, 3950, 3967
\seq_set_from_clist:Nn	3935
\seq_set_map_e:NNn	3944
\seq_show:N	3888
series	1347
\setcounter	560, 564, 566, 2547, 2588, 3001
\setenumext	6-9, 104, 3847, 3852, 3857, 3862, 3867, 3927
\setlength	1873, 1881
show-ans	1738, 1774

show-length	712
skip commands:	
\skip_add:Nn .	851, 857, 863, 873, 877, 901, 905, 1002, 1008, 1014, 1024, 1028, 1050, 1103, 1107, 3022
\skip_eval:n	1872, 1880
\skip_gset:Nn	1123, 1127, 1131
\skip_gzero_new:N	1118, 1119
\skip_horizontal:N	684, 696, 708, 3397, 3411, 3765
\skip_horizontal:n ...	670, 1901, 1909, 2465, 2467, 3395, 3773
\skip_if_eq:nnTF	849, 855, 861, 925, 959, 1000, 1006, 1012, 1043, 1048, 1069, 1120, 1142, 1255, 1269, 1283, 1294, 1305, 1316, 1327, 1338
\skip_new:N	58, 59, 63, 64, 65, 66, 67, 122, 176
\skip_set:Nn .	834, 838, 887, 891, 928, 932, 936, 943, 947, 951, 962, 967, 971, 977, 982, 987, 1045, 1046, 1047, 1054, 1058, 1062, 1071, 1076, 1080, 1083, 1087, 1091, 1122, 1126, 1144, 1148, 1152, 1158, 1162, 1166, 3016, 3030
\skip_set_eq:NN	2545, 2584, 2585, 3414, 3415, 3768, 3769
\skip_use:N	836, 840, 875, 879, 883, 903, 907, 926, 945, 954, 960, 965, 969, 980, 984, 985, 990, 1026, 1030, 1056, 1256, 1260, 1263, 1270, 1274, 1277, 2787
\skip_zero:N	2586, 2766, 2904, 3027, 3028
\skip_zero_new:N	1038, 1039, 1040, 1117, 1139, 1140, 1141
\c_zero_skip	849, 855, 861, 926, 960, 1000, 1006, 1012, 1043, 1048, 1069, 1120, 1142, 1256, 1270, 1283, 1294, 1305, 1316, 1327, 1338
\small	3849, 3854, 3859, 3864, 3869
\star	2326
start	572
\stepcounter	2348, 2964
str commands:	
\c_backslash_str	4061, 4070, 4071, 4075, 4076, 4080, 4081, 4112, 4113, 4117, 4122, 4123
\c_colon_str	2081, 2185, 3828
\str_case:nn	218
\str_case:nnTF	1370, 1379
\str_clear:N	2645, 3222
\str_count:n	205, 4052
\str_if_empty:N	1390, 1433, 1460
\str_if_eq:nnTF	2548, 2590
\str_if_in:nnTF	3824
\str_new:N	117, 171
\str_set:Nn ...	413, 414, 415, 1755, 1756, 1779, 1780
\string	279
\strutbox	930, 934, 938, 949, 953, 964, 973, 979, 989, 1002, 1008, 1014, 1045, 1046, 1047, 1050, 1060, 1064, 1073, 1080, 1085, 1093, 1122, 1123, 1126, 1133, 1146, 1154, 1160, 1168, 3032

T

TeX and L ^A T _E X 2 _ε commands:	
\auxout	295
\@currentenv	218
\protected@write	295
text commands:	
\text_expand:n	3820
\textasteriskcentered	1752, 1766
\thepage	301
tl commands:	
\c_space_tl	2273, 4011, 4026
\tl_clear:N	386, 392, 1955, 2128, 2202, 3685

\tl_clear_new:N	343
\tl_const:Nn	148, 327
\tl_gclear:N	1412, 1418, 2248, 2483, 2827, 3171, 3398, 3466, 3581
\tl_gclear_new:N	1398
\tl_gput_right:Nn	328
\tl_greplace_all:Nnn	349
\tl_gset:Nn	1399, 1413, 1419, 2236, 2795, 3341
\tl_gset_eq:NN	345, 2393, 3391
\tl_if_blank:nTF	3389
\tl_if_empty:N	443, 477, 483, 1426, 1431, 1453, 1458, 1516, 1580, 1588, 1608, 1619, 1836, 1867, 1981, 2139, 2212, 2244, 2267, 2462, 3688, 3965
\tl_if_empty:nTF	1481
\tl_if_exist:N	1486
\tl_if_novalue:nTF ..	1956, 1967, 2136, 2210, 2252, 2346, 2371, 2389, 2394, 2423, 2643, 2994, 3220, 3622, 3686, 3929
\tl_map_inline:Nn	346, 423
\tl_new:N	34, 39, 41, 42, 43, 45, 46, 79, 80, 81, 87, 88, 89, 91, 92, 93, 94, 95, 97, 98, 104, 105, 115, 116, 127, 137, 138, 139, 142, 150, 151, 154, 155, 170, 173
\tl_put_left:Nn	1844, 1877, 1964, 2279, 2313, 3704, 3707
\tl_put_right:Nn	344, 435, 456, 466, 1798, 1805, 1848, 1883, 1966, 1972, 1980, 1983, 1993, 1998, 2001, 2007, 2033, 2043, 2057, 2073, 2079, 2084, 2131, 2134, 2141, 2143, 2170, 2175, 2180, 2183, 2192, 2205, 2208, 2214, 2216, 2226, 2670, 2683, 3240, 3253, 3690, 3691, 3845, 3850, 3855, 3860, 3865
\tl_remove_all:Nn	3964
\tl_remove_once:Nn	2021, 2155
\tl_replace_all:Nnn	348
\tl_reverse:N	2020, 2022, 2154, 2156
\tl_set:Nn	156, 313, 387, 391, 396, 397, 431, 450, 667, 681, 693, 705, 1515, 1607, 1618, 2112, 2254, 2288, 2301, 2391, 3693, 3715, 3962
\tl_set_eq:NN	354, 432, 434, 453, 455, 463, 465, 2019, 2153, 2166, 2435, 2439, 2975, 2977
\tl_to_str:n	1486, 1492, 1497, 3820
\tl_trim_spaces:n	344, 3950, 3962, 3968
\tl_use:N .	350, 353, 445, 479, 485, 738, 742, 746, 750, 754, 758, 762, 766, 770, 774, 778, 782, 786, 790, 794, 798, 1906, 2026, 2034, 2045, 2059, 2064, 2076, 2378, 2384, 2408, 2426, 2430, 2438, 2474, 2475, 2482, 2490, 2491, 2497, 2611, 2836, 2980, 3165, 3401, 3412, 3416, 3575, 3755, 3766, 3772, 3776, 3874, 3875, 3876, 3877, 3878, 3946

token commands:

\token_to_str:N	297
\topsep	1873, 1881
topsep	592
\typeout	228, 239, 265, 268, 278, 279, 1411, 1417, 1694, 1713, 2637, 3214

U

\u	426
use commands:	
\use:N	206, 2479, 2613
\use:n	1361, 3826
\use_none:nn	289
\usecounter	2546, 2587

V

\value	1429, 1435, 1442, 1448, 1456, 1462, 1469, 1475
-------------	--

<code>\vspace</code>	917, 1260, 1263, 1274, 1277, 1287, 1289, 1298, 1300, 1309, 1311, 1320, 1322, 1331, 1333, 1342, 1344, 1872, 1880, 2991, 3002, 3452, 3813	<code>wrap-ans</code>	1738
		<code>wrap-label</code>	357
		<code>wrap-label*</code>	357
W		<code>wrap-opt</code>	1738
<code>widest</code>	572		