

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-21^{*}

©2024 by Pablo González[†]

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

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

Abstract

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

Contents

1	Introduction	2	4	The storage system	9
1.1	Description and usage	3	4.1	Keys for storage	9
1.2	The concept of left margin	3	4.2	Keys for internal label and ref	10
1.3	User interface	3	4.3	Keys for debugging and checking	10
1.3.1	Internal counters	3	4.4	The command <code>\anskey</code>	10
1.3.2	Support for multicol	4	4.5	The environment <code>keyans</code>	11
1.3.3	Support for minipage	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	12
1.3.5	Support for <code>\footnote</code>	4	4.6.1	The command <code>\anspic</code>	12
2	The environment <code>enumext</code>	4	4.7	Printing stored content	13
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	13
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	13
3	The command <code>\setenumext</code>	5	5	Full examples	14
3.1	Keys for 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, series and resume	8	11	Index of Implementation	108
3.5	Keys for multicol	9			
3.6	Keys for minipage	9			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-05-21.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

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

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

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* $(x - 1)^2$

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

* $3(x + y + z)$

3. True False

(a) $\alpha > \delta$

* False

(b) \LaTeX 2e is cool?

* Very True!

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

* Yes

(b) Usually uses the package manager?

* Yes, dnf

(c) Rate the following package and class

i. `xsim-exam`

* doesn’t exist for now :(

ii. `xsim`

* very good

iii. `exsheets`

* obsolete

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

(c) i. doesn’t exist for now :(

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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


(A) value

(B) value


(C) value

(D) value


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




(A)




(B)



(C)



(D)



(E)

5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value


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

2. (D)

3. (C), some note

4. (B)

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

1.1 Description and usage

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Support for multicol

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

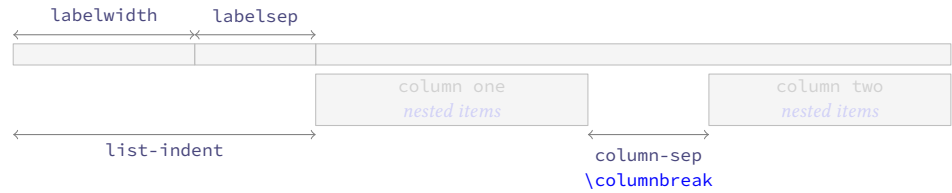


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

1.3.3 Support for minipage

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



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

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

1.3.4 The \label and \ref system

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

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

1.3.5 Support for \footnote

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

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

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

2 The environment enumext

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

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

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

Example

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

X This text is in the first level.

- ★ 2. This text is in the first level.

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

2.1 The `\item*` in `enumext`

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

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Set the *extra vertical space* 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, series and resume

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

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

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

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

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

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

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

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

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

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

3.5 Keys for multicols

`columns = {⟨integer⟩}` default: 1

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

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

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

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

3.6 Keys for minipage

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

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

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

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

3.6.1 The command `\miniright`

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

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

3.6.2 The key `miniright`

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

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

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

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

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

4 The storage system

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

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

4.1 Keys for storage

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

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

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

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

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

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

`save-sep = {<text symbol>}` default: {,}

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

`mark-ans = {<symbol>}` default: \textasteriskcentered

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

`mark-pos = {<left | right>}` default: left

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

4.2 Keys for internal label and ref

`save-ref = {<true | false>}` default: false

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

`mark-ref = {<symbol>}` default: \textasteriskcentered

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

4.3 Keys for debugging and checking

`show-ans = {<true | false>}` default: false

Displays the *current <argument>* passed to `\anskey` in `enumext` environment, the current *<label>* for `\item*` in `keyans` environment and the current *<label>* for `\anspic*` in `keyanspic` environment at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown in square brackets.

`show-pos = {<true | false>}` default: false

Displays the *position* occupied by the “stored content” by `\anskey` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment in *<store name>* set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans = {<true | false>}` default: false

Enables the *checking answer* mechanism. This key works under the logic that each question will contain “only one answer”, it is intended to be used in conjunction with `no-store` key.

`no-store <value forbidden>` default: not used

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

4.4 The command \anskey

`\anskey <anskey>{<content>}`

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

Example

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

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

4.5 The environment keyans

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

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

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

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

4.5.1 The `\item*` in `keyans`

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

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

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

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

Example

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

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

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

1. Text containing a question.

(A) Choice

* (B) Correct choice

(C) Choice

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

(A) Choice

(B) Choice

(C) Choice

(D) Choice

* (E) [note] Correct choice



Some text

4.6 The environment keyanspic

keyanspic

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

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

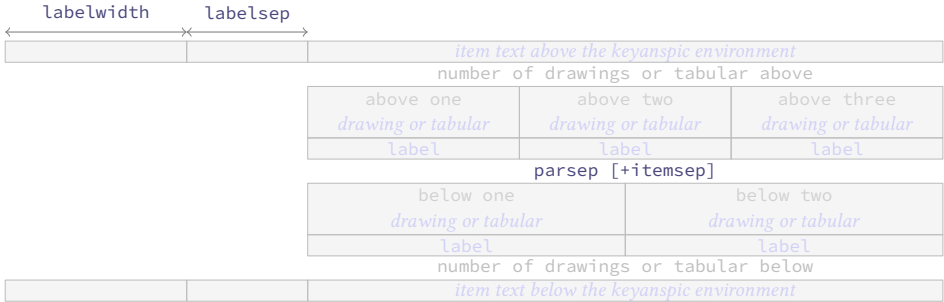


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

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

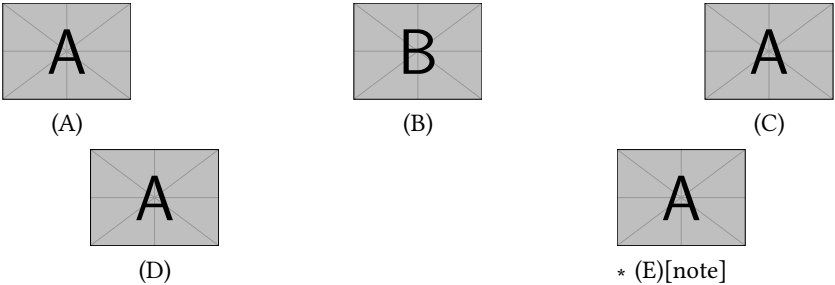
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

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

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

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

(a) ~~LaTeXe~~ is cool?

[2] Very True!
3. Related to Linux

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

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*


5 Full examples

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

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

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

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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


C

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

D

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

Example 2

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

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

A

 36 km/h.

✓

 B

 360 km/h.

C

 27,8 km/h.

D

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

✓

 A

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

B

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

C

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

D

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

A

 36 km/h.

✓

 B

 360 km/h.

C

 27,8 km/h.

D

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

✓

 A

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

B

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

C

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

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
1. B
2. A
3. B
4. A
- *
- *
- *
- *

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

©2024 by Pablo González L

15 / 120

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

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

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

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

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

A

 value

B

 value

C

 value

D

 value

E

 value
4. Question with image and label below:



A



B



C



D



E

5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

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

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

True False
- (a) $\alpha > \delta$

(b) ~~ETX~~ze is cool?
- 4

Related to Linux
- (a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

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

ii. very good

iii. obsolete

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

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

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

Fake description environment

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

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

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

LoNg vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[4]` package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in [Create a fake label ref using list](#) and the answer given by David Carlisle in [Change the use of label ref by data save in an array \(list\)](#) I managed to create a more solid code than the original version, now using the `l3prop`[10] and `l3seq`[10] modules together with the `hyperref`[7] and `enumitem`[5] packages, which did the job, but with some limitations.

As time went by I took these limitations as a personal challenge which I called “*reinventing the wheel*”, since there were packages and classes that did more or less what I was looking for, but did not fit my simple requirements. This “*reinventing the wheel*” finally ended up becoming `enumext`.

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn’t have in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.

Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

The “*random*” type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with \LaTeX , that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

7 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [5] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [6] BERRY, KARL. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [7] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [8] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [9] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [10] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.
- [12] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [13] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [14] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

8 Change history

v1.0 2024-05-21 – First public release.

9 Index of Documentation

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

C

Document class:

article 2

book 2

exam 3

letter 2

report 2

\columnbreak 5

\columnsep 10

Commands provide by enumext:

\anskey 4, 10–12

\anspic* 4, 10, 11, 13

\anspic 11, 13

\getkeyans 4, 11, 14

\item* 4–7, 10–12

\item 6, 7, 9–12

\miniright 4, 5, 10

\printkeyans 4, 6, 11, 14

\setenumext 4, 6, 7, 10–12, 14

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

enumext* 4, 5, 9, 10

enumext 4–6, 9–12, 14, 17

keyans* 4, 5, 10, 11

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

keyans 4–13, 17

Environments:

enumerate 1, 3, 4, 6, 19

list 4, 9, 19

minipage 3–5, 10, 19

multicols 3, 5, 10

I

\item 4, 5

\itemsep 8

K

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 18

before* 9

before 9

below* 8

below 8

check-ans 11

columns-sep 5, 10

columns 5, 8, 10

first 9

font 7

item-pos* 6

item-sym* 6

itemindent 8

itemsep 8, 13

labelsep 4, 6–11, 18

labelwidth 4, 6, 7, 9–11, 18

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

list-indent 4, 8, 9

list-offset 4, 8, 18

listparindent 8

mark-ans 11

mark-pos 11

mark-ref 11

mini-env 5, 8, 10

mini-sep 5, 10

miniright* 10

miniright 10

no-store 11

noitemsep 8

nosep 8, 17

parsep 8, 13

partopsep 8

ref 5, 7

resume* 9

resume* 9

resume 9

rightmargin 8

save-ans 5, 9–14

save-key 9

save-ref 5, 7, 11, 14

save-sep 11

series 9

show-ans 11

show-length 7

show-pos 11, 14

start 9

topsep 8

widest 7

wrap-ans 10

wrap-label* 7, 18

wrap-label 7

wrap-opt 10

L

\label 5

Labels provide by enumext:

\Alph* 7, 12

\Roman* 7

\alph* 7

\arabic* 7

\roman* 7

\labelsep 4, 7

\labelwidth 4, 7

\linewidth 10

\listparindent 8

P

Packages:

enumerate 18

enumext 1–4, 13, 18, 19

enumitem 4, 5, 9, 18, 19

©2024 by Pablo González L

20 / 120

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

10 Implementation

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

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

10.1 General conventions

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

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

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

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

10.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

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

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

10.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-21}
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.)

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq
```

Internal variables used by the command `\setenumext` (§10.39).

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

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

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

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

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

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

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

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

```
124 \bool_new:N \l__enumext_show_answer_bool
125 \bool_new:N \l__enumext_show_position_bool
126 \tl_new:N \l__enumext_mark_ref_sym_tl
127 \tl_new:N \l__enumext_mark_answer_sym_tl
128 \str_new:N \l__enumext_mark_position_str
```

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

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

Internal variables used by `keyanspic` environment (§10.34.2).

```
129 \seq_new:N \l__enumext_keyans_pic_body_seq
130 \dim_new:N \l__enumext_keyans_pic_width_dim
131 \int_new:N \l__enumext_keyans_pic_above_int
132 \int_new:N \l__enumext_keyans_pic_below_int
133 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

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

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

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

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

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

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

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

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

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

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

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

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

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

Internal variables used for redefinition of `\footnote`.

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

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

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

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

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

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

```
\c__enumext_all_envs_clist
```

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

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

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 Some utility functions

```
\__enumext_at_begin_document:n
```

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

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

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

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

```

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

```

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

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

```

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

```

(End of definition for `__enumext_level:`.)

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

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

```

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

```

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

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

```

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

```

(End of definition for `__enumext_regex_counter_style:`.)

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

```

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

```

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

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

```

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

```

(End of definition for `__enumext_zero_count_level:`.)

__enumext_current_env_set_bool:
 __enumext_check_first_level:

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.

```

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

```

The function __enumext_check_first_level: will set the variables \l__enumext_standar_level_one_bool and \l__enumext_standar_level_one_bool to “true” only if the environment is not nested and we are at the “first level” of it.

```

252 \cs_new_protected:Nn \__enumext_check_first_level:
253 {
254   \bool_lazy_all:nT
255   {
256     { \bool_if_p:N \g__enumext_standar_bool }
257     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
258     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
259   }
260   {
261     \bool_set_true:N \l__enumext_standar_level_one_bool
262   }
263   \bool_lazy_all:nT
264   {
265     { \bool_if_p:N \g__enumext_starred_bool }
266     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
267     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
268   }
269   {
270     \bool_set_true:N \l__enumext_starred_level_one_bool
271   }
272 }

```

(End of definition for __enumext_current_env_set_bool: and __enumext_check_first_level:.)

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

10.6 Copying list and minipage environments

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

```

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

```

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

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w

```

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

```

273 \__enumext_at_begin_document:n
274 {
275   \cs_new_eq:NN \__enumext_start_list:nn \list
276   \cs_new_eq:NN \__enumext_stop_list: \endlist
277   \cs_new_eq:NN \__enumext_item_std:w \item
278 }

```

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

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

```

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

```

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

```

\__enumext_minipage:w
\__enumext_endminipage:

```

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

```

279 \__enumext_at_begin_document:n
280 {
281   \cs_new_eq:NN \__enumext_minipage:w \minipage
282   \cs_new_eq:NN \__enumext_endminipage: \endminipage
283 }

```

(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.

```

284 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
285 \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 `\l__enumext_footnotes_key_bool` to “true”.

```

286 \cs_new_protected:Nn \__enumext_after_hyperref:
287 {
288   \IfPackageLoadedTF { hyperref }
289   {
290     \msg_info:nnn { enumext } { package-load } { hyperref }
291     \bool_set_true:N \l__enumext_hyperref_bool
292     \IfHyperBoolean{hyperfootnotes}
293     {
294       \typeout{hyperfootnotes=true}
295       \bool_set_true:N \l__enumext_footnotes_key_bool
296     }
297     { \typeout{hyperfootnotes=false} }
298   }
299   { }

```

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`.

```

300   \bool_if:NT \l__enumext_footnotes_key_bool
301   {
302     \IfPackageLoadedTF { footnotehyper }
303     {
304       \msg_info:nnn { enumext } { package-load } { footnotehyper }
305     }
306     {
307       \typeout{No ~ footnotehyper ~ load}
308       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
309       \bool_set_false:N \l__enumext_footnotes_key_bool
310     }
311   }

```

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.

```

312 \bool_if:NTF \l__enumext_hyperref_bool
313 {
314   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
315   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
316 }
317 {
318   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
319   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
320 }
321 }

```

(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.

```

322 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
323 {
324   \protected@write \@auxout { }
325   {
326     \token_to_str:N \newlabel {#1}
327     {
328       {#2}
329       \bool_if:NT \l__enumext_hyperref_bool
330       { { \thepage } {#2} {#1} }
331       { }
332     }
333   }
334   \__enumext_hypertarget:nn {#1} { }
335   \__enumext_phantomsection:
336 }

```

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

10.8 Definition of counters

`__enumext_define_counters:Nn`

`__enumext_define_counters:cn`

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

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

#2: The counter’s name.

```

337 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
338 {
339   \cs_if_exist:cTF { c@ #2 }
340   { \msg_fatal:nnn { enumext } { counters } { #2 } }
341   {
342     \tl_set:Nn #1 { #2 }
343     \newcounter { #2 }
344   }
345 }

```

(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    346 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii   347 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii  348 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv   349 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXvii  350 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXviii 351 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
          352 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
          353 \__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.

```
354 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
355 {
356   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
357   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
358 }
359 \__enumext_register_counter_style:Nn \arabic { 0 }
360 \__enumext_register_counter_style:Nn \Alph { M }
361 \__enumext_register_counter_style:Nn \alph { m }
362 \__enumext_register_counter_style:Nn \Roman { VIII }
363 \__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.

```
364 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
365 {
366   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
367   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
368 }
369 \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.

```
370 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
371 {
372   \tl_clear_new:N #1
373   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
374   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
375   \tl_map_inline:Nn \g__enumext_counter_styles_tl
376   {
377     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
378     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
379     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
380   }
381   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
382   { \tl_use:N \g__enumext_widest_label_tl }
383   \tl_set_eq:cN { the #2 } #1
384 }
385 \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.

```
386 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
387 {
388   \keys_define:nn { enumext / #1 }
389   {
390     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
391     font      .value_required:n = true,
392     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
393     labelsep  .initial:n    = {0.3333em},
394     labelsep  .value_required:n = true,
395     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
```

```

396     labelwidth .value_required:n = true,
397     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
398     wrap-label .initial:n = {##1},
399     wrap-label .value_required:n = true,
400     wrap-label* .code:n = {
401         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
402         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
403     },
404     wrap-label* .value_required:n = true,
405 }
406 }
407 \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.

```

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 =
414             {
415                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
416                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
417             },
418         align / right .code:n =
419             {
420                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
421                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
422             },
423         align / center .code:n =
424             {
425                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
426                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
427             },
428         align .initial:n = left,
429         align .value_required:n = true,
430     }
431 }
432 \clist_map_inline:nn
433 {
434     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
435 }
436 { \__enumext_tmp:nn #1 }

437 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
438 {
439     \keys_define:nn { enumext / #1 }
440     {
441         align .choice:,
442         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
443         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
444         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
445         align .initial:n = left,
446         align .value_required:n = true,
447     }
448 }
449 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

10.11 Setting label and ref keys

The implementation of the keys `label` and `ref` are part of the core of the package `enumext`, here the default values for `\label`, the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “label and ref” system.

10.11.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl 450 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 451 {
\l__enumext_label_iii_tl 452   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 453   {
454     label .code:n = {
455       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
456       { l__enumext_counter_#2_tl } {##1}
457       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
458       \l__enumext_current_widest_dim
459     },
460     label .initial:n = #3,
461     label .value_required:n = true,
462     ref .code:n = \__enumext_standar_ref:n {##1},
463     ref .value_required:n = true,
464   }
465 }
466 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
467 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
468 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
469 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

```

470 \cs_new_protected:Npn \__enumext_standar_ref:n #1
471 {
472   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
473   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
474   {
475     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
476   }
477   {
478     \tl_set_eq:Nc
479     \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
480     \__enumext_regex_counter_style:
481     \tl_set_eq:Nc
482     \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
483     \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
484     {
485       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
486       { \exp_not:V \l__enumext_ref_key_arg_tl }
487     }
488   }
489 }

```

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

```

490 \cs_new_protected:Nn \__enumext_standar_ref:
491 {
492   \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
493   {
494     \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
495   }
496 }

```

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

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

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

```

\l__enumext_label_vii_tl 497 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 498 {

```

```

499 \keys_define:nn { enumext / #1 }
500 {
501   label .code:n = {
502     \__enumext_label_style:cvn { l__enumext_label_#2_tl }
503     { l__enumext_counter_#2_tl } {##1}
504     \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
505     \l__enumext_current_widest_dim
506   },
507   label .initial:n = #3,
508   label .value_required:n = true,
509   ref .code:n = \__enumext_starred_ref:n {##1},
510   ref .value_required:n = true,
511 }
512 }
513 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
514 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

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

```

515 \cs_new_protected:Npn \__enumext_starred_ref:n #1
516 {
517   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
518   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
519   {
520     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
521     {
522       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
523     }
524     {
525       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
526       \__enumext_regex_counter_style:
527       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
528       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
529       {
530         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
531         { \exp_not:V \l__enumext_ref_key_arg_tl }
532       }
533     }
534   }
535   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
536   {
537     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
538     {
539       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
540     }
541     {
542       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
543       \__enumext_regex_counter_style:
544       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
545       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
546       {
547         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
548         { \exp_not:V \l__enumext_ref_key_arg_tl }
549       }
550     }
551   }
552 }

```

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

```

553 \cs_new_protected:Npn \__enumext_starred_ref:
554 {
555   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
556   {
557     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
558     {
559       \tl_use:N \l__enumext_renew_the_count_vii_tl
560     }
561   }

```

```

562 \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
563 {
564   \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
565   {
566     \tl_use:N \l__enumext_renew_the_count_viii_tl
567   }
568 }
569 }

```

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

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

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

```

label \__enumext_label_v_tl
ref   \__enumext_label_vi_tl
570 \keys_define:nn { enumext / keyans }
571 {
572   label .code:n = {
573     \__enumext_label_style:cnv { \__enumext_label_v_tl }
574     { \__enumext_counter_v_tl } {#1}
575     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
576     \__enumext_current_widest_dim
577     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
578     { \__enumext_counter_vi_tl } {#1}
579     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
580     \__enumext_current_widest_dim
581   },
582   label .initial:n = (\Alph*),
583   label .value_required:n = true,
584   ref .code:n = \__enumext_keyans_ref:n {#1},
585   ref .value_required:n = true,
586 }

```

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

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

```

587 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
588 {
589   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
590   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
591   {
592     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
593   }
594   {
595     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
596     \__enumext_regex_counter_style:
597     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
598     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
599     {
600       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
601       { \exp_not:V \l__enumext_ref_key_arg_tl }
602     }
603   }
604 }

```

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

```

605 \cs_new_protected:Nn \__enumext_keyans_ref:
606 {
607   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
608   {
609     \tl_use:N \l__enumext_renew_the_count_v_tl
610   }
611 }

```

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

10.12 Setting start and widest keys

```
\__enumext_start_from:NNn
\__enumext_start_from:ccn
```

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

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

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

```
612 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
613 {
614   \__enumext_if_is_int:nTF { #3 }
615   {
616     \int_set:Nn #2 {#3}
617   }
618   {
619     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
620     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
621     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
622     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
623   }
624 }
625 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }
```

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

```
\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn
```

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

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

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

```
626 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
627 {
628   \__enumext_if_is_int:nTF {#4}
629   {
630     \setcounter{enumX#1} { #4 }
631   }
632   {
633     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
634     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
635     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
636     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
637   }
638   \__enumext_label_width_by_box:cv
639   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
640 }
641 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

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

```
start
widest
\l__enumext_start_X_int
```

```
642 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
643 {
644   \keys_define:nn { enumext / #1 }
645   {
646     start .code:n = {
647       \__enumext_start_from:ccn
648       { \l__enumext_label_#2_tl }
649       { \l__enumext_start_#2_int } {##1}
650     },
651     start .initial:n = 1,
652     widest .code:n = {
653       \__enumext_widest_from:nccn {#2}
654       { \l__enumext_label_#2_tl }
655       { \l__enumext_labelwidth_#2_dim } {##1}
656     },

```

```

657         widest .value_required:n = true,
658         start .value_required:n = true,
659     }
660 }
661 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `start`, `widest`, and `__enumext_start_X_int`.)

10.13 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
662 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
663 {
664     \keys_define:nn { enumext / #1 }
665     {
666         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
667         topsep .initial:n = {#3},
668         topsep .value_required:n = true,
669         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
670         partopsep .initial:n = {#4},
671         partopsep .value_required:n = true,
672         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
673         parsep .initial:n = {#5},
674         parsep .value_required:n = true,
675         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
676         itemsep .initial:n = {#6},
677         itemsep .value_required:n = true,
678         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
679         noitemsep .value_forbidden:n = true,
680         nosep .meta:n = {
681             itemsep = 0pt, parsep = 0pt,
682             topsep = 0pt, partopsep = 0pt,
683         },
684         nosep .value_forbidden:n = true,
685     }
686 }

```

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

```

687 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
688 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
689 { 4.0pt plus 2.0pt minus 1.0pt }
690 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
691 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
692 { 2.0pt plus 1.0pt minus 1.0pt }
693 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
694 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
695 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
696 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
697 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
698 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
699 { 2.0pt plus 1.0pt minus 1.0pt }
700 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
701 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
702 { 4.0pt plus 2.0pt minus 1.0pt }
703 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
704 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
705 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

10.14 Setting keys for horizontal spaces

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

```

706 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
707 {
708     \keys_define:nn { enumext / #1 }
709     {
710         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
711         itemindent .value_required:n = true,
712         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },

```

```

713     rightmargin .value_required:n = true,
714     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
715     listparindent .value_required:n = true,
716     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
717     list-offset .value_required:n = true,
718     list-indent .code:n =
719         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
720         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
721     list-indent .value_required:n = true,
722 }
723 }
724 \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.

```

725 \cs_set_protected:Npn \__enumext_tmp:n #1
726 {
727     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
728 }
729 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

10.14.1 Functions for setting the fake *itemindent*

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

```

730 \cs_set_protected:Nn \__enumext_fake_item:
731 {
732     \dim_compare:nNnT
733     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
734     >
735     { \c_zero_dim }
736     {
737         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
738         {
739             \exp_not:N \mode_leave_vertical:
740             \exp_not:n { \skip_horizontal:n }
741             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
742             \ignorespaces
743         }
744     }
745 }
746 \cs_set_protected:Nn \__enumext_keyans_fake_item:
747 {
748     \dim_compare:nNnT
749     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
750     {
751         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
752         {
753             \exp_not:N \mode_leave_vertical:
754             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
755         }
756     }
757 }
758 \cs_set_protected:Nn \__enumext_fake_item_vii:
759 {
760     \dim_compare:nNnT
761     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
762     {
763         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
764         {
765             \exp_not:N \mode_leave_vertical:
766             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
767         }
768     }
769 }
770 \cs_set_protected:Nn \__enumext_fake_item_viii:
771 {
772     \dim_compare:nNnT

```

```

773     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
774     {
775         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
776         {
777             \exp_not:N \mode_leave_vertical:
778             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
779         }
780     }
781 }

```

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

10.15 Setting show-length key

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

```

782 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
783 {
784     \keys_define:nn { enumext / #1 }
785     {
786         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
787         show-length .initial:n = false,
788     }
789 }
790 \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
791 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
792 {
793     \keys_define:nn { enumext / #1 }
794     {
795         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
796         before .value_required:n = true,
797         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
798         before* .value_required:n = true,
799         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
800         after .value_required:n = true,
801         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
802         first .value_required:n = true,
803     }
804 }
805 \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.

```

806 \cs_new_protected:Nn \__enumext_before_args_exec:
807 {
808     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
809 }

```

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

```

810 \cs_new_protected:Nn \__enumext_before_keys_exec:
811 {
812     \tl_use:c { \l__enumext_before_no_starred_key_ \__enumext_level: _tl }
813 }

```

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

```

814 \cs_new_protected:Nn \__enumext_after_stop_list:
815 {

```



```

816     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
817 }

```

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

```

818 \cs_new_protected:Nn \__enumext_after_args_exec:
819 {
820     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
821 }

```

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

10.16.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
822 \cs_new_protected:Nn \__enumext_before_args_exec_v:
823 {
824     \tl_use:N \l__enumext_before_starred_key_v_tl
825 }

```

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

```

826 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
827 {
828     \tl_use:N \l__enumext_before_no_starred_key_v_tl
829 }

```

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

```

830 \cs_new_protected:Nn \__enumext_after_stop_list_v:
831 {
832     \tl_use:N \l__enumext_after_stop_list_v_tl
833 }

```

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`.

```

834 \cs_new_protected:Nn \__enumext_after_args_exec_v:
835 {
836     \tl_use:N \l__enumext_after_list_args_v_tl
837 }

```

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

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

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

```

\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
838 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
839 {
840     \tl_use:N \l__enumext_before_starred_key_vii_tl
841 }
842 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
843 {
844     \tl_use:N \l__enumext_before_starred_key_viii_tl
845 }

```

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⟩`.

```

846 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
847 {
848     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
849 }
850 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
851 {
852     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
853 }

```

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

```

854 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
855 {
856   \tl_use:N \l__enumext_after_stop_list_vii_tl
857 }
858 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
859 {
860   \tl_use:N \l__enumext_after_stop_list_viii_tl
861 }

```

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`.

```

862 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
863 {
864   \tl_use:N \l__enumext_after_list_args_vii_tl
865 }
866 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
867 {
868   \tl_use:N \l__enumext_after_list_args_viii_tl
869 }

```

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

10.17 Setting keys for multicols and minipage

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

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

`columns-sep`

`columns`

```

870 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
871 {
872   \keys_define:nn { enumext / #1 }
873   {
874     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
875     mini-env .value_required:n = true,
876     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
877     mini-sep .initial:n = 0.3333em,
878     mini-sep .value_required:n = true,
879     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
880     columns-sep .value_required:n = true,
881     columns .int_set:c = { \l__enumext_columns_#2_int },
882     columns .initial:n = 1,
883     columns .value_required:n = true,
884   }
885 }
886 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

```

887 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
888 {
889   \keys_define:nn { enumext / #1 }
890   {
891     columns .initial:n = 2,
892     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
893     miniright .value_required:n = true,
894     miniright* .code:n = {
895       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
896       \keys_set:nn { enumext / #1 } { miniright = {##1} }
897     },
898     miniright* .value_required:n = true,
899   }
900 }
901 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for multicol

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



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

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

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

10.18.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

902 \cs_new_protected:Nn \__enumext_multi_set_vskip:
903 {
904   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip {
905     {
906       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
907         }
908       \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip {
909         {
910           \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
911             }
912           \__enumext_add_pre_parsep:
913         }
914       }
915     }
916   }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

914 \cs_new_protected:Nn \__enumext_add_pre_parsep:
915 {
916   \int_case:nn { \__enumext_level_int }
917   {
918     { 2 } {
919       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip } {
920         {
921           \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
922         }
923       }
924     { 3 } {
925       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip } {
926         {
927           \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
928         }
929       }
930     { 4 } {
931       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip } {
932         {
933           \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
934         }
935       }
936     }
937   }

```

```

936     }
937 }

```

(End of definition for `__enumext_add_pre_parse:`)

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

```

938 \cs_new_protected:Nn \__enumext_multi_addvspace:
939 {
940   \__enumext_multi_set_vskip:
941   \mode_if_vertical:T
942   {
943     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
944     {
945       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
946     }
947     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
948     {
949       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
950     }
951   }
952   \par\nopagebreak
953   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
954 }

```

(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:`

```

955 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
956 {
957   \skip_set:Nn \l__enumext_multicols_above_v_skip
958   {
959     \l__enumext_topsep_v_skip
960   }
961   \skip_set:Nn \l__enumext_multicols_below_v_skip
962   {
963     \l__enumext_topsep_v_skip
964   }
965 }
966 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
967 {
968   \__enumext_keyans_multi_set_vskip:
969   \mode_if_vertical:T
970   {
971     \skip_add:Nn \l__enumext_multicols_above_v_skip
972     {
973       \skip_use:N \l__enumext_partopsep_v_skip
974     }
975     \skip_add:Nn \l__enumext_multicols_below_v_skip
976     {
977       \skip_use:N \l__enumext_partopsep_v_skip
978     }
979   }
980   \par\nopagebreak
981   \addvspace{ \l__enumext_multicols_above_v_skip }
982 }

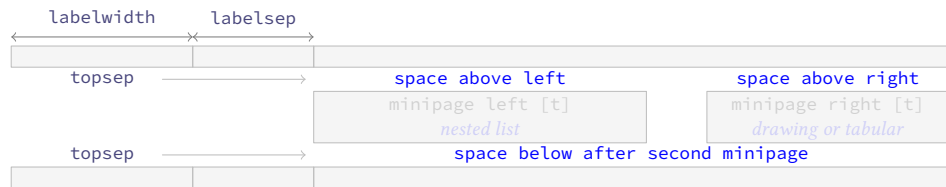
```

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

10.19 Adjustment of vertical spaces for minipage

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

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

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

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

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

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

```

983 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
984 {
985     \__enumext_minipage:w [ t ] { #1 }
986     \legacy_if_gset_false:n { @minipage }
987     \vspace { 0pt }
988 }
989 { \__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`.

```

990 \cs_new_protected:Nn \__enumext_mini_set_vskip:
991 {
992     \int_compare:nNnTF
993     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
994     {
```

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

```

995     \skip_if_eq:nnTF
996     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
997     {
998         \skip_set:Nn \l__enumext_minipage_left_skip
999         {
1000             -0.150\box_dp:N \strutbox
1001         }
1002         \skip_set:Nn \l__enumext_minipage_right_skip
1003         {
1004             0.695\box_dp:N \strutbox
1005         }
1006         \skip_set:Nn \l__enumext_minipage_after_skip
1007         {
1008             \box_dp:N \strutbox
1009         }
1010         \__enumext_zero_parsep:
1011     }
```

```

1012     {
1013         \skip_set:Nn \l__enumext_minipage_left_skip
1014         {
1015             \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1016         }
1017         \skip_set:Nn \l__enumext_minipage_right_skip
1018         {
1019             0.695\box_dp:N \strutbox
1020         }
1021         \skip_set:Nn \l__enumext_minipage_after_skip
1022         {
1023             1.85\box_dp:N \strutbox
1024             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1025         }
1026     }
1027 }
1028 {

```

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.

```

1029 \skip_if_eq:nnTF
1030 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1031 {
1032     \skip_set:Nn \l__enumext_minipage_left_skip
1033     {
1034         0.5\box_dp:N \strutbox
1035         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1036     }
1037     \skip_set:Nn \l__enumext_minipage_right_skip
1038     {
1039         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1040     }
1041     \skip_set:Nn \l__enumext_minipage_after_skip
1042     {
1043         1.6\box_dp:N \strutbox
1044     }
1045 }
1046 {
1047     \skip_set:Nn \l__enumext_minipage_left_skip
1048     {
1049         0.5875\box_dp:N \strutbox
1050         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1051     }
1052     \skip_set:Nn \l__enumext_minipage_right_skip
1053     {
1054         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1055         + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1056     }
1057     \skip_set:Nn \l__enumext_minipage_after_skip
1058     {
1059         0.325\box_dp:N \strutbox
1060         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1061     }
1062 }
1063 }
1064 }

```

(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.

```

1065 \cs_new_protected:Nn \__enumext_zero_parsep:
1066 {
1067     \int_case:nn { \l__enumext_level_int }
1068     {
1069         { 2 }{
1070             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1071             {
1072                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }

```

```

1073         }
1074     }
1075     { 3 }{
1076         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1077         {
1078             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1079         }
1080     }
1081     { 4 }{
1082         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1083         {
1084             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1085         }
1086     }
1087 }
1088 }

```

(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*.

```

1089 \cs_new_protected:Nn \__enumext_mini_addvspace:
1090 {
1091     \__enumext_mini_set_vskip:
1092     \mode_if_vertical:T
1093     {
1094         \skip_add:Nn \l__enumext_minipage_left_skip
1095         {
1096             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1097         }
1098         \skip_add:Nn \l__enumext_minipage_after_skip
1099         {
1100             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1101         }
1102     }
1103     \par\nopagebreak
1104     \addvspace { \l__enumext_minipage_left_skip }
1105 }

```

(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`.

```

1106 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1107 {
1108     \skip_zero_new:N \l__enumext_minipage_after_skip
1109     \skip_zero_new:N \l__enumext_minipage_left_skip
1110     \skip_zero_new:N \l__enumext_minipage_right_skip
1111     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1112     {
1113         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1114         {
1115             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1116             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1117             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1118             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1119             {
1120                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1121             }
1122         }
1123     }
1124     {
1125         \skip_set:Nn \l__enumext_minipage_left_skip
1126         {
1127             \skip_use:N \l__enumext_topsep_v_skip
1128         }
1129         \skip_set:Nn \l__enumext_minipage_right_skip

```



```

1129         {
1130             0.705\box_dp:N \strutbox
1131         }
1132         \skip_set:Nn \l__enumext_minipage_after_skip
1133         {
1134             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1135         }
1136     }
1137 }
1138 {
1139     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1140     {
1141         \skip_set:Nn \l__enumext_minipage_left_skip
1142         {
1143             0.5\box_dp:N \strutbox
1144             + \l__enumext_partopsep_v_skip
1145         }
1146         \skip_set:Nn \l__enumext_minipage_right_skip
1147         {
1148             \l__enumext_partopsep_v_skip
1149         }
1150         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1151     }
1152     {
1153         \skip_set:Nn \l__enumext_minipage_left_skip
1154         {
1155             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1156         }
1157         \skip_set:Nn \l__enumext_minipage_right_skip
1158         {
1159             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1160         }
1161         \skip_set:Nn \l__enumext_minipage_after_skip
1162         {
1163             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1164         }
1165     }
1166 }
1167 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

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

```

1168 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1169 {
1170     \__enumext_keyans_mini_set_vskip:
1171     \mode_if_vertical:T
1172     {
1173         \skip_add:Nn \l__enumext_minipage_left_skip
1174         {
1175             \l__enumext_partopsep_v_skip
1176         }
1177         \skip_add:Nn \l__enumext_minipage_after_skip
1178         {
1179             \l__enumext_partopsep_v_skip
1180         }
1181     }
1182     \par\nopagebreak
1183     \addvspace { \l__enumext_minipage_left_skip }
1184 }

```

(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*`.

```

1185 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1186 {
1187   \skip_zero_new:N \l__enumext_minipage_left_skip
1188   \skip_gzero_new:N \g__enumext_minipage_right_skip
1189   \skip_gzero_new:N \g__enumext_minipage_after_skip
1190   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1191   {
1192     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1193     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1194   }
1195   {
1196     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1197     \skip_gset:Nn \g__enumext_minipage_right_skip
1198       {
1199         \l__enumext_topsep_vii_skip
1200       }
1201     \skip_gset:Nn \g__enumext_minipage_after_skip
1202       {
1203         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1204       }
1205   }
1206 }
1207 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1208 {
1209   \skip_zero_new:N \l__enumext_minipage_after_skip
1210   \skip_zero_new:N \l__enumext_minipage_left_skip
1211   \skip_zero_new:N \l__enumext_minipage_right_skip
1212   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1213   {
1214     \skip_set:Nn \l__enumext_minipage_left_skip
1215       {
1216         0.5\box_dp:N \strutbox
1217       }
1218     \skip_set:Nn \l__enumext_minipage_right_skip
1219       {
1220         \l__enumext_partopsep_viii_skip
1221       }
1222     \skip_set:Nn \l__enumext_minipage_after_skip
1223       {
1224         1.6\box_dp:N \strutbox
1225       }
1226   }
1227   {
1228     \skip_set:Nn \l__enumext_minipage_left_skip
1229       {
1230         0.5875\box_dp:N \strutbox
1231       }
1232     \skip_set:Nn \l__enumext_minipage_right_skip
1233       {
1234         \l__enumext_topsep_viii_skip
1235       }
1236     \skip_set:Nn \l__enumext_minipage_after_skip
1237       {
1238         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1239       }
1240   }
1241 }

```

(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.

```

1242 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1243 {
1244   \__enumext_mini_set_vskip_vii:
1245   \par\nopagebreak
1246   \addvspace { \l__enumext_minipage_left_skip }
1247 }

```

```

1248 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1249 {
1250   \__enumext_mini_set_vskip_viii:
1251   \par\nopagebreak
1252   \addvspace { \__enumext_minipage_left_skip }
1253 }

```

(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.

```

1254 \NewDocumentCommand \miniright { s }
1255 {
1256   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1257   {
1258     \msg_error:nnn { enumext } { wrong-miniright-place }
1259   }
1260   \int_compare:nNt { \__enumext_level_int } = { 0 }
1261   {
1262     \msg_error:nnn { enumext } { wrong-miniright-place }
1263   }
1264   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1265   {
1266     \__enumext_keyans_mini_right_cmd:n {#1}
1267   }
1268   { \__enumext_mini_right_cmd:n {#1} }
1269 }

```

(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.

```

1270 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1271 {
1272   \dim_compare:nNtF
1273   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1274   {
1275     \__enumext_multicols_stop:
1276     \end{__enumext_mini_env*}
1277     \hfill
1278     \begin{__enumext_mini_env*}
1279     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1280     \par\addvspace { \__enumext_minipage_right_skip }
1281     \bool_if:nF {#1}
1282     {
1283       \centering
1284     }
1285     \int_gzero:N \g__enumext_minipage_stat_int
1286   }
1287   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1288 }

```

(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.

```

1289 \\cs_new_protected:Npn \\_enumext_keyans_mini_right_cmd:n #1
1290 {
1291   \\dim_compare:nNnTF { \\_enumext_minipage_right_v_dim } > { \\c_zero_dim }
1292   {
1293     \\_enumext_keyans_multicols_stop:
1294     \\end{\\_enumext_mini_env*}
1295     \\hfill
1296     \\begin{\\_enumext_mini_env*}{ \\_enumext_minipage_right_v_dim }
1297     \\par\\addvspace { \\_enumext_minipage_right_skip }
1298     \\bool_if:nF {#1}
1299     {
1300       \\centering
1301     }
1302     \\int_gzero:N \\g__enumext_minipage_stat_int
1303   }
1304   { \\msg_error:nnn { enumext } { wrong-miniright-use } }
1305 }

```

(End of definition for _enumext_keyans_mini_right_cmd:n.)

10.20 Setting above and below keys

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

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

```

1306 \\cs_set_protected:Npn \\_enumext_tmp:nn #1 #2
1307 {
1308   \\keys_define:nn { enumext / #1 }
1309   {
1310     above .skip_set:c = { \\_enumext_vspace_above_#2_skip },
1311     above .value_required:n = true,
1312     above* .code:n = \\bool_set_true:c { \\_enumext_vspace_a_star_#2_bool }
1313               \\keys_set:nn { enumext / #1 } { above = {##1} },
1314     above* .value_required:n = true,
1315     below .skip_set:c = { \\_enumext_vspace_below_#2_skip },
1316     below .value_required:n = true,
1317     below* .code:n = \\bool_set_true:c { \\_enumext_vspace_b_star_#2_bool }
1318               \\keys_set:nn { enumext / #1 } { below = {##1} },
1319     below* .value_required:n = true,
1320   }
1321 }
1322 \\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.

```

1323 \\cs_new_protected:Nn \\_enumext_vspace_above:
1324 {
1325   \\skip_if_eq:nnF
1326   { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } } { \\c_zero_skip }
1327   {
1328     \\bool_if:cTF { \\_enumext_vspace_a_star_ \\_enumext_level: _bool }
1329     {
1330       \\vspace*{ \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1331     }
1332     {
1333       \\vspace { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1334     }
1335   }
1336 }

```

(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.

```

1337 \cs_new_protected:Nn \__enumext_vspace_below:
1338 {
1339   \skip_if_eq:nnF
1340     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1341   {
1342     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1343     {
1344       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1345     }
1346     {
1347       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1348     }
1349   }
1350 }

```

(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.

```

1351 \cs_new_protected:Nn \__enumext_vspace_above_v:
1352 {
1353   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1354   {
1355     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1356     {
1357       \vspace*{ \l__enumext_vspace_above_v_skip }
1358     }
1359     { \vspace { \l__enumext_vspace_above_v_skip } }
1360   }
1361 }

```

(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.

```

1362 \cs_new_protected:Nn \__enumext_vspace_below_v:
1363 {
1364   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1365   {
1366     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1367     {
1368       \vspace*{ \l__enumext_vspace_below_v_skip }
1369     }
1370     { \vspace { \l__enumext_vspace_below_v_skip } }
1371   }
1372 }

```

(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`:

```

1373 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1374 {
1375   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1376   {
1377     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1378     {
1379       \vspace*{ \l__enumext_vspace_above_vii_skip }
1380     }
1381     { \vspace { \l__enumext_vspace_above_vii_skip } }
1382   }
1383 }
1384 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1385 {
1386   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1387   {

```

```

1388         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1389         {
1390             \vspace*{ \l__enumext_vspace_above_viii_skip }
1391         }
1392         { \vspace { \l__enumext_vspace_above_viii_skip } }
1393     }
1394 }

```

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

```

\__enumext_vspace_below_vii:
\__enumext_vspace_below_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.

```

1395 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1396 {
1397     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1398     {
1399         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1400         {
1401             \vspace*{ \l__enumext_vspace_below_vii_skip }
1402         }
1403         { \vspace { \l__enumext_vspace_below_vii_skip } }
1404     }
1405 }
1406 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1407 {
1408     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1409     {
1410         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1411         {
1412             \vspace*{ \l__enumext_vspace_below_viii_skip }
1413         }
1414         { \vspace { \l__enumext_vspace_below_viii_skip } }
1415     }
1416 }

```

(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 $\langle keys \rangle$.

```

series
resume
resume*

```

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

```

1417 \cs_set_protected:Npn \__enumext_tmp:n #1
1418 {
1419     \keys_define:nn { enumext / #1 }
1420     {
1421         series .str_set:N = \l__enumext_series_str,
1422         series .value_required:n = true,
1423         resume .code:n = \__enumext_resume_series:n {##1},
1424         resume* .code:n = \__enumext_resume_starred:,
1425         resume* .value_forbidden:n = true,
1426     }
1427 }
1428 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.21.1 Internal functions for series key

```

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

```

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

```

1429 \cs_new:Npn \__enumext_filter_series:n #1
1430 {
1431     \use:e
1432     {
1433         \keyval_parse:NNn
1434         \__enumext_filter_series_key:n
1435         \__enumext_filter_series_pair:nn {#1}
1436     }
1437 }

```

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.

```

1438 \cs_new:Npn \__enumext_filter_series_key:n #1
1439 {
1440   \str_case:nnF {#1}
1441   {
1442     { resume } {}
1443     { resume* } {}
1444   }
1445   { , { \exp_not:n {#1} } }
1446 }

```

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

```

1447 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1448 {
1449   \str_case:nnF {#1}
1450   {
1451     { series } {}
1452     { resume } {}
1453     { start } {}
1454     { save-ans } {}
1455     { save-key } {}
1456   }
1457   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1458 }

```

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

```

__enumext_parse_series:n
__enumext_resume_last:n

```

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

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

```

1459 \cs_new_protected:Npn \__enumext_parse_series:n #1
1460 {
1461   \str_if_empty:NTF \l__enumext_series_str
1462   {
1463     \bool_if:NF \l__enumext_resume_active_bool
1464     {
1465       \__enumext_resume_last:n {#1}
1466     }
1467   }
1468   {
1469     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1470     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1471     { \__enumext_filter_series:n {#1} }
1472     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1473     {
1474       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1475     }
1476   }
1477 }

```

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.

```

1478 \cs_new_protected:Npn \__enumext_resume_last:n #1
1479 {
1480   \bool_lazy_all:NNT \l__enumext_standar_level_one_bool
1481   {
1482     \tl_gclear:N \g__enumext_standar_series_tl
1483     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }

```



```

1484     }
1485     \bool_if:NT \l__enumext_starred_level_one_bool
1486     {
1487         \tl_gclear:N \g__enumext_starred_series_tl
1488         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1489     }
1490 }

```

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

10.21.2 Internal function to save counter value

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

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

```

1491 \cs_new_protected:Nn \__enumext_resume_save_counter:
1492 {
1493     \bool_if:NT \g__enumext_standar_bool
1494     {
1495         \tl_if_empty:NF \l__enumext_series_str
1496         {
1497             \int_gset_eq:cN
1498             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1499         }
1500         \tl_if_empty:NTF \l__enumext_resume_name_tl
1501         {
1502             \str_if_empty:NT \l__enumext_series_str
1503             {
1504                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1505             }
1506         }
1507         {
1508             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1509             {
1510                 \int_gset_eq:cN
1511                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1512             }
1513         }
1514         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1515         {
1516             \int_gset_eq:cN
1517             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1518         }
1519     }
1520     \bool_if:NT \g__enumext_starred_bool
1521     {
1522         \tl_if_empty:NF \l__enumext_series_str
1523         {
1524             \int_gset_eq:cN
1525             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1526         }
1527         \tl_if_empty:NTF \l__enumext_resume_name_tl
1528         {
1529             \str_if_empty:NT \l__enumext_series_str
1530             {
1531                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1532             }
1533         }
1534         {
1535             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1536             {
1537                 \int_gset_eq:cN
1538                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1539             }
1540         }
1541     }

```

```

1540     }
1541     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1542     {
1543         \int_gset_eq:cN
1544         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1545     }
1546 }
1547 }

```

(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.

```

1548 \cs_new_protected:Npn \__enumext_resume_series:n #1
1549 {
1550     \tl_if_empty:NTF {#1}
1551     {
1552         \__enumext_resume_counter:n { }
1553     }
1554     {
1555         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1556         {
1557             \__enumext_resume_counter:n {#1}
1558             \bool_if:NT \g__enumext_standar_bool
1559             {
1560                 \keys_set:nv { enumext / level-1 }
1561                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1562             }
1563             \bool_if:NT \g__enumext_starred_bool
1564             {
1565                 \keys_set:nv { enumext / enumext* }
1566                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1567             }
1568         }
1569         {
1570             \bool_if:NT \g__enumext_standar_bool
1571             {
1572                 \msg_error:nnn { enumext } { unknown-series } {#1}
1573             }
1574             \bool_if:NT \g__enumext_starred_bool
1575             {
1576                 \msg_error:nnn { enumext } { unknown-series } {#1}
1577             }
1578         }
1579     }
1580 }

```

(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`.

```

1581 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1582 {
1583     \bool_set_true:N \l__enumext_resume_active_bool
1584     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1585     \tl_if_empty:NTF \l__enumext_resume_name_tl
1586     {
1587         \__enumext_resume_counter:
1588     }

```

```

1589     {
1590         \__enumext_resume_counter_series:
1591     }
1592     \__enumext_resume_counter_save_ans:
1593 }

```

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.

```

1594 \cs_new_protected:Nn \__enumext_resume_counter:
1595 {
1596     \bool_if:NT \g__enumext_standar_bool
1597     {
1598         \int_gincr:N \g__enumext_resume_int
1599         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1600     }
1601     \bool_if:NT \g__enumext_starred_bool
1602     {
1603         \int_gincr:N \g__enumext_resume_vii_int
1604         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1605     }
1606 }

```

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.

```

1607 \cs_new_protected:Nn \__enumext_resume_counter_series:
1608 {
1609     \bool_if:NT \g__enumext_standar_bool
1610     {
1611         \int_set:Nn \l__enumext_start_i_int
1612         {
1613             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1614         }
1615     }
1616     \bool_if:NT \g__enumext_starred_bool
1617     {
1618         \int_set:Nn \l__enumext_start_vii_int
1619         {
1620             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1621         }
1622     }
1623 }

```

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.

```

1624 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1625 {
1626     \bool_lazy_and:nnT
1627     { \bool_if_p:N \l__enumext_standar_level_one_bool }
1628     { \bool_if_p:N \l__enumext_store_active_bool }
1629     {
1630         \int_set:Nn \l__enumext_start_i_int
1631         {
1632             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1633         }
1634     }
1635     \bool_lazy_and:nnT
1636     { \bool_if_p:N \l__enumext_starred_level_one_bool }
1637     { \bool_if_p:N \l__enumext_store_active_bool }
1638     {
1639         \int_set:Nn \l__enumext_start_vii_int
1640         {
1641             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1642         }
1643     }
1644 }

```

(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.

```

1645 \cs_new_protected:Npn \__enumext_resume_starred:
1646 {
1647   \bool_if:NT \g__enumext_standar_bool
1648   {
1649     \tl_if_empty:NF \g__enumext_standar_series_tl
1650     {
1651       \__enumext_resume_counter:n { }
1652       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1653     }
1654   }
1655   \bool_if:NT \g__enumext_starred_bool
1656   {
1657     \tl_if_empty:NF \g__enumext_starred_series_tl
1658     {
1659       \__enumext_resume_counter:n { }
1660       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1661     }
1662   }
1663 }

```

(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*`.

```

1664 \cs_set_protected:Npn \__enumext_tmp:n #1
1665 {
1666   \keys_define:nn { enumext / #1 }
1667   {
1668     save-ans .code:n = \__enumext_storing_set:n {##1},
1669     save-ans .value_required:n = true,
1670   }
1671 }
1672 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.22.1 Internal functions for save-ans key

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

`__enumext_storing_exec:`

```

1673 \cs_new_protected:Npn \__enumext_storing_set:n #1
1674 {
1675   \tl_set:Nx \l__enumext_store_name_tl {#1}
1676   \tl_if_empty:NTF \l__enumext_store_name_tl
1677   {
1678     \bool_if:NT \l__enumext_standar_level_one_bool
1679     {
1680       \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1681     }
1682     \bool_if:NT \l__enumext_starred_level_one_bool
1683     {
1684       \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1685     }
1686   }
1687   {
1688     \bool_if:NT \l__enumext_standar_level_one_bool
1689     {
1690       \msg_note:nnnV
1691       { enumext } { save-ans-ok } { enumext } \l__enumext_store_name_tl
1692       \__enumext_storing_exec:

```

```

1693     }
1694     \bool_if:NT \l__enumext_starred_level_one_bool
1695     {
1696         \msg_note:nnnV
1697         { enumext } { save-ans-ok } { enumext* } \l__enumext_store_name_tl
1698         \__enumext_storing_exec:
1699     }
1700 }
1701 }

```

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 $\langle prop list \rangle$ `\g__enumext_series_<store name>_prop` and the $\langle sequence \rangle$ `\g__enumext_series_<store name>_seq` will be created globally to “store content” in case they do not exist together with the integer variable `\g__enumext_series_<store name>_int` used by the keys `resume` and `resume*`.

```

1702 \cs_new_protected:Nn \__enumext_storing_exec:
1703 {
1704     \bool_set_true:N \l__enumext_store_active_bool
1705     \bool_set_true:N \l__enumext_store_ans_bool
1706     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1707     {
1708         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1709     }
1710     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1711     {
1712         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1713     }
1714     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1715     {
1716         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1717     }
1718 }

```

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

10.23 The check answer mechanism

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

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

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

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

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

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

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

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

10.23.1 Setting check-ans key

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

```
no-store 1719 \cs_set_protected:Npn \__enumext_tmp:n #1
1720 {
1721   \keys_define:nn { enumext / #1 }
1722   {
1723     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1724     check-ans .initial:n = false,
1725     no-store .code:n = {
1726       \bool_set_false:N \l__enumext_store_ans_bool
1727       \bool_set_false:N \l__enumext_check_ans_bool
1728     },
1729     no-store .value_forbidden:n = true,
1730   }
1731 }
1732 \clist_map_inline:nn
1733 {
1734   level-1, level-2, level-3, level-4, enumext*
1735 }
1736 { \__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.

```
1737 \cs_new_protected:Nn \__enumext_check_ans_set:
1738 {
1739   \int_case:nn { \l__enumext_level_int }
1740   {
1741     { 1 }{
1742       \bool_lazy_all:nT
1743       {
1744         { \bool_if_p:N \g__enumext_starred_bool }
1745         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1746       }
1747       {
1748         \int_gdecr:N \g__enumext_count_item_number_int
1749         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1750       }
1751     }
1752     { 2 }{
1753       \int_gdecr:N \g__enumext_count_item_number_int
1754     }
1755     { 3 }{
1756       \int_gdecr:N \g__enumext_count_item_number_int
1757     }
1758     { 4 }{
1759       \int_gdecr:N \g__enumext_count_item_number_int
1760     }
1761   }
1762   \int_case:nn { \l__enumext_level_h_int }
1763   {
1764     { 1 }{
1765       \bool_if:NT \g__enumext_standar_bool
1766       {
1767         \int_gdecr:N \g__enumext_count_item_number_int
1768         \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1769       }
1770     }
1771   }
1772 }
```

(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.

```

1773 \cs_new_protected:Nn \__enumext_check_ans_exec:
1774 {
1775   \bool_if:NT \l__enumext_check_ans_bool
1776   {
1777     \__enumext_check_ans_set:
1778   }
1779 }

```

(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.

```

1780 \cs_new_protected:Nn \__enumext_check_ans_show:
1781 {
1782   \int_compare:nNnTF
1783   { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1784   {
1785     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1786   }
1787   {
1788     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1789   }
1790   \int_gzero:N \g__enumext_count_item_number_int
1791   \int_gzero:N \g__enumext_count_item_anskey_int
1792 }

```

(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*`.

```

1793 \cs_set_protected:Npn \__enumext_tmp:n #1
1794 {
1795   \keys_define:nn { enumext / #1 }
1796   {
1797     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1798     wrap-ans .initial:n = \fbox{##1},
1799     wrap-ans .value_required:n = true,
1800     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1801     wrap-opt .initial:n = [{##1}],
1802     wrap-opt .value_required:n = true,
1803     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1804     save-sep .initial:n = {, ~ },
1805     save-sep .value_required:n = true,
1806     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1807     mark-ans .initial:n = \textasteriskcentered,
1808     mark-ans .value_required:n = true,
1809     mark-pos .choice:,
1810     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1811     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1812     mark-pos .initial:n = right,
1813     mark-pos .value_required:n = true,
1814     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1815     show-ans .initial:n = false,
1816     show-ans .value_required:n = true,
1817     show-pos .bool_set:N = \l__enumext_show_position_bool,
1818     show-pos .initial:n = false,
1819     show-pos .value_required:n = true,
1820     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
1821     mark-ref .initial:n = \textasteriskcentered,
1822     mark-ref .value_required:n = true,
1823     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1824     save-ref .initial:n = false,
1825     save-ref .value_required:n = true,
1826   }
1827 }
1828 \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`.

```

1829 \cs_set_protected:Npn \__enumext_tmp:n #1
1830 {
1831   \keys_define:nn { enumext / #1 }
1832   {
1833     mark-pos .choice:,
1834     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1835     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1836     mark-pos .initial:n = right,
1837     mark-pos .value_required:n = true,
1838     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1839     show-ans .initial:n = false,
1840     show-ans .value_required:n = true,
1841     show-pos .bool_set:N = \l__enumext_show_position_bool,
1842     show-pos .initial:n = false,
1843     show-pos .value_required:n = true,
1844   }
1845 }
1846 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1847 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1848 {
1849   \keys_define:nn { enumext / #1 }
1850   {
1851     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1852               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1853               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1854               {
1855                 columns = \exp_not:v { l__enumext_store_columns_#2_int },
1856               },
1857     columns* .value_required:n = true,
1858     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1859               \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1860               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1861               {
1862                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1863               },
1864     columns-sep* .value_required:n = true,
1865   }
1866 }
1867 \clist_map_inline:nn
1868 {
1869   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1870 }
1871 { \__enumext_tmp:nn #1 }

```

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

10.24.1 Function for storing content in prop list

`__enumext_store_addto_prop:n` The function `__enumext_store_addto_prop:n` stores the content in `⟨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 `{⟨position⟩}{⟨content⟩}`. This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic` in `keyanspic` environment.

```

1872 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1873 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1874 {
1875   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1876   {
1877     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1878   }
1879   { #1 }
1880 }

```

```
1881 \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 an 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.

```
1882 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1883 {
1884   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
1885 }
1886 \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 `__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`.

```
1887 \cs_new_protected:Npn \__enumext_store_level_open:
1888 {
1889   \bool_if:NT \__enumext_store_ans_bool
1890   {
1891     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1892     {
1893       \__enumext_store_addto_seq:n
1894       {
1895         \item \begin{enumext}
1896       }
1897     }
1898     {
1899       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1900       {
1901         \item \begin{enumext} [
1902       }
1903       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1904       {
1905         ]
1906       }
1907       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1908     }
1909   }
1910 }
1911 \cs_new_protected:Npn \__enumext_store_level_close:
1912 {
1913   \bool_if:NT \__enumext_store_ans_bool
1914   {
1915     \__enumext_store_addto_seq:n { \end{enumext} }
1916   }
1917 }
```

(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.

```
1918 \cs_new_protected:Npn \__enumext_store_level_open_vii:
1919 {
1920   \bool_if:NT \__enumext_store_ans_bool
1921   {
1922     \tl_if_empty:NTF \__enumext_store_opt_vii_tl
```

```

1923         {
1924             \__enumext_store_addto_seq:n
1925             {
1926                 \item \mode_leave_vertical:
1927                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1928                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1929             }
1930         }
1931     {
1932         \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1933         {
1934             \item \mode_leave_vertical:
1935             \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1936             \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1937         }
1938         \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1939         {
1940             ]
1941         }
1942         \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1943     }
1944 }
1945 }
1946 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1947 {
1948     \bool_if:NT \l__enumext_store_ans_bool
1949     {
1950         \__enumext_store_addto_seq:n { \end{enumext*} }
1951     }
1952 }

```

(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`

```

1953 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1954 {
1955     \mode_leave_vertical:
1956     \skip_horizontal:n { -\dim_use:N #2 }
1957     \makebox[\opt][ r ]
1958     {
1959         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1960         {
1961             \tl_use:N \l__enumext_mark_answer_sym_tl
1962         }
1963     }
1964     \skip_horizontal:n { \dim_use:N #2 }
1965 }
1966 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

10.25 The command \anskey and internal label and ref

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

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

```

1967 \keys_define:nn { enumext / anskey }
1968 {
1969     item-sym* .tl_set:N    = \l__enumext_store_item_symbol_tl,
1970     item-sym* .value_required:n = true,
1971     item-pos* .dim_set:N   = \l__enumext_store_item_symbol_sep_dim,

```

```

1972     item-pos* .value_required:n = true,
1973     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1974     store-brk .default:n = true,
1975     store-brk .value_forbidden:n = true,
1976 }

```

This command `\anskey` will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error. If the `check-ans` key is active, increment `\g__enumext_count_item_with_ans_int`, then call internal function `__enumext_store_anskey_code:nnnn` will “store content” in the `<sequence>` and in the `<prop list>`.

```

1977 \NewDocumentCommand \anskey { d() s o +m }
1978 {
1979   \bool_if:NF \l__enumext_store_active_bool
1980   {
1981     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1982   }
1983   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1984   {
1985     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1986   }
1987   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1988   {
1989     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1990   }
1991   \group_begin:
1992     \bool_if:NT \l__enumext_store_ans_bool
1993     {
1994       \bool_if:NT \l__enumext_check_ans_bool
1995       {
1996         \int_gincr:N \g__enumext_count_item_anskey_int
1997       }
1998       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1999     }
2000   \group_end:
2001 }

```

(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.

```

2002 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2003 {
2004   \__enumext_store_addto_prop:n {#4}
2005   \bool_if:NT \l__enumext_store_ref_key_bool
2006   {
2007     \__enumext_store_internal_ref:
2008   }
2009   \__enumext_store_anskey_show_left:n { #4 }

```

Now we start processing the optional arguments passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “store” in the `<sequence>`. First we clear the variable `\l__enumext_store_anskey_arg_tl` and process `[<key = val>]`, if the `store-brk` key is present and the command is running under `enumext` (not in the starred version) we will add `\columnbreak` and then `\item`.

```

2010   \tl_clear:N \l__enumext_store_anskey_arg_tl
2011   \tl_if_novalue:nF {#3}
2012   {
2013     \keys_set:nn { enumext / anskey } {#3}
2014   }
2015   \bool_lazy_and:nnT
2016   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2017   { \bool_not_p:n { \l__enumext_starred_bool } }
2018   {
2019     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2020   }
2021   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2022 \tl_if_novalue:nF {#1}
2023 {
2024   \int_set:Nn \l__enumext_store_columns_join_int {#1}
2025   \bool_if:NT \l__enumext_starred_bool
2026   {
2027     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2028     {
2029       ( \exp_not:V \l__enumext_store_columns_join_int )
2030     }
2031   }
2032 }

```

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`.

```

2033 \bool_if:nTF {#2}
2034 {
2035   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2036   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2037   {
2038     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2039     {
2040       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2041     }
2042   }
2043   \dim_compare:nT
2044   {
2045     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2046   }
2047   {
2048     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2049     {
2050       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2051     }
2052   }
2053   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2054 }
2055 {
2056   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2057 }

```

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

```

2058 \bool_lazy_and:nnT
2059 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2060 { \bool_if_p:N \l__enumext_hyperref_bool }
2061 {
2062   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2063   {
2064     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2065     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2066   }
2067 }
2068 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2069 }

```

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

`\l__enumext_store_internal_ref:`

The function `\l__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 ($\langle labels \rangle$), we do not want to get double dots in our references, then we will place this in the variable `\l__enumext_newlabel_arg_two_tl`.

```

2070 \cs_new_protected:Nn \l__enumext_store_internal_ref:
2071 {
2072   \cs_set_protected:Npn \l__enumext_tmp:n ##1
2073   {
2074     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2075     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2076     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }

```

```

2077     \tl_reverse:c { l__enumext_label_copy_##1_tl }
2078   }
2079   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2080   \cs_set:Npn \__enumext_tmp:n ##1
2081     { . \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

Here we need to analyse the cases where the environment is started with `enumext*` and if `\anskey` is running alone in it or if it is running in a nested `enumext` environment within the starting environment.

```

2082   \bool_lazy_all:nT
2083   {
2084     { \bool_if_p:N \g__enumext_starred_bool }
2085     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2086   }
2087   {
2088     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2089     { \tl_use:N \l__enumext_label_copy_vii_tl }
2090   }
2091   \bool_lazy_all:nT
2092   {
2093     { \bool_if_p:N \l__enumext_standar_bool }
2094     { \bool_if_p:N \g__enumext_starred_bool }
2095     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2096   }
2097   {
2098     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2099     {
2100       \tl_use:N \l__enumext_label_copy_vii_tl
2101       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2102     }
2103   }

```

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.

```

2104   \bool_lazy_all:nT
2105   {
2106     { \bool_if_p:N \l__enumext_standar_bool }
2107     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2108     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2109     { \bool_not_p:n { \l__enumext_starred_bool } }
2110   }
2111   {
2112     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2113     {
2114       \tl_use:N \l__enumext_label_copy_i_tl
2115       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2116     }
2117   }
2118   \cs_set:Npn \__enumext_tmp:n ##1
2119     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2120   \bool_lazy_all:nT
2121   {
2122     { \bool_if_p:N \l__enumext_standar_bool }
2123     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2124     { \bool_not_p:n { \g__enumext_starred_bool } }
2125     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2126   }
2127   {
2128     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2129     {
2130       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2131       . \tl_use:N \l__enumext_label_copy_vii_tl
2132     }
2133   }

```

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

```

2134   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2135   {
2136     \l__enumext_store_name_tl \c_colon_str
2137     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2138   }

```

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

```

2139 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2140 {
2141   \__enumext_newlabel:n
2142   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2143   { \l__enumext_newlabel_arg_two_tl }
2144 }
2145 \l__enumext_store_write_aux_file_tl
2146 }

```

(End of definition for `__enumext_store_internal_ref:.`)

`__enumext_store_anskey_show_wrap:n`

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

```

2147 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2148 {
2149   \par
2150   \bool_if:NT \l__enumext_starred_bool
2151   {
2152     \cs_set:Nn \__enumext_level: { vii }
2153   }
2154   \__enumext_print_keyans_box:cc
2155   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2156   { \l__enumext_labelsep_ \__enumext_level: _dim }
2157   \__enumext_anskey_wrapper:n { #1 }
2158 }

```

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

`__enumext_store_anskey_show_left:n`

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

```

2159 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2160 {
2161   \bool_if:NT \l__enumext_show_answer_bool
2162   {
2163     \__enumext_store_anskey_show_wrap:n { #1 }
2164   }
2165   \bool_if:NT \l__enumext_show_position_bool
2166   {
2167     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2168     {
2169       \group_begin:
2170       \exp_not:N \normalfont
2171       \exp_not:N \footnotesize [ \int_eval:n
2172       {
2173         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2174       }
2175       ]
2176       \group_end:
2177     }
2178     \__enumext_store_anskey_show_wrap:n { #1 }
2179   }
2180 }

```

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

10.26 Common functions for keyans, keyans* and keyanspic

10.26.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

2181 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2182 {

```



```

2183 \tl_clear:N \l__enumext_store_keyans_label_tl
2184 \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2185 {
2186   \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2187 }
2188 {
2189   \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2190 }
2191 \tl_if_novalue:nF { #1 }
2192 {
2193   % Set save-sep
2194   \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2195   {
2196     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2197   }
2198   \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2199 }
2200 \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2201 }

```

(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)`.

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.

```

2202 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2203 {
2204   \bool_if:NT \l__enumext_store_ref_key_bool
2205   {
2206     \cs_set_protected:Npn \__enumext_tmp:n ##1
2207     {
2208       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2209       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2210       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2211       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2212     }
2213     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2214     \__enumext_keyans_store_ref_aux_i:
2215   }
2216 }

```

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

```

2217 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2218 {
2219   \bool_if:NT \g__enumext_starred_bool
2220   {
2221     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2222   }
2223   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2224   {
2225     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2226     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2227   }
2228   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2229   {
2230     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2231     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2232   }
2233   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2234   {
2235     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2236     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2237   }

```

```

2238 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2239 {
2240   \l__enumext_store_name_tl \c_colon_str
2241   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2242 }
2243 \__enumext_keyans_store_ref_aux_ii:
2244 }

```

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.

```

2245 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2246 {
2247   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2248   {
2249     \__enumext_newlabel:nn
2250     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2251     { \l__enumext_newlabel_arg_two_tl }
2252   }
2253   \l__enumext_store_write_aux_file_tl
2254 }

```

(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.

```

2255 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2256 {
2257   \tl_clear:N \l__enumext_store_keyans_label_tl
2258   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2259   {
2260     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2261   }
2262   {
2263     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2264   }
2265   \tl_if_novalue:nF { #1 }
2266   {
2267     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2268     {
2269       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2270     }
2271     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2272   }
2273   \__enumext_keyans_addto_seq_link:
2274 }

```

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.

```

2275 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2276 {
2277   \bool_lazy_and:nnT
2278   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2279   { \bool_if_p:N \l__enumext_hyperref_bool }
2280   {
2281     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2282     {
2283       \hfill \exp_not:N \hyperlink
2284       {
2285         \exp_not:V \l__enumext_newlabel_arg_one_tl
2286       }
2287       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2288     }
2289   }

```

```

2289     }
2290     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2291     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2292     \bool_if:NT \l__enumext_check_ans_bool
2293     {
2294         \int_gincr:N \g__enumext_count_item_anskey_int
2295     }
2296 }

```

(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.

```

2297 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2298 {
2299     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2300     {
2301         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2302     }
2303     { \tl_gclear:N \g__enumext_check_ans_item_tl }
2304 }

```

(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.

```

2305 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2306 {
2307     \tl_if_novalue:nF { #1 }
2308     {
2309         \tl_set:Nc \l__enumext_keyans_item_opt_tl { #1 }
2310     }
2311     \bool_if:NT \l__enumext_show_answer_bool
2312     {
2313         \__enumext_keyans_show_ans:
2314     }
2315     \bool_if:NT \l__enumext_show_position_bool
2316     {
2317         \__enumext_keyans_show_pos:
2318     }
2319 }
2320 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2321 {
2322     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2323     {
2324         \bool_lazy_or:nnT
2325         { \bool_if_p:N \l__enumext_show_answer_bool }
2326         { \bool_if_p:N \l__enumext_show_position_bool }
2327         {
2328             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2329         }
2330     }
2331 }
2332 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2333 {
2334     \tl_put_left:Nn \l__enumext_label_v_tl
2335     {
2336         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2337     }
2338 }
2339 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2340 {
2341     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2342     {

```

```

2343     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2344     {
2345         \group_begin:
2346         \exp_not:N \normalfont
2347         \exp_not:N \footnotesize [ \int_eval:n
2348         {
2349             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2350         }
2351         ]
2352         \group_end:
2353     }
2354 }
2355 {
2356     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2357     {
2358         \group_begin:
2359         \exp_not:N \normalfont
2360         \exp_not:N \footnotesize [ \int_eval:n
2361         {
2362             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2363         }
2364         ]
2365         \group_end:
2366     }
2367 }
2368 \tl_put_left:Nn \l__enumext_label_v_tl
2369 {
2370     \__enumext_print_keyans_box:NN
2371     \l__enumext_labelwidth_i_dim
2372     \l__enumext_labelsep_i_dim
2373 }
2374 }

```

(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*
2375 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2376 {
2377     \keys_define:nn { enumext / #1 }
2378     {
2379         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2380         item-sym* .value_required:n = true,
2381         item-sym* .initial:n = { $\star$ },
2382         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2383         item-pos* .value_required:n = true,
2384     }
2385 }
2386 \clist_map_inline:nn
2387 {
2388     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2389 }
2390 { \__enumext_tmp:nn #1 }

```

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

10.28 Redefining `\footnote` command

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

```

2391 \cs_new_protected:Nn \__enumext_footnotetext:nn
2392 {
2393     \footnotetext[#1]{#2}
2394 }
2395 \cs_new_protected:Nn \__enumext_renew_footnote:
2396 {
2397     \seq_gclear:N \g__enumext_footnote_arg_seq

```

```

2398 \seq_gclear:N \g__enumext_footnote_int_seq
2399 \RenewDocumentCommand \footnote { o +m }
2400 {
2401   \tl_if_novalue:nTF {##1}
2402   {
2403     \stepcounter{footnote}
2404     \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2405   }
2406   {
2407     \int_gset:Nn \g__enumext_footnote_int { ##1 }
2408   }
2409   \footnotemark [ \g__enumext_footnote_int ]
2410   \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2411   \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2412 }
2413 }
2414 \cs_new_protected:Nn \__enumext_print_footnote:
2415 {
2416   \seq_if_empty:NF \g__enumext_footnote_int_seq
2417   {
2418     \seq_map_pairwise_function:NNN
2419     \g__enumext_footnote_int_seq
2420     \g__enumext_footnote_arg_seq
2421     \__enumext_footnotetext:nn
2422   }
2423 }

```

(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).

```

2424 \cs_new_protected:Npn \__enumext_default_item:n #1
2425 {
2426   \tl_if_novalue:nTF {#1}
2427   {
2428     \bool_if:NT \l__enumext_check_ans_bool
2429     {
2430       \int_gincr:N \g__enumext_count_item_number_int
2431     }
2432     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2433     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2434   }
2435   {
2436     \bool_set_eq:cc
2437     { l__enumext_wrap_label_ \__enumext_level: _bool }
2438     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2439     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2440   }
2441 }

```

(End of definition for __enumext_default_item:n.)

__enumext_starred_item:nn

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

#1: \l__enumext_item_symbol_X_tl

#2: \l__enumext_item_symbol_sep_X_dim

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

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

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

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

```

2442 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2443 {
2444   \tl_if_novalue:nF {#1}
2445   {
2446     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2447   }
2448   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2449   \tl_if_novalue:nTF {#2}
2450   {
2451     \dim_set_eq:cc
2452     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2453     { \__enumext_labelsep_ \__enumext_level: _dim }
2454   }
2455   {
2456     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2457   }
2458   \bool_if:NT \l__enumext_check_ans_bool
2459   {
2460     \int_gincr:N \g__enumext_count_item_number_int
2461   }
2462   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2463   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2464 }

```

(End of definition for `__enumext_starred_item:nn`)

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

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

```

2465 \cs_new_protected:Nn \__enumext_redefine_item:
2466 {
2467   \RenewDocumentCommand \item { s o o }
2468   {
2469     \bool_if:nTF {##1}
2470     {
2471       \__enumext_starred_item:nn {##2} {##3}
2472     }
2473     { \__enumext_default_item:n {##2} }
2474   }
2475 }

```

(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 `\label` next to the `[\langle content \rangle]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key.

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

```

2476 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2477 {
2478   \tl_if_novalue:nTF { #1 }
2479   {
2480     \bool_set_true:N \__enumext_wrap_label_v_bool
2481     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```

```

2482     }
2483     {
2484         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2485         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2486     }
2487 }

```

(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 `<label>`, 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 `<label>`.

```

2488 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2489 {
2490     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2491     \__enumext_keyans_show_left:n { #1 }
2492     \bool_set_true:N \l__enumext_wrap_label_v_bool
2493     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

2494     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2495     \__enumext_keyans_addto_prop:n { #1 }
2496     \__enumext_keyans_store_ref:
2497     \__enumext_keyans_addto_seq:n { #1 }
2498 }

```

(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 (§10.31.2).

```

2499 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2500 {
2501     \RenewDocumentCommand \item { s o }
2502     {
2503         \bool_if:nTF {##1}
2504         {
2505             \peek_remove_spaces:n
2506             {
2507                 \__enumext_keyans_starred_item:n {##2}
2508             }
2509         }
2510         {
2511             \__enumext_keyans_default_item:n {##2}
2512         }
2513     }
2514 }

```

(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.

```

2515 \cs_new_protected:Nn \__enumext_item_starred:
2516 {
2517     \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2518     {
2519         \mode_leave_vertical:
2520         \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2521         \makebox[ opt ][ r ]{ \g__enumext_item_symbol_tl }

```



```

2522         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2523     }
2524 }

```

(End of definition for __enumext_item_starred:.)

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

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

```

2525 \cs_new_protected:Nn \__enumext_make_label:
2526 {
2527     \RenewDocumentCommand \make_label { m }
2528     {
2529         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2530         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2531         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2532         {
2533             \__enumext_item_starred:
2534             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2535         }
2536         { ##1 }
2537         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2538         \tl_gclear:N \g__enumext_item_symbol_tl
2539     }
2540 }

```

(End of definition for __enumext_make_label:.)

10.30.2 Redefining \make_label for keyans

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

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

```

2541 \cs_new_protected:Nn \__enumext_keyans_make_label:
2542 {
2543     \RenewDocumentCommand \make_label { m }
2544     {
2545         \tl_use:N \l__enumext_label_fill_left_v_tl
2546         \tl_use:N \l__enumext_label_font_style_v_tl
2547         \bool_if:NTF \l__enumext_wrap_label_v_bool
2548         {
2549             \__enumext_wrapper_label_v:n { ##1 }
2550         }
2551         { ##1 }
2552         \tl_use:N \l__enumext_label_fill_right_v_tl
2553     }
2554 }

```

(End of definition for __enumext_keyans_make_label:.)

10.31 Second argument of the lists

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

10.31.1 Calculation of \leftmargin and \itemindent

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

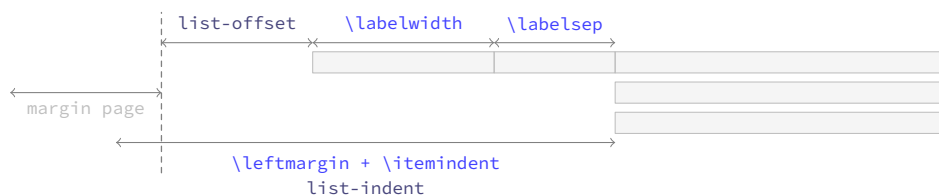
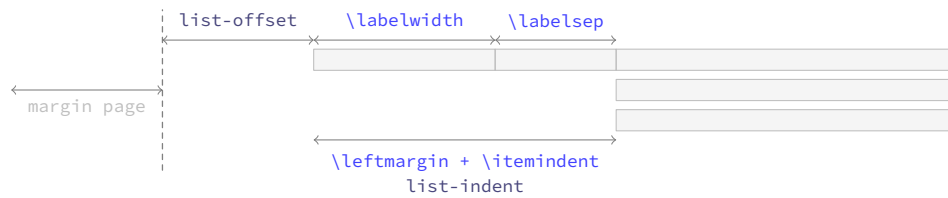
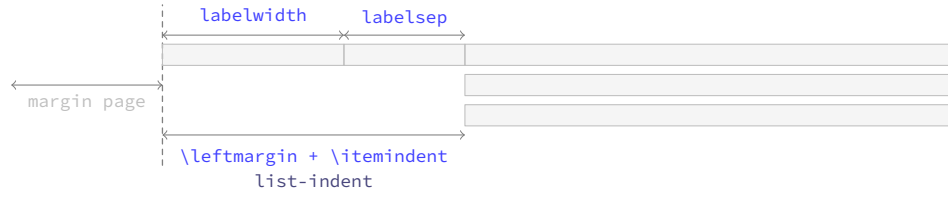


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

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

Figure 10: Representation of horizontal lengths concept in list in `enumext`.Figure 11: Default horizontal lengths in `enumext`.

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc
```

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

```
#1: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__enumext_leftmargin_tmp_X_bool
```

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

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

```
2555 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2556 {
2557   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2558   {
2559     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2560     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2561   }
2562   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2563   {
2564     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2565     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2566   }
2567 }
```

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

```
2567 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2568 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2569 {
2570   \dim_set:Nn #6 { #1 + #2 - #4 }
2571   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2572 }
2573 {
2574   \dim_compare:nNnT { #4 } = { #1 + #2 }
2575   { \dim_set:Nn #6 { \c_zero_dim } }
2576   \dim_compare:nNnT { #4 } < { #1 + #2 }
2577   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2578   \dim_compare:nNnT { #4 } > { #1 + #2 }
2579   {
2580     \dim_set:Nn #6 { -#1 - #2 + #4 }
2581     \dim_set:Nn #6 { #6*-1 }
2582   }
2583   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2584 }
2585 }
2586 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

10.31.2 Setting second argument of the lists

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

```

2587 \cs_set_protected:Npn \__enumext_tmp:n #1
2588 {
2589   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2590   {
2591     \__enumext_calc_hspace:cccccc
2592     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2593     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2594     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2595     { \__enumext_leftmargin_tmp_#1_bool }
2596     \clist_map_inline:nn
2597       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2598       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2599     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2600       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2601     \usecounter { enumX#1 }
2602     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2603     \str_if_eq:nnTF {#1} { v }
2604     {
2605       \__enumext_keyans_redefine_item:
2606       \__enumext_keyans_make_label:
2607       \__enumext_keyans_ref:
2608       \__enumext_keyans_fake_item:
2609       \bool_if:cT { \__enumext_show_length_#1_bool }
2610       {
2611         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2612       }
2613     }
2614     {
2615       \__enumext_redefine_item:
2616       \__enumext_make_label:
2617       \__enumext_standar_ref:
2618       \__enumext_fake_item:
2619       \bool_if:cT { \__enumext_show_length_#1_bool }
2620       {
2621         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2622       }
2623     }
2624   }
2625 }
2626 \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 `\listparindent` and `parsep` to set the value of `\parskip` locally.

```

2627 \cs_set_protected:Npn \__enumext_tmp:n #1
2628 {
2629   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2630   {
2631     \__enumext_calc_hspace:cccccc
2632     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2633     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2634     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2635     { \__enumext_leftmargin_tmp_#1_bool }
2636     \clist_map_inline:nn
2637       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2638       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2639     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2640       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2641     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2642     \skip_zero:N \partopsep
2643     \usecounter { enumX#1 }
2644     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2645     \__enumext_starred_ref:
2646     \str_if_eq:nnTF {#1} { vii }

```

```

2647     {
2648         \__enumext_fake_item_vii:
2649         \bool_if:cT { l__enumext_show_length_vii_bool }
2650         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2651     }
2652     {
2653         \__enumext_fake_item_viii:
2654         \bool_if:cT { l__enumext_show_length_#1_bool }
2655         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2656     }
2657 }
2658 }
2659 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

10.32 The environment enumext

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

```

2660 \NewDocumentEnvironment{enumext}{0}{ }
2661 {
2662     \__enumext_safe_exec:
2663     \__enumext_parse_keys:n {#1}
2664     \__enumext_before_list:
2665     \__enumext_start_store_level:
2666     \__enumext_start_list:nn
2667     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2668     {
2669         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2670         \__enumext_before_keys_exec:
2671     }
2672     \__enumext_after_args_exec:
2673 }
2674 {
2675     \__enumext_stop_list:
2676     \__enumext_stop_store_level:
2677     \__enumext_after_list:
2678 }

```

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

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

```

2679 \cs_new_protected:Nn \__enumext_safe_exec:
2680 {
2681     \__enumext_current_env_set_bool:
2682     \int_incr:N \l__enumext_level_int
2683     \int_compare:nNtF { \l__enumext_level_int } > { 4 }
2684     { \msg_fatal:nn { enumext } { list-too-deep } }
2685     \bool_set_true:N \l__enumext_standard_bool
2686     \__enumext_check_first_level:
2687 }

```

(End of definition for __enumext_safe_exec:.)

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

```

2688 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2689 {
2690     \tl_if_novalue:nF {#1}
2691     {
2692         \str_clear:N \l__enumext_series_str
2693         \int_compare:nNtF { \l__enumext_level_int } = { 1 }

```

```

2694     {
2695         \keys_set:nn { enumext / level-1 } {#1}
2696         \__enumext_parse_series:n {#1}
2697     }
2698     {
2699         \exp_args:Ne \keys_set:nn
2700         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2701     }
2702     \bool_if:NT \l__enumext_store_active_bool
2703     {
2704         \__enumext_parse_store_keys:n {#1}
2705     }
2706 }
2707 }

```

(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:`.

```

2708 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2709 {
2710     \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2711     {
2712         \regex_match:nnT { \b columns\b } {#1}
2713         {
2714             \int_set_eq:cc
2715             { l__enumext_store_columns_ \__enumext_level: _int }
2716             { l__enumext_columns_ \__enumext_level: _int }
2717             \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2718             {
2719                 columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2720             }
2721         }
2722     }
2723     \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2724     {
2725         \regex_match:nnT { \b columns-sep\b } {#1}
2726         {
2727             \dim_set_eq:cc
2728             { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2729             { l__enumext_columns_sep_ \__enumext_level: _dim }
2730             \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2731             {
2732                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2733             }
2734         }
2735     }
2736 }

```

(End of definition for __enumext_parse_store_keys:n.)

__enumext_start_store_level:

__enumext_stop_store_level:

The `__enumext_start_store_level:` and `__enumext_stop_store_level:` functions activate the level saving mechanism for storage in *sequence* of the `\anskey` command.

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

```

2737 \cs_new_protected:Nn \__enumext_start_store_level:
2738 {
2739     \bool_lazy_all:nT
2740     {
2741         { \bool_if_p:N \l__enumext_store_active_bool }
2742         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2743         { \bool_not_p:n { \g__enumext_starred_bool } }
2744     }
2745     {
2746         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2747         {
2748             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2749             \__enumext_store_level_open:
2750         }
2751     }

```

```

2751     }
2752     \bool_lazy_all:nT
2753     {
2754         { \bool_if_p:N \l__enumext_store_active_bool }
2755         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2756         { \bool_if_p:N \g__enumext_starred_bool }
2757     }
2758     {
2759         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2760         {
2761             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2762             \__enumext_store_level_open:
2763         }
2764     }
2765 }
2766 \cs_new_protected:Nn \__enumext_stop_store_level:
2767 {
2768     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2769     {
2770         \__enumext_store_level_close:
2771     }
2772 }

```

(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.

```

2773 \cs_new_protected:Nn \__enumext_before_list:
2774 {
2775     \__enumext_vspace_above:
2776     \__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.

```

2777     \__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.

```

2778     \dim_compare:nNnT
2779     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2780     {
2781         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2782         {
2783             \linewidth
2784             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2785             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2786         }

```

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

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

```

2787         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2788         \int_gincr:N \g__enumext_minipage_stat_int
2789         \__enumext_mini_addvspace:
2790         \nointerlineskip\noindent
2791         \begin{\__enumext_mini_env*}
2792         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2793     }
2794     \__enumext_multicols_start:
2795 }

```

(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.

```

2796 \cs_new_protected:Nn \__enumext_multicols_start:
2797 {
2798   \int_compare:nNt
2799     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2800   {
2801     \dim_compare:nNt
2802       { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2803     {
2804       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
2805       {
2806         ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
2807           + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
2808         ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
2809         - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
2810       }
2811     }
2812     \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
2813     \skip_zero:N \multicolsep
2814     \int_compare:nNt { \__enumext_level_int } > { 1 }
2815     {
2816       \dim_zero:N \columnseprule
2817     }

```

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.

```

2818   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2819   {
2820     \__enumext_multi_addvspace:
2821   }
2822   \raggedcolumns
2823   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2824 }
2825 }

```

(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.

```

2826 \cs_new_protected:Nn \__enumext_multicols_stop:
2827 {
2828   \int_compare:nNt
2829     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2830   {
2831     \end{multicols}
2832     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2833     {
2834       \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2835     }
2836   }
2837 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_after_list:` The function `__enumext_after_list:` will check the state of the boolean variable `__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* `__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

2838 \cs_new_protected:Nn \__enumext_after_list:
2839 {
2840   \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
2841   {
2842     \int_compare:nNt { \__enumext_minipage_stat_int } = { 1 }
2843     {

```



```

2844         \msg_warning:nn { enumext } { missing-miniright }
2845         \miniright
2846     }
2847     \int_gzero:N \g__enumext_minipage_stat_int
2848     \end{__enumext_mini_env*}
2849     \par\addvspace { \l__enumext_minipage_after_skip }
2850 }
2851 { \__enumext_multicols_stop: }

```

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

```

2852     \bool_lazy_and:nnT
2853     { \bool_if_p:N \l__enumext_check_ans_bool }
2854     { \bool_not_p:n { \g__enumext_starred_bool } }
2855     {
2856         \bool_gset_true:N \g__enumext_check_ans_show_bool
2857         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2858     }

```

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

```

2859     \__enumext_after_stop_list:
2860     \__enumext_vspace_below:
2861     \bool_set_false:N \l__enumext_standar_bool
2862     \__enumext_resume_save_counter:
2863 }

```

(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`.

```

2864 \__enumext_after_env:nn {enumext}
2865 {
2866     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2867     {
2868         \bool_if:NT \g__enumext_check_ans_show_bool
2869         {
2870             \__enumext_check_ans_show:
2871         }
2872         \bool_gset_false:N \g__enumext_standar_bool
2873         \bool_gset_false:N \g__enumext_check_ans_show_bool
2874         \tl_gclear:N \g__enumext_store_name_tl
2875     }
2876 }

```

10.33 The environment `keyans`

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “*multiple choice questions*”.

keyans Now we define the environment `keyans` also based on lists.

```

2877 \NewDocumentEnvironment{keyans}{ 0{} }
2878 {
2879     \__enumext_keyans_safe_exec:
2880     \__enumext_keyans_parse_keys:n {#1}
2881     \__enumext_before_list_v:
2882     \__enumext_start_list:nn
2883     { \tl_use:N \l__enumext_label_v_tl }
2884     {
2885         \__enumext_list_arg_two_v:
2886         \__enumext_before_keys_exec_v:
2887     }
2888     \__enumext_after_args_exec_v:
2889 }
2890 {
2891     \__enumext_keyans_check_ans:nn { item }{ keyans }
2892     \__enumext_stop_list:
2893     \__enumext_after_list_v:
2894 }

```

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

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

```

2895 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2896 {
2897   \bool_if:NF \l__enumext_store_active_bool
2898   {
2899     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
2900   }
2901   \int_incr:N \l__enumext_keyans_level_int
2902   \bool_set_true:N \l__enumext_keyans_env_bool
2903   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2904   \bool_set_false:N \l__enumext_store_active_bool
2905   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2906   {
2907     \msg_error:nn { enumext } { keyans-nested }
2908   }
2909   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2910   {
2911     \msg_error:nn { enumext } { keyans-wrong-level }
2912   }
2913 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

2914 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2915 {
2916   \keys_set:nn { enumext / keyans } { #1 }
2917 }

```

(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.

```

2918 \cs_new_protected:Nn \__enumext_before_list_v:
2919 {
2920   \__enumext_vspace_above_v:
2921   \__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.

```

2922   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2923   {
2924     \dim_set:Nn \l__enumext_minipage_left_v_dim
2925     {
2926       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2927     }

```

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

```

2928     \bool_set_true:N \l__enumext_minipage_active_v_bool
2929     \int_gincr:N \g__enumext_minipage_stat_int
2930     \__enumext_keyans_mini_addvspace:
2931     \nointerlineskip\noindent
2932     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2933   }

```

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

```

2934   \__enumext_keyans_multicols_start:
2935 }

```

(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.

```

2936 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2937 {
2938   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
2939   {

```

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

```

2940     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
2941     {
2942       \dim_set:Nn \__enumext_columns_sep_v_dim
2943       {
2944         (
2945           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
2946         ) / \__enumext_columns_v_int
2947         - \__enumext_listoffset_v_dim
2948       }
2949     }
2950     \dim_set_eq:NN \columnsep \__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).

```

2951     \skip_zero:N \multicolsep
2952     \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.

```

2953     \bool_if:NF \__enumext_minipage_active_v_bool
2954     {
2955       \__enumext_keyans_multi_addvspace:
2956     }
2957     \raggedcolumns
2958     \begin{multicols}{ \__enumext_columns_v_int }
2959   }
2960 }

```

(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 `__enumext_minipage_active_v_bool` is false (not nested in `__enumext_minipage*`) we will apply our vertical “adjust” spacing.

```

2961 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2962 {
2963   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
2964   {
2965     \end{multicols}
2966     \bool_if:NF \__enumext_minipage_active_v_bool
2967     {
2968       \par\addvspace{ \__enumext_multicols_below_v_skip }
2969     }
2970   }
2971 }

```

(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 `__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_minipage*` environment has not been closed), then close `__enumext_minipage*` and add the vertical adjustment space `__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

2972 \cs_new_protected:Nn \__enumext_after_list_v:
2973 {
2974   \bool_if:NTF \__enumext_minipage_active_v_bool
2975   {
2976     \int_compare:nNt { \__enumext_minipage_stat_int } = { 1 }
2977     {
2978       \msg_warning:nn { enumext } { missing-miniright }
2979       \miniright
2980     }
2981     \int_gzero:N \__enumext_minipage_stat_int

```

```

2982         \end{__enumext_mini_env*}
2983         \par\addvspace{ \l__enumext_minipage_after_skip }
2984     }
2985     { __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.

```

2986     \bool_set_false:N \l__enumext_keyans_env_bool
2987     \__enumext_after_stop_list_v:
2988     \__enumext_vspace_below_v:
2989 }

```

(End of definition for `__enumext_after_list_v:`)

10.34 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a list-based environment that uses the same configuration for “*spacing*” and `\label` as the `keyans` environment, but it does not use `\item`.

The contents are passed to the environment by means of the `\anspic` command and are placed inside `minipage` environments, with the `\label` underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to “adjust” the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.

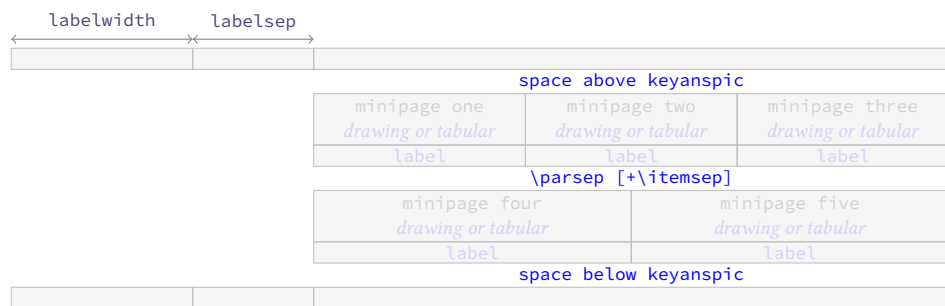


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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?](#).

10.34.1 The command `\anspic`

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

```

2990 \NewDocumentCommand \anspic { s o +m }
2991 {

```

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

```

2992     \bool_if:NF \l__enumext_store_active_bool
2993     {
2994         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2995     }
2996     \int_compare:nNt { \l__enumext_level_int } > { 1 }
2997     {
2998         \msg_error:nn { enumext } { keyanspic-wrong-level }
2999     }
3000     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3001     {
3002         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3003     }

```

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

```

3004     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3005     {
3006         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3007     }
3008 }

```

(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.

```

3009 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3010 {
3011   \stepcounter { enumXvi }
3012   #3 \\\
3013   \bool_if:nT { #1 }
3014   {
3015     \__enumext_keyans_addto_prop:n { #2 }
3016     \__enumext_keyans_store_ref:
3017     \__enumext_keyans_addto_seq:n { #2 }
3018     \bool_lazy_or:nnT
3019     { \bool_if_p:N \l__enumext_show_answer_bool }
3020     { \bool_if_p:N \l__enumext_show_position_bool }
3021     {
3022       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3023       \__enumext_keyans_show_left:n { #2 }
3024       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3025     }
3026   }
3027   \tl_use:N \l__enumext_label_font_style_v_tl
3028   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3029 }

```

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

10.34.2 The environment `keyanspic`

`keyanspic` Now we define the environment `keyanspic` based on list. The optional argument [*⟨number above, number below⟩*] will determine the number of `minipage` environments that will be above and below separated by `\parsep+ \itemsep` within it.

```

3030 \NewDocumentEnvironment{keyanspic}{ o }
3031 {
3032   \__enumext_keyans_pic_safe_exec:
3033   \__enumext_start_list:nn
3034   { }
3035   {
3036     \__enumext_keyans_pic_arg_two:
3037   }

```

We apply the “adjusted” vertical spacing above the environment

```

3038   \vspace { \l__enumext_keyans_pic_above_skip }
3039 }

```

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.

```

3040 {
3041   \tl_if_novalue:nTF { #1 }
3042   {
3043     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3044   }
3045   { \__enumext_keyans_pic_do:n { #1 } }
3046   \__enumext_stop_list:
3047   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
3048   \setcounter { enumXvi } { 0 }
3049   \vspace { \l__enumext_topsep_v_skip }
3050   %\bool_set_false:N \l__enumext_store_active_bool
3051 }

```

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

`__enumext_keyans_pic_safe_exec:`

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

```

3052 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3053 {
3054   \int_incr:N \l__enumext_keyans_pic_level_int
3055   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3056   {
3057     \msg_error:nn { enumext } { keyanspic-nested }
3058   }
3059 }

```

(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`.

```
3060 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3061 {
3062   \dim_compare:nNt { #1 } < { 0pt }
3063   { \skip_set:Nn #1 { -#1 } }
3064 }
```

(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.

```
3065 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3066 {
```

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.

```
3067   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3068   \__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.

```
3069   \skip_add:Nn \parsep { \itemsep }
3070   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3071   \dim_zero:N \labelwidth
3072   \dim_zero:N \listparindent
3073   \dim_zero:N \labelsep
3074   \skip_zero:N \partopsep
3075   \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 \TeX when not using the `\item` command.

```
3076   \__enumext_keyans_pic_skip_abs:N \parsep
3077   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3078   {
3079     \box_dp:N \strutbox
3080     + \l__enumext_topsep_v_skip
3081     - \parsep
3082   }
3083   \__enumext_item_std:w \scan_stop:
3084 }
```

(End of definition for `__enumext_keyans_pic_arg_two:.`)

`__enumext_keyans_pic_do:n` 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`.

```
3085 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3086 {
3087   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3088 }
3089 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }
```

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

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

```
3090 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3091 {
3092   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3093   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3094   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3095   \int_step_inline:nnn
3096   { \l__enumext_keyans_pic_above_int + 1 }
3097   { \l__enumext_keyans_pic_below_int }
3098   {
```

```

3099     \__enumext_minipage:w [ b ]{ \__enumext_keyans_pic_width_dim }
3100     \centering
3101     \seq_item:Nn \__enumext_keyans_pic_body_seq { ##1 }
3102     \__enumext_endminipage:
3103   }
3104 \par
3105 }

```

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

10.35 The environment `enumext*`

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

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

10.35.1 Functions for item box width

`__enumext_starred_columns_set_vii:`

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

```

3106 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3107 {
3108   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3109   {
3110     \dim_set:Nn \__enumext_columns_sep_vii_dim
3111     {
3112       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
3113       / \__enumext_columns_vii_int
3114     }
3115   }
3116   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - \c_one_int }
3117   \dim_set:Nn \__enumext_item_width_vii_dim
3118   {
3119     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
3120     / \__enumext_columns_vii_int - \__enumext_labelwidth_vii_dim
3121     - \__enumext_labelsep_vii_dim
3122   }
3123   \dim_zero_new:N \itemwidth
3124 }

```

(End of definition for `__enumext_starred_columns_set_vii:`.)

`__enumext_starred_joined_item_vii:n`

The function `__enumext_starred_joined_item_vii:n` will set the *width* of the box in which the content passed to `\item\langle number \rangle` will be stored together with the value of `\itemwidth`.

```

3125 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3126 {
3127   \int_set:Nn \__enumext_joined_item_vii_int {#1}
3128   \int_compare:nNnT { \__enumext_joined_item_vii_int } > { \__enumext_columns_vii_int }
3129   {
3130     \msg_warning:nnee { enumext } { item-joined }
3131     { \int_use:N \__enumext_joined_item_vii_int }
3132     { \int_use:N \__enumext_columns_vii_int }
3133     \int_set:Nn \__enumext_joined_item_vii_int
3134     {
3135       \__enumext_columns_vii_int - \__enumext_item_column_pos_vii_int + \c_one_int
3136     }
3137   }
3138   \int_compare:nNnT
3139   { \__enumext_joined_item_vii_int }
3140   >
3141   { \__enumext_columns_vii_int - \__enumext_item_column_pos_vii_int + \c_one_int }
3142   {
3143     \msg_warning:nnee { enumext } { item-joined-columns }
3144     { \int_use:N \__enumext_joined_item_vii_int }
3145     {

```



```

3146         \int_eval:n
3147         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3148     }
3149     \int_set:Nn \l__enumext_joined_item_vii_int
3150     {
3151         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3152     }
3153 }

```

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

```

3154     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3155     {
3156         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3157         \int_decr:N \l__enumext_joined_item_aux_vii_int
3158         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3159         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3160         \dim_set:Nn \l__enumext_joined_width_vii_dim
3161         {
3162             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3163             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3164                 + \l__enumext_columns_sep_vii_dim
3165                 ) * \l__enumext_joined_item_aux_vii_int
3166         }
3167         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3168     }
3169     {
3170         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3171         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3172     }
3173 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n.)

__enumext_start_mini_vii:

The implementation of the mini-env key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

3174 \cs_new_protected:Nn \__enumext_start_mini_vii:
3175 {
3176     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3177     {
3178         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3179         {
3180             \linewidth
3181             - \l__enumext_minipage_right_vii_dim
3182             - \l__enumext_minipage_hsep_vii_dim
3183         }
3184         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3185         \dim_gset_eq:NN
3186             \g__enumext_minipage_right_vii_dim
3187             \l__enumext_minipage_right_vii_dim
3188         \__enumext_mini_advspace_vii:
3189         \nointerlineskip\noindent
3190         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3191     }
3192 }

```

(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:n` to execute the `__enumext-mini_env*` on the “right side”.

```

3193 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3194 {
3195     \bool_if:NT \l__enumext_minipage_active_vii_bool
3196     {
3197         \end{__enumext_mini_env*}
3198         \hfill
3199         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3200     }
3201 }

```

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”.

```

3202 \__enumext_after_env:nn {enumext*}
3203 {
3204   \bool_if:NT \g__enumext_minipage_active_vii_bool
3205   {
3206     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3207     \par\addvspace { \g__enumext_minipage_right_skip }
3208     \bool_if:NF \g__enumext_minipage_center_vii_bool
3209     {
3210       \centering
3211     }
3212     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3213     \end{__enumext_mini_env*}
3214     \par\addvspace{ \g__enumext_minipage_after_skip }
3215   }
3216   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3217   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3218   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3219   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3220   \bool_gset_false:N \g__enumext_starred_bool
3221 }

```

(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.

```

3222 \NewDocumentEnvironment{enumext*}{ o }
3223 {
3224   \__enumext_safe_exec_vii:
3225   \__enumext_parse_keys_vii:n {#1}
3226   \__enumext_before_list_vii:
3227   \__enumext_start_store_level_vii:
3228   \__enumext_start_list:nn { }
3229   {
3230     \__enumext_list_arg_two_vii:
3231     \__enumext_before_keys_exec_vii:
3232   }
3233   \__enumext_starred_columns_set_vii:
3234   \item[] \scan_stop:
3235   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3236   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3237 }
3238 {
3239   \__enumext_stop_item_tmp_vii:
3240   \__enumext_remove_extra_parsep_vii:
3241   \__enumext_stop_list:
3242   \__enumext_stop_store_level_vii:
3243   \__enumext_after_list_vii:
3244 }

```

(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`.

```

3245 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3246 {
3247   \__enumext_current_env_set_bool:
3248   \int_incr:N \l__enumext_level_h_int
3249   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3250   {
3251     \msg_error:nn { enumext } { nested }
3252   }
3253   \bool_set_true:N \l__enumext_starred_bool
3254   \__enumext_check_first_level:
3255 }

```

(End of definition for `__enumext_safe_exec_vii:`)

`__enumext_parse_keys_vii:n`

Parse [*key = val*] for `enumext*`. If the variable `__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.

```

3256 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3257 {
3258   \tl_if_novalue:nF {#1}
3259   {
3260     \str_clear:N \__enumext_series_str
3261     \keys_set:nn { enumext / enumext* } {#1}
3262     \__enumext_parse_series:n {#1}
3263     \bool_if:NT \__enumext_store_active_bool
3264     {
3265       \__enumext_parse_store_keys_vii:n {#1}
3266     }
3267   }
3268 }

```

(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 `__enumext_store_opt_vii_tl` which is used by the function `__enumext_store_level_open_vii:`.

```

3269 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3270 {
3271   \bool_if:NF \__enumext_store_columns_vii_bool
3272   {
3273     \regex_match:nnT { \b columns\b } {#1}
3274     {
3275       \int_set_eq:NN
3276       \__enumext_store_columns_vii_int
3277       \__enumext_columns_vii_int
3278       \tl_put_right:Ne \__enumext_store_opt_vii_tl
3279       {
3280         columns = \exp_not:V \__enumext_store_columns_vii_int ,
3281       }
3282     }
3283   }
3284   \bool_if:NF \__enumext_store_columns_sep_vii_bool
3285   {
3286     \regex_match:nnT { \b columns-sep\b } {#1}
3287     {
3288       \dim_set_eq:NN
3289       \__enumext_store_columns_sep_vii_dim
3290       \__enumext_columns_sep_vii_dim
3291       \tl_put_right:Ne \__enumext_store_opt_vii_tl
3292       {
3293         columns-sep = \exp_not:V \__enumext_store_columns_sep_vii_dim,
3294       }
3295     }
3296   }
3297 }

```

(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`.

```

3298 \cs_new_protected:Nn \__enumext_before_list_vii:
3299 {
3300   \__enumext_vspace_above_vii:
3301   \__enumext_check_ans_exec: % need by chek-ans
3302   \__enumext_before_args_exec_vii:
3303   \__enumext_start_mini_vii:
3304 }

```

(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.

```

3305 \cs_new_protected:Nn \__enumext_after_list_vii:
3306 {
3307   \__enumext_stop_mini_vii:
3308   \__enumext_after_stop_list_vii:
3309   \__enumext_vspace_below_vii:
3310   \bool_set_false:N \l__enumext_starred_bool
3311   \__enumext_resume_save_counter:
3312 }

```

(End of definition for `__enumext_after_list_vii:`.)

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

`__enumext_stop_store_level_vii:`

```

3313 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3314 {
3315   \bool_if:NT \l__enumext_store_active_bool
3316   {
3317     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3318     {
3319       \__enumext_store_level_open_vii:
3320     }
3321   }
3322 }
3323 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3324 {
3325   \bool_if:NT \l__enumext_store_active_bool
3326   {
3327     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3328     {
3329       \__enumext_store_level_close_vii:
3330     }
3331   }
3332 }

```

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

10.35.2 The command `\item` in `enumext*`

`__enumext_start_item_tmp_vii:` First we will call the function `__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_vii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```

3333 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3334 {
3335   \__enumext_stop_item_tmp_vii:
3336   \int_incr:N \l__enumext_item_column_pos_vii_int
3337   \int_gincr:N \g__enumext_item_count_all_vii_int
3338   \__enumext_item_peek_args_vii:
3339 }

```

(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).

```

3340 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3341 {
3342   \peek_meaning:NTF (
3343     { \__enumext_joined_item_vii:w }
3344     { \__enumext_joined_item_vii:w (1) }
3345   }

```

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

`__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`.

```

3346 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3347 {
3348   \__enumext_starred_joined_item_vii:n {#1}
3349   \peek_meaning_remove:NTF *
3350   { \__enumext_starred_item_vii:w }
3351   { \__enumext_standard_item_vii:w }
3352 }

```

(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]`.

```

3353 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3354 {
3355   \bool_set_false:N \l__enumext_item_starred_vii_bool
3356   \peek_meaning:NTF [
3357   {
3358     \bool_set_eq:NN
3359     \l__enumext_wrap_label_vii_bool
3360     \l__enumext_wrap_label_opt_vii_bool
3361     \__enumext_start_item_vii:w
3362   }
3363   {
3364     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3365     \legacy_if_set_true:n { @noitemarg }
3366     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3367   }
3368 }

```

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

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

```

3369 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3370 {
3371   \bool_set_true:N \l__enumext_item_starred_vii_bool
3372   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3373   \peek_meaning:NTF [
3374   { \__enumext_starred_item_vii_aux_i:w }
3375   { \__enumext_starred_item_vii_aux_ii:w }
3376 }
3377 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3378 {
3379   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3380   \__enumext_starred_item_vii_aux_ii:w
3381 }
3382 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3383 {
3384   \peek_meaning:NTF [
3385   { \__enumext_starred_item_vii_aux_iii:w }
3386   {
3387     \dim_set_eq:NN
3388     \l__enumext_item_symbol_sep_vii_dim
3389     \l__enumext_labelsep_vii_dim
3390     \legacy_if_set_true:n { @noitemarg }
3391     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3392   }
3393 }
3394 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]

```

```

3395 {
3396   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3397   \legacy_if_set_true:n { @noitemarg }
3398   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3399 }

```

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

10.35.3 Real definition of `\item` in `enumext*`

`__enumext_start_item_vii:w` The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment.

The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to the value of `__enumext_stop_item_vii:` which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

3400 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3401 {
3402   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3403   \legacy_if:nT { @noitemarg }
3404   {
3405     \legacy_if_set_false:n { @noitemarg }
3406     \legacy_if:nT { @nmbrrlist }
3407     {
3408       \bool_if:NT \l__enumext_hyperref_bool
3409       {
3410         \legacy_if_set_true:n { @hyper@item }
3411       }
3412       \refstepcounter{enumXvii}
3413       \bool_if:NT \l__enumext_check_ans_bool
3414       {
3415         \int_gincr:N \g__enumext_count_item_number_int
3416       }
3417     }
3418   }

```

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 `\symbol` 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.

```

3419   \group_begin:
3420   \lrbox{ \l__enumext_item_text_vii_box }
3421   \bool_if:NF \l__enumext_footnotes_key_bool
3422   {
3423     \__enumext_renew_footnote:
3424   }
3425   \bool_if:NT \l__enumext_item_starred_vii_bool
3426   {
3427     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3428     {
3429       \tl_gset_eq:NN
3430       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3431     }
3432     \mode_leave_vertical:
3433     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3434     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3435     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3436     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3437   }
3438   \group_begin:
3439   \tl_use:N \l__enumext_label_font_style_vii_tl
3440   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3441   {
3442     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3443     { \__enumext_wrapper_label_vii:n {#1} }
3444   }
3445   {

```

```

3446         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3447     }
3448     \group_end:
3449     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3450     \tl_use:N \l__enumext_after_list_args_vii_tl
3451     \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3452     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3453     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3454     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3455 }

```

(End of definition for `\l__enumext_start_item_vii:w`.)

`\l__enumext_stop_item_vii:` The function `\l__enumext_stop_item_vii:` 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.

```

3456 \cs_new_protected_nopar:Nn \l__enumext_stop_item_vii:
3457 {
3458     \l__enumext_endminipage:
3459     \endlrbox
3460     \group_end:
3461     \box_set_wd:Nn \l__enumext_item_text_vii_box
3462     {
3463         \l__enumext_joined_width_vii_dim
3464         + \l__enumext_labelwidth_vii_dim
3465         + \l__enumext_labelsep_vii_dim
3466     }
3467     \int_set:Nn \hbadness { 10000 }
3468     \box_use:N \l__enumext_item_text_vii_box
3469     \bool_if:NF \l__enumext_footnotes_key_bool
3470     {
3471         \l__enumext_print_footnote:
3472     }
3473     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3474     {
3475         \par\noindent
3476         \int_zero:N \l__enumext_item_column_pos_vii_int
3477     }
3478     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3479 }

```

(End of definition for `\l__enumext_stop_item_vii:.`)

`\l__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.

```

3480 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
3481 {
3482     \int_compare:nNnT
3483     {
3484         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3485     }
3486     =
3487     { \c_zero_int }
3488     {
3489         \par
3490         \vspace{ -\l__enumext_itemsep_vii_skip }
3491         \int_gzero:N \g__enumext_item_count_all_vii_int
3492     }
3493 }

```

(End of definition for `\l__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 `\l__enumext_after_env:nn`.

```

3494 \l__enumext_after_env:nn {enumext*}
3495 {
3496     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3497     {
3498         \bool_if:NT \g__enumext_check_ans_show_h_bool
3499         {
3500             \l__enumext_check_ans_show:

```

```

3501     }
3502     \bool_gset_false:N \g__enumext_starred_bool
3503     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3504     \tl_gclear:N \g__enumext_store_name_tl
3505   }
3506 }

```

10.36 The environment keyans*

10.36.1 Functions for item box width

__enumext_starred_columns_set_viii:

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

```

3507 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3508 {
3509   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3510   {
3511     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3512     {
3513       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3514       / \l__enumext_columns_viii_int
3515     }
3516   }
3517   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3518   \dim_set:Nn \l__enumext_item_width_viii_dim
3519   {
3520     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3521     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3522     - \l__enumext_labelsep_viii_dim
3523   }
3524   \dim_zero_new:N \itemwidth
3525 }

```

(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`.

```

3526 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3527 {
3528   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3529   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3530   {
3531     \msg_warning:nnee { enumext } { item-joined }
3532     { \int_use:N \l__enumext_joined_item_viii_int }
3533     { \int_use:N \l__enumext_columns_viii_int }
3534     \int_set:Nn \l__enumext_joined_item_viii_int
3535     {
3536       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3537     }
3538   }
3539   \int_compare:nNnT
3540   { \l__enumext_joined_item_viii_int }
3541   >
3542   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3543   {
3544     \msg_warning:nnee { enumext } { item-joined-columns }
3545     { \int_use:N \l__enumext_joined_item_viii_int }
3546     {
3547       \int_eval:n
3548       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3549     }
3550     \int_set:Nn \l__enumext_joined_item_viii_int
3551     {
3552       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3553     }
3554   }
3555   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3556   {
3557     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3558     \int_decr:N \l__enumext_joined_item_aux_viii_int
3559     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3560     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }

```



```

3561 \dim_set:Nn \l__enumext_joined_width_viii_dim
3562 {
3563   \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3564   + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3565       + \l__enumext_columns_sep_viii_dim
3566       )*\l__enumext_joined_item_aux_viii_int
3567 }
3568 \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3569 }
3570 {
3571   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3572   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3573 }
3574 }

```

(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:

```

3575 \cs_new_protected:Nn \__enumext_start_mini_viii:
3576 {
3577   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3578   {
3579     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3580     {
3581       \linewidth
3582       - \l__enumext_minipage_right_viii_dim
3583       - \l__enumext_minipage_hsep_viii_dim
3584     }
3585     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3586     \dim_gset_eq:NN
3587       \g__enumext_minipage_right_viii_dim
3588       \l__enumext_minipage_right_viii_dim
3589     \__enumext_mini_addvspace_viii:
3590     \nointerlineskip\noindent
3591     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3592   }
3593 }
3594 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3595 {
3596   \bool_if:NT \l__enumext_minipage_active_viii_bool
3597   {
3598     \end{__enumext_mini_env*}
3599     \hfill
3600     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3601   }
3602 }
3603 \__enumext_after_env:nn {keyans*}
3604 {
3605   \bool_if:NT \g__enumext_minipage_active_viii_bool
3606   {
3607     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3608     \par\addvspace { \g__enumext_minipage_right_skip }
3609     \bool_if:NF \g__enumext_minipage_center_viii_bool
3610     {
3611       \centering
3612     }
3613     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3614     \end{__enumext_mini_env*}
3615     \par\addvspace{ \g__enumext_minipage_after_skip }
3616   }
3617   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3618   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3619   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3620   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3621 }

```

(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.

```

3622 \NewDocumentEnvironment{keyans*}{ o }
3623 {
3624   \__enumext_safe_exec_viii:
3625   \__enumext_parse_keys_viii:n {#1}
3626   \__enumext_before_list_viii:
3627   \__enumext_start_list:nn { }
3628   {
3629     \__enumext_list_arg_two_viii:
3630     \__enumext_before_keys_exec_viii:
3631   }
3632   \__enumext_starred_columns_set_viii:
3633   \item[] \scan_stop:
3634   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3635   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3636 }
3637 {
3638   \__enumext_stop_item_tmp_viii:
3639   \__enumext_remove_extra_parsep_viii:
3640   \__enumext_keyans_check_ans:nn { item }{ keyans* }
3641   \__enumext_stop_list:
3642   \__enumext_after_list_viii:
3643 }

```

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

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

```

3644 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3645 {
3646   \int_incr:N \__enumext_keyans_level_h_int
3647   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
3648   {
3649     \msg_error:nn { enumext } { nested }
3650   }
3651   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3652   \bool_set_false:N \__enumext_store_active_bool
3653   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3654   {
3655     \msg_error:nn { enumext } { keyans-wrong-level }
3656   }
3657 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

3658 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3659 {
3660   \tl_if_novalue:nF {#1}
3661   {
3662     \keys_set:nn { enumext / keyans* } {#1}
3663   }
3664 }

```

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

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

```

3665 \cs_new_protected:Nn \__enumext_before_list_viii:
3666 {
3667   \__enumext_vspace_above_viii:
3668   \__enumext_before_args_exec_viii:
3669   \__enumext_start_mini_viii:
3670 }

```

(End of definition for `__enumext_before_list_viii:`.)

`__enumext_after_list_viii:` The function `__enumext_after_list:` first call the function `__enumext_stop_mini_viii:`, then apply the `{<code>}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

3671 \cs_new_protected:Nn \__enumext_after_list_viii:
3672 {

```

```

3673     \__enumext_stop_mini_viii:
3674     \__enumext_after_stop_list_viii:
3675     \__enumext_vspace_below_viii:
3676 }

```

(End of definition for __enumext_after_list_viii:.)

10.36.2 The command \item in keyans*

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the optional argument (`\langle number \rangle`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\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 for `\item*`, `\item* [\langle content \rangle]`, `\item (\langle number \rangle) *` and `\item (\langle number \rangle) * [\langle content \rangle]` commands.

`__enumext_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

3677 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3678 {
3679     \__enumext_stop_item_tmp_viii:
3680     \int_incr:N \l__enumext_item_column_pos_viii_int
3681     \int_gincr:N \g__enumext_item_count_all_viii_int
3682     \__enumext_item_peek_args_viii:
3683 }

```

(End of definition for __enumext_start_item_tmp_viii:.)

`__enumext_item_peek_args_viii:` The function `__enumext_item_peek_args_viii:` will handle the `\item (\langle number \rangle)`. Look for the argument “`(`”, if it is present we will call the function `__enumext_joined_item_viii:w (\langle number \rangle)`, 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)`.

```

3684 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3685 {
3686     \peek_meaning:NTF (
3687         { \__enumext_joined_item_viii:w }
3688         { \__enumext_joined_item_viii:w (1) }
3689     }

```

(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`.

```

3690 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3691 {
3692     \__enumext_starred_joined_item_viii:n {#1}
3693     \peek_meaning_remove:NTF *
3694         { \__enumext_starred_item_viii:w }
3695         { \__enumext_standard_item_viii:w }
3696 }

```

(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 [\langle custom \rangle]` 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]`.

```

3697 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3698 {
3699     \bool_set_false:N \l__enumext_item_starred_viii_bool
3700     \peek_meaning:NTF [
3701         {

```

```

3702         \bool_set_eq:NN
3703         \l__enumext_wrap_label_viii_bool
3704         \l__enumext_wrap_label_opt_viii_bool
3705         \__enumext_start_item_viii:w
3706     }
3707     {
3708         \bool_set_true:N \l__enumext_wrap_label_viii_bool
3709         \legacy_if_set_true:n { @noitemarg }
3710         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3711     }
3712 }

```

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

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w

```

The function `__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[\langle content \rangle]`.

```

3713 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3714 {
3715     \bool_set_true:N \l__enumext_item_starred_viii_bool
3716     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3717     \peek_meaning:NTF [
3718         { \__enumext_starred_item_viii_aux_i:w }
3719         { \__enumext_starred_item_viii_aux_ii:w }
3720     }

```

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

```

3721 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3722 {
3723     \tl_clear:N \l__enumext_store_keyans_label_tl
3724     \tl_if_no_value:nF { #1 }
3725     {
3726         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3727         {
3728             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3729             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3730         }
3731         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3732     }
3733     \__enumext_starred_item_viii_aux_ii:w
3734 }
3735 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3736 {
3737     \legacy_if_set_true:n { @noitemarg }
3738     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3739 }

```

(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 `[\langle content \rangle]` for `\item*[\langle content \rangle]` 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.

```

3740 \cs_new_protected:Nn \__enumext_starred_item_exec:
3741 {
3742     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3743     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3744     \__enumext_keyans_store_ref:
3745     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3746     \__enumext_keyans_addto_seq_link:
3747     \bool_if:NT \l__enumext_show_answer_bool
3748     {
3749         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3750     }
3751     \bool_if:NT \l__enumext_show_position_bool
3752     {
3753         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3754         {
3755             \group_begin:

```

```

3756         \exp_not:N \normalfont
3757         \exp_not:N \footnotesize [ \int_eval:n
3758         {
3759             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3760         }
3761     ]
3762     \group_end:
3763 }
3764 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3765 }
3766 }

```

(End of definition for `__enumext_starred_item_exec:`.)

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

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

```

3767 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3768 {
3769     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3770     \legacy_if:nT { @noitemarg }
3771     {
3772         \legacy_if_set_false:n { @noitemarg }
3773         \legacy_if:nT { @nmbrrlist }
3774         {
3775             \bool_if:NT \l__enumext_hyperref_bool
3776             {
3777                 \legacy_if_set_true:n { @hyper@item }
3778             }
3779             \refstepcounter{enumXviii}
3780         }
3781     }

```

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

```

3782     \group_begin:
3783     \lrbox{ \l__enumext_item_text_viii_box }
3784     \bool_if:NF \l__enumext_footnotes_key_bool
3785     {
3786         \__enumext_renew_footnote:
3787     }
3788     \bool_if:NT \l__enumext_item_starred_viii_bool
3789     {
3790         \__enumext_starred_item_exec:
3791     }
3792     \group_begin:
3793     \tl_use:N \l__enumext_label_font_style_viii_tl
3794     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3795     {
3796         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3797         { \__enumext_wrapper_label_viii:n {#1} }
3798     }
3799     {
3800         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3801     }
3802     \group_end:
3803     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3804     \tl_use:N \l__enumext_after_list_args_viii_tl
3805     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3806     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3807     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3808     \bool_if:NT \l__enumext_item_starred_viii_bool
3809     {
3810         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3811         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3812     }
3813     {
3814         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3815     }
3816 }

```

(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.

```

3817 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3818 {
3819     \__enumext_endminipage:
3820     \endlrbox
3821     \group_end:
3822     \box_set_wd:Nn \l__enumext_item_text_viii_box
3823     {
3824         \l__enumext_joined_width_viii_dim
3825         + \l__enumext_labelwidth_viii_dim
3826         + \l__enumext_labelsep_viii_dim
3827     }
3828     \int_set:Nn \hbadness { 10000 }
3829     \box_use:N \l__enumext_item_text_viii_box
3830     \bool_if:NF \l__enumext_footnotes_key_bool
3831     {
3832         \__enumext_print_footnote:
3833     }
3834     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3835     {
3836         \par\noindent
3837         \int_zero:N \l__enumext_item_column_pos_viii_int
3838     }
3839     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3840 }

```

(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.

```

3841 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3842 {
3843     \int_compare:nNnT
3844     {
3845         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3846     }
3847     =
3848     { \c_zero_int }
3849     {
3850         \par
3851         \vspace{ -\l__enumext_itemsep_viii_skip }
3852         \int_gzero:N \g__enumext_item_count_all_viii_int
3853     }
3854 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`.)

10.37 The command `\getkeyans`

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form `{⟨store name : position⟩}`. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from *⟨prop list⟩* defined by `save-ans` key.

```

3855 \NewDocumentCommand \getkeyans { m }
3856 {
3857     \exp_args:Ne \__enumext_getkeyans_aux:n
3858     { \tl_to_str:e { \text_expand:n {#1} } }
3859 }

```

(End of definition for `\getkeyans`. This function is documented on page 13.)

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

```

3860 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3861 {
3862     \str_if_in:nnTF {#1} { : }
3863     {
3864         \use:e
3865         {
3866             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3867             { {##1} {##2} }

```

```

3868     }
3869     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3870   }
3871   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3872 }

```

(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⟩*.

```

3873 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3874 {
3875   \prop_if_exist:cF { g__enumext_#1_prop }
3876   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3877   \group_begin:
3878     \prop_item:cn { g__enumext_#1_prop }{#2}
3879   \group_end:
3880 }

```

(End of definition for __enumext_getkeyans:nn.)

10.38 The command \printkeyans

The \printkeyans command prints “all stored content” in the *⟨sequence⟩* defined by the save-ans key. The first thing we will do is to define a set of *⟨keys⟩* with which we will control the options of the different nesting levels for the enumext and enumext* environment by storing the values of these in the token list variables \l__enumext_print_keyans_X_tl.

```

3881 \keys_define:nn { keyanskey / print }
3882 {
3883   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3884                   {
3885                     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3886                   },
3887   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3888   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3889                   {
3890                     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3891                   },
3892   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3893   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3894                   {
3895                     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3896                   },
3897   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3898   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3899                   {
3900                     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3901                   },
3902   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3903   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3904                   {
3905                     \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3906                   },
3907   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3908 }

```

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

```

3909 \NewDocumentCommand \printkeyans { s O{} m }
3910 {
3911   \group_begin:
3912     \tl_use:N \l__enumext_print_keyans_i_tl
3913     \tl_use:N \l__enumext_print_keyans_ii_tl
3914     \tl_use:N \l__enumext_print_keyans_iii_tl
3915     \tl_use:N \l__enumext_print_keyans_iv_tl
3916     \tl_use:N \l__enumext_print_keyans_vii_tl
3917     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3918   \group_end:
3919 }

```

(End of definition for \printkeyans. This function is documented on page 13.)

`__enumext_printkeyans:nnn`

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

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

```

3920 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3921 {
3922   \seq_if_exist:cTF { g__enumext_#3_seq }
3923   {
3924     \seq_if_empty:cF { g__enumext_#3_seq }
3925     {
3926       %%\seq_show:c { g__enumext_#3_seq }
3927       \bool_if:nTF {#1}
3928       {
3929         \begin{enumext*}[#2]
3930         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3931         \end{enumext*}
3932       }
3933       {
3934         \begin{enumext}[#2]
3935         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3936         \end{enumext}
3937       }
3938     }
3939   }
3940   {
3941     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3942   }
3943 }

```

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

10.39 The command `\setenumext`

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

```

3944 \keys_define:nn { enumext / meta-families }
3945 {
3946   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3947   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3948   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3949   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3950   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3951   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3952   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3953   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3954   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3955   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3956   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3957   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3958   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3959 }

```

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

```

3960 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3961 {
3962   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3963   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3964 }

```

`\setenumext`

Now we define the user command `\setenumext`.

```

3965 \NewDocumentCommand \setenumext { o +m }
3966 {
3967   \tl_if_novalue:nTF {#1}
3968   {
3969     \seq_map_inline:Nn \c__enumext_all_families_seq
3970     }
3971   {
3972     \seq_clear:N \l__enumext_setkey_tmpa_seq
3973     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3974     \int_set:Nn \l__enumext_setkey_tmpa_int

```



```

3975         {
3976             \seq_count:N \l__enumext_setkey_tmpb_seq
3977         }
3978     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3979     {
3980         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3981         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3982         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3983         {
3984             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3985         }
3986     }
3987     {
3988         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3989     }
3990     \seq_if_empty:NNTF \l__enumext_setkey_tmpa_seq
3991     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3992     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3993 }
3994 {
3995     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3996 }
3997 }

```

(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.

```

3998 \cs_new_protected:Npn \__enumext_set_parse:n #1
3999 {
4000     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4001     \int_step_inline:nnn { 0 } { 4 } %<- max level
4002     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4003     \tl_if_empty:NNTF \l__enumext_setkey_tmpb_tl
4004     {
4005         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4006         { \tl_trim_spaces:n {#1} }
4007     }
4008     { \__enumext_set_error:nn {#1} { } }
4009 }
4010 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4011 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

10.40 Messages

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

```

4012 \msg_new:nnn { enumext } { package-load }
4013 {
4014     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4015 }

4016 \msg_new:nnn { enumext } { package-not-load }
4017 {
4018     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4019 }

4020 \msg_new:nnn { enumext } { package-load-foot }
4021 {
4022     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4023 }

```

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

```

4024 \msg_new:nnn { enumext } { counters }
4025 {
4026     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
4027     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4028 }

```

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

```

4029 \msg_new:nnn { enumext } { invalid-key }
4030 {
4031     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.

```

```

4032     }
4033     \msg_new:nnn { enumext } { unknown-key-family }
4034     {
4035         Unknown~key~family~`\l_keys_key_str'~for~enumext.
4036     }

```

Messages used in length calculation.

```

4037     \msg_new:nnn { enumext } { width-negative }
4038     {
4039         Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4040         The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4041     }
4042     \msg_new:nnn { enumext } { width-zero }
4043     {
4044         Invalid ~ '#1=#2' ~ \msg_line_context:.\
4045         The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4046     }

```

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

```

4047     \msg_new:nnn { enumext } { list-lengths }
4048     {
4049         **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4050         \__enumext_show_length:nnn { dim } { labelsep } {#1}
4051         \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4052         \__enumext_show_length:nnn { dim } { itemindent } {#1}
4053         \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4054         \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4055         \__enumext_show_length:nnn { dim } { listparindent } {#1}
4056         \__enumext_show_length:nnn { skip } { topsep } {#1}
4057         \__enumext_show_length:nnn { skip } { parsep } {#1}
4058         \__enumext_show_length:nnn { skip } { partopsep } {#1}
4059         \__enumext_show_length:nnn { skip } { itemsep } {#1}
4060         ****~
4061     }

```

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

```

4062     \msg_new:nnn { enumext } { list-lengths-not-nested }
4063     {
4064         **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4065         \__enumext_show_length:nnn { dim } { labelsep } {#1}
4066         \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4067         \__enumext_show_length:nnn { dim } { itemindent } {#1}
4068         \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4069         \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4070         \__enumext_show_length:nnn { dim } { listparindent } {#1}
4071         \__enumext_show_length:nnn { skip } { topsep } {#1}
4072         \__enumext_show_length:nnn { skip } { parsep } {#1}
4073         \__enumext_show_length:nnn { skip } { partopsep } {#1}
4074         \__enumext_show_length:nnn { skip } { itemsep } {#1}
4075         ****~
4076     }

```

Messages used by `ref` key.

```

4077     \msg_new:nnn { enumext } { key-ref-empty }
4078     {
4079         Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4080     }

```

Messages used by `save-ans` key.

```

4081     \msg_new:nnn { enumext } { save-ans-empty }
4082     {
4083         Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4084     }
4085
4086     \msg_new:nnn { enumext } { save-ans-ok }
4087     {
4088         Set ~ 'save-ans=#2' ~ in ~ '#1' ~ \msg_line_context:.
4089     }

```

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

```

4090     \msg_new:nnn { enumext } { items-same-answer }
4091     {
4092         *****~Checking~answers~on~'#1'~OK~*****\
4093         **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \

```

```

4094      *****
4095      \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4096    }
4097    \msg_new:nnn { enumext } { item-different-answer }
4098    {
4099      Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4100      answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4101    }

```

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

```

4102    \msg_new:nnn { enumext } { missing-starred }
4103    {
4104      Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4105    }

```

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

```

4106    \msg_new:nnn { enumext } { list-too-deep }
4107    {
4108      Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4109      The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4110    }

```

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

```

4111    \msg_new:nnn { enumext } { anskey-wrong-place }
4112    {
4113      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4114      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4115    }
4116    \msg_new:nnn { enumext } { anspic-wrong-place }
4117    {
4118      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4119      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4120    }
4121    \msg_new:nnn { enumext } { command-wrong-place }
4122    {
4123      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4124      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4125    }

```

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

```

4126    \msg_new:nnn { enumext } { keyans-nested }
4127    {
4128      The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4129    }
4130    \msg_new:nnn { enumext } { keyans-wrong-level }
4131    {
4132      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4133      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4134    }
4135    \msg_new:nnn { enumext } { wrong-place }
4136    {
4137      Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4138      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4139    }
4140    \msg_new:nnn { enumext } { keyanspic-nested }
4141    {
4142      The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4143    }
4144    \msg_new:nnn { enumext } { keyanspic-wrong-level }
4145    {
4146      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4147      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4148    }

```

Messages used by `\getkeyans` command.

```

4149    \msg_new:nnn { enumext } { undefined-storage-anskey }
4150    {
4151      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4152    }

```

Messages used by `\miniright` command.

```

4153    \msg_new:nnn { enumext } { missing-miniright }
4154    {
4155      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~\\

```

```

4156     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4157   }
4158   \msg_new:nnn { enumext } { wrong-miniright-place }
4159   {
4160     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4161     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4162   }
4163   \msg_new:nnn { enumext } { wrong-miniright-use }
4164   {
4165     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4166     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4167   }

```

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

```

4168   \msg_new:nnn { enumext } { nested }
4169   {
4170     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4171   }
4172   \msg_new:nnn { enumext } { item-joined }
4173   {
4174     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4175   }
4176   \msg_new:nnn { enumext } { item-joined-columns }
4177   {
4178     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
4179   }

```

10.41 Finish package

Finish package implementation.

```

4180   \file_input_stop:
4181   </package>

```

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

109 / 120

enumXiv 23, 30
 enumXi 23, 30
 enumXviii 23, 30
 enumXvii 23, 30, 94
 enumXvi 23, 30
 enumXv 23, 30

cs commands:

\cs_generate_variant:Nn 369, 385, 625, 641, 1872,
 1881, 1886, 1966, 2586, 3089
 \cs_if_exist:NTF 339
 \cs_new:Nn 192
 \cs_new:Npn 210, 1429, 1438, 1447
 \cs_new_eq:NN 275, 276, 277, 281, 282, 314, 315, 318,
 319
 \cs_new_protected:Nn . 202, 224, 252, 286, 490, 553,
 605, 806, 810, 814, 818, 822, 826, 830, 834, 838, 842,
 846, 850, 854, 858, 862, 866, 902, 914, 938, 955, 966,
 990, 1065, 1089, 1106, 1168, 1185, 1207, 1242, 1248,
 1323, 1337, 1351, 1362, 1373, 1384, 1395, 1406, 1491,
 1594, 1607, 1624, 1645, 1702, 1737, 1773, 1780, 1887,
 1911, 1918, 1946, 1953, 2070, 2202, 2217, 2245, 2275,
 2320, 2332, 2339, 2391, 2395, 2414, 2465, 2499, 2515,
 2525, 2541, 2679, 2737, 2766, 2773, 2796, 2826, 2838,
 2895, 2918, 2936, 2961, 2972, 3009, 3052, 3065, 3085,
 3090, 3106, 3174, 3193, 3245, 3298, 3305, 3313, 3323,
 3340, 3480, 3507, 3575, 3594, 3644, 3665, 3671, 3684,
 3740, 3841
 \cs_new_protected:Npn 184, 188, 322, 337, 354, 364,
 370, 470, 515, 587, 612, 626, 1270, 1289, 1459, 1478,
 1548, 1581, 1673, 1873, 1882, 2002, 2147, 2159, 2181,
 2255, 2297, 2305, 2424, 2442, 2476, 2488, 2555, 2589,
 2629, 2688, 2708, 2914, 3060, 3125, 3256, 3269, 3346,
 3353, 3369, 3377, 3382, 3394, 3526, 3658, 3690, 3697,
 3713, 3721, 3735, 3860, 3873, 3920, 3998, 4010
 \cs_new_protected_nopar:Nn . . . 3333, 3456, 3677,
 3817
 \cs_new_protected_nopar:Npn 3400, 3767
 \cs_set:Nn 2152
 \cs_set:Npn 2080, 2118, 3866
 \cs_set_eq:NN . . 3235, 3236, 3402, 3634, 3635, 3769
 \cs_set_protected:Nn 216, 730, 746, 758, 770
 \cs_set_protected:Npn . 32, 41, 59, 67, 79, 85, 113,
 151, 159, 218, 386, 408, 437, 450, 497, 642, 662, 706,
 725, 782, 791, 870, 887, 1306, 1417, 1664, 1719, 1793,
 1829, 1847, 2072, 2206, 2375, 2587, 2627
 \cs_to_str:N 356, 379

D

\d 198
 \DeclareDocumentEnvironment 983
 dim commands:

\dim_abs:n 2560, 2565
 \dim_add:Nn 3070
 \dim_compare:nNnTF . 732, 748, 760, 772, 1272, 1291,
 2557, 2562, 2568, 2574, 2576, 2578, 2778, 2801, 2922,
 2940, 3062, 3108, 3176, 3509, 3577
 \dim_compare:nTF 2043
 \dim_gset_eq:NN 3185, 3586
 \dim_gzero:N 3219, 3620
 \dim_new:N . 55, 62, 63, 64, 81, 107, 120, 130, 168, 169,
 175
 \dim_set:Nn . . 367, 720, 1859, 2456, 2560, 2565, 2567,
 2570, 2571, 2575, 2577, 2580, 2581, 2583, 2781, 2804,
 2924, 2942, 3092, 3110, 3117, 3160, 3178, 3396, 3511,
 3518, 3561, 3579

\dim_set_eq:NN 457, 504, 575, 579, 2451, 2598, 2638,
 2727, 2812, 2950, 3167, 3170, 3171, 3288, 3387, 3568,
 3571, 3572
 \dim_use:N 733, 741, 1273, 1279, 1956, 1959, 1964, 2520,
 2522, 2779, 2784, 2785, 2792, 2802, 2806, 2807, 2809
 \dim_zero:N 2816, 2952, 3071, 3072, 3073
 \dim_zero_new:N 3123, 3524
 \c_zero_dim 735, 749, 761, 773, 1273, 1291, 2045, 2557,
 2562, 2568, 2575, 2779, 2802, 2922, 2940, 3108, 3176,
 3509, 3577

E

\end . . 1276, 1294, 1915, 1950, 2831, 2848, 2965, 2982, 3197,
 3213, 3598, 3614, 3931, 3936
 \endlist 29
 \endlist 276
 \endlrbox 3459, 3820
 \endminipage 29
 \endminipage 282
 enumext 5, 2660

enumext internal commands:

\l_enumext_ref_the_count_tl 33
 \g_enumext_ t__enumext_store_name_tl
 _prop 70
 \l_enumext__resume_name_tl 54
 __enumext_add_pre_parsep: . . . 42, 912, 914, 914
 __enumext_after_args_exec: . 40, 806, 818, 2672
 __enumext_after_args_exec_v: . 40, 41, 822, 834,
 2888
 __enumext_after_args_exec_vii: . . . 838, 862
 __enumext_after_args_exec_viii: 866
 __enumext_after_env:nn . . 82, 95, 188, 188, 2864,
 3202, 3494, 3603
 __enumext_after_hyperref: . . . 29, 284, 286, 286
 __enumext_after_list: 81, 92, 98, 2677, 2838, 2838
 \l_enumext_after_list_args_v_tl 836
 \l_enumext_after_list_args_vii_tl 864, 3450
 \l_enumext_after_list_args_viii_tl 868, 3804
 __enumext_after_list_v: . . 84, 2893, 2972, 2972
 __enumext_after_list_vii: . . . 3243, 3305, 3305
 __enumext_after_list_viii: . . 3642, 3671, 3671
 __enumext_after_star_env:nn 89
 __enumext_after_stop_list: . . . 39, 41, 806, 814,
 2859
 __enumext_after_stop_list_v: 40, 822, 830, 2987
 \l_enumext_after_stop_list_v_tl 832
 __enumext_after_stop_list_vii: 838, 854, 3308
 \l_enumext_after_stop_list_vii_tl . . . 856
 __enumext_after_stop_list_viii: . 858, 3674
 \l_enumext_after_stop_list_viii_tl . . . 860
 \l_enumext_align_label_vii_str . . 3442, 3446
 \l_enumext_align_label_viii_str . 3796, 3800
 \l_enumext_align_label_X_str 159
 \c_enumext_all_envs_clist . . 179, 407, 661, 724,
 790, 805, 886, 1322
 \c_enumext_all_families_seq . 104, 3960, 3969,
 3991
 __enumext_anskey_wrapper:n 1797, 2157
 __enumext_at_begin_document:n 28, 29, 184, 184,
 273, 279
 __enumext_before_args_exec: 39, 806, 806, 2776
 __enumext_before_args_exec_v: . . 40, 822, 822,
 2921
 __enumext_before_args_exec_vii: . . 838, 838,
 3302

```

\__enumext_before_args_exec_viii: 842, 3668
\__enumext_before_keys_exec: 39, 806, 810, 2670
\__enumext_before_keys_exec_v: .. 40, 822, 826, 2886
\__enumext_before_keys_exec_vii ..... 838
\__enumext_before_keys_exec_vii: 40, 846, 3231
\__enumext_before_keys_exec_viii: .. 40, 850, 3630
\__enumext_before_list: ... 80, 2664, 2773, 2773
\__enumext_before_list_v: . 83, 2881, 2918, 2918
\__enumext_before_list_vii: 91, 3226, 3298, 3298
\__enumext_before_list_viii: .. 98, 3626, 3665, 3665
\l__enumext_before_no_starred_key_v_tl 828
\l__enumext_before_no_starred_key_vii_-tl ..... 848
\l__enumext_before_no_starred_key_viii_-tl ..... 852
\l__enumext_before_starred_key_v_tl ... 824
\l__enumext_before_starred_key_vii_tl . 840
\l__enumext_before_starred_key_viii_tl 844
\__enumext_calc_hspace:NNNNNNN 76, 2555, 2555, 2586, 2591, 2631
\l__enumext_check_ans_bool ... 72, 73, 134, 1723, 1727, 1775, 1994, 2292, 2428, 2458, 2853, 3413
\__enumext_check_ans_exec: .. 59, 80, 1773, 1773, 2777, 3301
\g__enumext_check_ans_item_tl .. 69, 134, 2291, 2299, 2303
\__enumext_check_ans_set: . 59, 1737, 1737, 1777
\__enumext_check_ans_show: 60, 1780, 1780, 2870, 3500
\g__enumext_check_ans_show_bool 82, 134, 2856, 2868, 2873
\g__enumext_check_ans_show_h_bool 134, 3498, 3503
\__enumext_check_first_level: . 28, 78, 224, 252, 2686, 3254
\l__enumext_columns_sep_v_dim 2940, 2942, 2950
\l__enumext_columns_sep_vii_dim .. 3108, 3110, 3119, 3164, 3290, 3478
\l__enumext_columns_sep_viii_dim . 3509, 3511, 3520, 3565, 3839
\l__enumext_columns_v_int 1111, 2938, 2946, 2958, 2963
\l__enumext_columns_vii_int .. 3113, 3116, 3120, 3128, 3132, 3135, 3141, 3147, 3151, 3277, 3473, 3484
\l__enumext_columns_viii_int . 3514, 3517, 3521, 3529, 3533, 3536, 3542, 3548, 3552, 3834, 3845
\g__enumext_count_item_anskey_int .. 69, 134, 1783, 1791, 1996, 2294
\g__enumext_count_item_number_int 134, 1748, 1753, 1756, 1759, 1767, 1783, 1790, 2430, 2460, 3415
\g__enumext_count_item_with_ans_int .... 64
\l__enumext_counter_i_tl ..... 32, 346
\l__enumext_counter_ii_tl ..... 32, 347
\l__enumext_counter_iii_tl ..... 32, 348
\l__enumext_counter_iv_tl ..... 32, 349
\c__enumext_counter_style_tl ..... 27, 37, 204
\g__enumext_counter_styles_tl . 23, 31, 55, 357, 375
\l__enumext_counter_v_tl ..... 32, 350, 595
\l__enumext_counter_vi_tl ..... 32, 351
\l__enumext_counter_vii_tl ..... 32, 352, 525
\l__enumext_counter_viii_tl ..... 32, 353, 542
\__enumext_current_env_set_bool: . 28, 78, 224, 224, 2681, 3247
\l__enumext_current_widest_dim 23, 55, 381, 458, 505, 576, 580
\__enumext_default_item:n ... 2424, 2424, 2473
\__enumext_define_counters:Nn 23, 337, 337, 346, 347, 348, 349, 350, 351, 352, 353
\__enumext_endminipage: . 29, 279, 282, 989, 3102, 3458, 3819
\__enumext_fake_item: ..... 730, 730, 2618
\l__enumext_fake_item_indent_v_dim 749, 754
\l__enumext_fake_item_indent_v_tl 751, 2481, 2485, 2493
\l__enumext_fake_item_indent_vii_dim 761, 766
\l__enumext_fake_item_indent_vii_tl 763, 3454
\l__enumext_fake_item_indent_viii_dim . 773, 778, 3811
\l__enumext_fake_item_indent_viii_tl .. 775, 3810, 3814
\l__enumext_fake_item_indent_X_tl ..... 85
\__enumext_fake_item_vii: .... 730, 758, 2648
\__enumext_fake_item_viii: .... 730, 770, 2653
\__enumext_filter_series:n 52, 1429, 1429, 1471, 1483, 1488
\__enumext_filter_series_key:n 53, 1429, 1434, 1438
\__enumext_filter_series_pair:nn .. 53, 1429, 1435, 1447
\g__enumext_footnote_arg_seq . 156, 2397, 2410, 2420
\g__enumext_footnote_int . 156, 2404, 2407, 2409, 2411
\g__enumext_footnote_int_seq . 156, 2398, 2411, 2416, 2419
\__enumext_footnotes_key_bool ..... 29
\l__enumext_footnotes_key_bool 25, 29, 94, 146, 295, 300, 309, 3421, 3469, 3784, 3830
\__enumext_footnotetext:nn ... 2391, 2391, 2421
\__enumext_getkeyans:nn .. 103, 3869, 3873, 3873
\__enumext_getkeyans_aux:n 102, 3857, 3860, 3860
\l__enumext_hyperref_bool . 25, 29, 30, 146, 291, 312, 329, 2060, 2279, 3408, 3775
\__enumext_hypertarget:nn 30, 286, 314, 318, 334
\__enumext_if_is_int:n ..... 196
\__enumext_if_is_int:nTF ..... 196, 614, 628
\l__enumext_item_column_pos_vii_int 92, 3135, 3141, 3147, 3151, 3158, 3336, 3473, 3476
\l__enumext_item_column_pos_viii_int ... 99, 3536, 3542, 3548, 3552, 3559, 3680, 3834, 3837
\l__enumext_item_column_pos_X_int ..... 159
\g__enumext_item_count_all_vii_int 92, 3159, 3337, 3484, 3491
\g__enumext_item_count_all_viii_int 99, 3560, 3681, 3845, 3852
\g__enumext_item_count_all_X_int ..... 159
\__enumext_item_peek_args_vii: 92, 3338, 3340, 3340
\__enumext_item_peek_args_viii: 99, 3682, 3684, 3684
\__enumext_item_starred: .. 74, 2515, 2515, 2533
\l__enumext_item_starred_vii_bool 3355, 3371, 3425
\l__enumext_item_starred_viii_bool 3699, 3715,

```


3788, 3808
 \l__enumext_item_starred_X_bool 159
 __enumext_item_std:w 29, 72–74, 87, 273, 277, 2433,
 2439, 2463, 2481, 2485, 2493, 3083
 \g__enumext_item_symbol_aux_vii_tl 3379, 3427,
 3430, 3434, 3436
 \g__enumext_item_symbol_aux_X_tl 159
 \l__enumext_item_symbol_sep_vii_dim . . 3388,
 3396, 3433, 3435
 \g__enumext_item_symbol_tl 23, 73, 48, 2448, 2521,
 2538
 \l__enumext_item_symbol_vii_tl 3430
 \l__enumext_item_text_vii_box 3420, 3461, 3468
 \l__enumext_item_text_viii_box 3783, 3822, 3829
 \l__enumext_item_text_X_box 159
 \l__enumext_item_width_vii_dim . . 3117, 3162,
 3170, 3171
 \l__enumext_item_width_viii_dim . . 3518, 3563,
 3571, 3572
 \l__enumext_item_width_X_dim 159
 \l__enumext_itemindent_X_dim 59
 \l__enumext_itemsep_vii_skip 3490
 \l__enumext_itemsep_viii_skip 3851
 \l__enumext_joined_item_aux_vii_int . . 3156,
 3157, 3158, 3159, 3165
 \l__enumext_joined_item_aux_viii_int . 3557,
 3558, 3559, 3560, 3566
 \l__enumext_joined_item_aux_X_int . . . 159
 __enumext_joined_item_vii:w 92, 93, 3343, 3344,
 3346, 3346
 \l__enumext_joined_item_vii_int . . 3127, 3128,
 3131, 3133, 3139, 3144, 3149, 3154, 3156, 3162
 __enumext_joined_item_viii:w . 99, 3687, 3688,
 3690, 3690
 \l__enumext_joined_item_viii_int . 3528, 3529,
 3532, 3534, 3540, 3545, 3550, 3555, 3557, 3563
 \l__enumext_joined_item_X_int 159
 \l__enumext_joined_width_vii_dim . 3160, 3167,
 3170, 3451, 3463
 \l__enumext_joined_width_viii_dim 3561, 3568,
 3571, 3805, 3824
 \l__enumext_joined_width_X_dim 159
 __enumext_keyans_addto_prop:n 67, 2181, 2181,
 2495, 3015
 __enumext_keyans_addto_seq:n . 69, 2255, 2255,
 2497, 3017
 __enumext_keyans_addto_seq_link: 2255, 2273,
 2275, 3746
 __enumext_keyans_anspic_code:nnn 85, 86, 3006,
 3009, 3009
 __enumext_keyans_check_ans:nn . . 69, 70, 2297,
 2297, 2891, 3047, 3640
 __enumext_keyans_default_item:n . . 73, 2476,
 2476, 2511
 \l__enumext_keyans_env_bool 20, 2742, 2755, 2902,
 2986
 __enumext_keyans_fake_item: . . 730, 746, 2608
 \l__enumext_keyans_item_opt_tl 97, 2309, 2322,
 2328, 3731
 \l__enumext_keyans_level_h_int . . 20, 535, 562,
 2233, 3646, 3647
 \l__enumext_keyans_level_int . . 20, 1264, 1983,
 2228, 2901, 2905, 3000
 __enumext_keyans_make_label: 32, 75, 2541, 2541,
 2606
 __enumext_keyans_mini_addvspace: 47, 83, 1168,
 1168, 2930
 __enumext_keyans_mini_right_cmd:n 50, 1266,
 1289, 1289
 __enumext_keyans_mini_set_vskip: . 46, 1106,
 1106, 1170
 __enumext_keyans_multi_addvspace: . 84, 955,
 966, 2955
 __enumext_keyans_multi_set_vskip: . 43, 955,
 955, 968
 __enumext_keyans_multicols_start: . . 83, 84,
 2934, 2936, 2936
 __enumext_keyans_multicols_stop: . 84, 1293,
 2961, 2961, 2985
 __enumext_keyans_parse_keys:n 2880, 2914, 2914
 \l__enumext_keyans_pic_above_int . 129, 3093,
 3094, 3096
 \l__enumext_keyans_pic_above_skip . . 87, 129,
 3038, 3077
 __enumext_keyans_pic_arg_two: 87, 3036, 3065,
 3065
 \l__enumext_keyans_pic_below_int . 129, 3093,
 3094, 3097
 \l__enumext_keyans_pic_body_seq . . 85–87, 129,
 3004, 3043, 3101
 __enumext_keyans_pic_do:n 87, 3043, 3045, 3085,
 3085, 3089
 \l__enumext_keyans_pic_level_int . . 20, 1256,
 1987, 2184, 2223, 2258, 2341, 3054, 3055
 __enumext_keyans_pic_row:n 87, 3087, 3090, 3090
 __enumext_keyans_pic_safe_exec: . . 86, 3032,
 3052, 3052
 __enumext_keyans_pic_skip_abs:N . . 87, 3060,
 3060, 3076
 \l__enumext_keyans_pic_width_dim . 129, 3092,
 3099
 __enumext_keyans_redefine_item: . . 74, 2499,
 2499, 2605
 __enumext_keyans_ref: 35, 587, 605, 2607
 __enumext_keyans_ref:n 35, 584, 587, 587
 __enumext_keyans_safe_exec: . 2879, 2895, 2895
 __enumext_keyans_show_ans: . . 2305, 2313, 2332
 __enumext_keyans_show_item_opt: . 2305, 2320,
 2493, 3028, 3811
 __enumext_keyans_show_left:n . 74, 2305, 2305,
 2491, 3023
 __enumext_keyans_show_pos: . . 2305, 2317, 2339
 __enumext_keyans_starred_item:n . . 74, 2488,
 2488, 2507
 __enumext_keyans_store_ref: . . 68, 2202, 2202,
 2496, 3016, 3744
 __enumext_keyans_store_ref_aux_i: 68, 2202,
 2214, 2217
 __enumext_keyans_store_ref_aux_ii: 69, 2202,
 2243, 2245
 \l__enumext_keyans_tmpa_dim 97
 \l__enumext_keyans_tmpa_tl 24, 100, 97, 2490, 2494
 \l__enumext_keyans_tmpb_tl 100, 97
 __enumext_keyans_wrapper_opt:n . . 1800, 2328
 \l__enumext_label_copy_i_tl . . 2114, 2221, 2226,
 2231, 2236
 \l__enumext_label_copy_v_tl 2231
 \l__enumext_label_copy_vi_tl 2226

`\l__enumext_label_copy_vii_tl` 2089, 2100, 2131, 2221
`\l__enumext_label_copy_viii_tl` 2236
`\l__enumext_label_copy_X_tl` 148
`\l__enumext_label_fill_left_v_tl` 2545
`\l__enumext_label_fill_left_X_tl` 85
`\l__enumext_label_fill_right_v_tl` 2552
`\l__enumext_label_fill_right_X_tl` 85
`\l__enumext_label_font_style_v_tl` 2546, 3027
`\l__enumext_label_font_style_vii_tl` . . . 3439
`\l__enumext_label_font_style_viii_tl` . . 3793
`\l__enumext_label_i_tl` 450
`\l__enumext_label_ii_tl` 450
`\l__enumext_label_iii_tl` 450
`\l__enumext_label_iv_tl` 450
`__enumext_label_style:Nnn` 23, 31, 370, 370, 385, 455, 502, 573, 577
`\l__enumext_label_v_tl` . . 67, 69, 570, 2189, 2263, 2334, 2368, 2490, 2494, 2883, 3022, 3024
`\l__enumext_label_vi_tl` . 67, 69, 570, 2186, 2260, 3022, 3024, 3028
`\l__enumext_label_vii_tl` . 497, 3366, 3391, 3398
`\l__enumext_label_viii_tl` 497, 3710, 3738, 3742
`\l__enumext_label_width_by_box` . . 55, 366, 367
`__enumext_label_width_by_box:Nn` 31, 364, 364, 369, 381, 638
`\l__enumext_labelsep_i_dim` . . 2336, 2372, 3749, 3764
`\l__enumext_labelsep_v_dim` 2945
`\l__enumext_labelsep_vii_dim` . 3112, 3121, 3163, 3389, 3449, 3465
`\l__enumext_labelsep_viii_dim` 3513, 3522, 3564, 3803, 3826
`\l__enumext_labelwidth_i_dim` . 2336, 2371, 3749, 3764
`\l__enumext_labelwidth_v_dim` 2945
`\l__enumext_labelwidth_vii_dim` . . 3112, 3120, 3163, 3442, 3446, 3464
`\l__enumext_labelwidth_viii_dim` . . 3513, 3521, 3564, 3796, 3800, 3825
`\l__enumext_leftmargin_tmp_v_bool` . 87, 3067
`\l__enumext_leftmargin_tmp_X_bool` 59
`\l__enumext_leftmargin_tmp_X_dim` 59
`\l__enumext_leftmargin_X_dim` 59
`__enumext_level:` 192, 192, 479, 482, 483, 492, 494, 733, 737, 741, 808, 812, 816, 820, 904, 906, 908, 910, 943, 945, 947, 949, 953, 993, 996, 1015, 1024, 1030, 1035, 1039, 1050, 1054, 1055, 1060, 1096, 1100, 1273, 1279, 1326, 1328, 1330, 1333, 1340, 1342, 1344, 1347, 1891, 1899, 1903, 1907, 2152, 2155, 2156, 2432, 2433, 2437, 2438, 2439, 2446, 2448, 2452, 2453, 2456, 2462, 2463, 2517, 2520, 2522, 2529, 2530, 2531, 2534, 2537, 2667, 2669, 2710, 2715, 2716, 2717, 2719, 2723, 2728, 2729, 2730, 2732, 2748, 2761, 2768, 2779, 2781, 2784, 2785, 2787, 2792, 2799, 2802, 2804, 2806, 2807, 2808, 2809, 2812, 2818, 2823, 2829, 2832, 2834, 2840
`\l__enumext_level_h_int` . . 20, 232, 258, 266, 518, 555, 1745, 1762, 2108, 2125, 3248, 3249
`\l__enumext_level_int` . 78, 20, 194, 243, 257, 267, 916, 1067, 1260, 1739, 2085, 2095, 2101, 2107, 2115, 2123, 2130, 2621, 2682, 2683, 2693, 2700, 2746, 2759, 2814, 2866, 2909, 2996, 3317, 3327, 3496, 3653
`__enumext_list_arg_two_i:` 2587
`__enumext_list_arg_two_ii:` 2587
`__enumext_list_arg_two_iii:` 2587
`__enumext_list_arg_two_iv:` 2587
`__enumext_list_arg_two_v:` . 74, 2587, 2885, 3068
`__enumext_list_arg_two_vii:` 2627, 3230
`__enumext_list_arg_two_viii:` 2627, 3629
`\l__enumext_listoffset_v_dim` 2947
`\l__enumext_listparindent_vii_dim` 3452
`\l__enumext_listparindent_viii_dim` . . . 3806
`__enumext_make_label:` 32, 72, 73, 75, 2525, 2525, 2616
`\l__enumext_mark_answer_sym_tl` . 63, 124, 1806, 1961, 2167, 2343, 2356, 3753
`\l__enumext_mark_position_str` 124, 1810, 1811, 1834, 1835, 1959
`\l__enumext_mark_ref_sym_tl` . . 124, 1820, 2065, 2287
`__enumext_mini_addvspace:` . . 46, 80, 1089, 1089, 2789
`__enumext_mini_addvspace_vii:` 48, 1242, 1242, 3188
`__enumext_mini_addvspace_viii:` 48, 1242, 1248, 3589
`__enumext_mini_env*` 983
`__enumext_mini_right_cmd:n` . 49, 50, 1268, 1270, 1270
`__enumext_mini_set_vskip:` . . 44, 990, 990, 1091
`__enumext_mini_set_vskip_vii:` 47, 1185, 1185, 1244
`__enumext_mini_set_vskip_viii:` 47, 1185, 1207, 1250
`__enumext_minipage:w` 29, 279, 281, 985, 3099, 3451, 3805
`\l__enumext_minipage_active_v_bool` . . 83, 84, 2928, 2953, 2966, 2974
`\g__enumext_minipage_active_vii_bool` . . . 89, 3199, 3204, 3216
`\l__enumext_minipage_active_vii_bool` . 3184, 3195
`\g__enumext_minipage_active_viii_bool` 3600, 3605, 3617
`\l__enumext_minipage_active_viii_bool` 3585, 3596
`\g__enumext_minipage_active_X_bool` . . . 159
`\l__enumext_minipage_active_X_bool` . . . 73
`\g__enumext_minipage_after_skip` 73, 1189, 1201, 3214, 3615
`\l__enumext_minipage_after_skip` 44, 45, 81, 84, 73, 1006, 1021, 1041, 1057, 1072, 1078, 1084, 1098, 1108, 1117, 1120, 1132, 1150, 1161, 1177, 1209, 1222, 1236, 2849, 2983
`\g__enumext_minipage_center_vii_bool` . 3208, 3217
`\g__enumext_minipage_center_viii_bool` 3609, 3618
`\g__enumext_minipage_center_X_bool` . . . 159
`\l__enumext_minipage_hsep_v_dim` . . . 83, 2926
`\l__enumext_minipage_hsep_vii_dim` . . . 3182
`\l__enumext_minipage_hsep_viii_dim` . . . 3583
`\l__enumext_minipage_left_skip` 44, 83, 73, 998, 1013, 1032, 1047, 1094, 1104, 1109, 1115, 1124, 1141, 1153, 1173, 1183, 1187, 1192, 1196, 1210, 1214, 1228, 1246, 1252
`\l__enumext_minipage_left_v_dim` 83, 2924, 2932
`\l__enumext_minipage_left_vii_dim` 3178, 3190

`\l__enumext_minipage_left_viii_dim` 3579, 3591
`\l__enumext_minipage_left_X_dim` 73
`\g__enumext_minipage_right_skip` 73, 1188, 1193, 1197, 3207, 3608
`\l__enumext_minipage_right_skip` . 44, 73, 1002, 1017, 1037, 1052, 1110, 1116, 1128, 1146, 1157, 1211, 1218, 1232, 1280, 1297
`\l__enumext_minipage_right_v_dim` .. 83, 1291, 1296, 2922, 2926
`\g__enumext_minipage_right_vii_dim` 89, 3186, 3206, 3219
`\l__enumext_minipage_right_vii_dim` 89, 3176, 3181, 3187
`\g__enumext_minipage_right_viii_dim` .. 3587, 3607, 3620
`\l__enumext_minipage_right_viii_dim` .. 3577, 3582, 3588
`\g__enumext_minipage_right_X_dim` 159
`\g__enumext_minipage_right_X_skip` 159
`\g__enumext_minipage_stat_int` . 80, 83, 73, 1285, 1302, 2788, 2842, 2847, 2929, 2976, 2981
`\g__enumext_miniright_code_vii_tl` . 90, 3212, 3218
`\g__enumext_miniright_code_viii_tl` 3613, 3619
`\g__enumext_miniright_code_X_tl` 159
`__enumext_multi_addvspace`: ... 43, 81, 938, 938, 2820
`__enumext_multi_set_vskip`: .. 42, 902, 902, 940
`\l__enumext_multicols_above_ii_skip` ... 921
`\l__enumext_multicols_above_iii_skip` .. 927
`\l__enumext_multicols_above_iv_skip` ... 933
`\l__enumext_multicols_above_v_skip` 957, 971, 981
`\l__enumext_multicols_above_X_skip` 67
`\l__enumext_multicols_below_v_skip` 961, 975, 2968
`\l__enumext_multicols_below_X_skip` 67
`__enumext_multicols_start`: . 80, 81, 2794, 2796, 2796
`__enumext_multicols_stop`: 81, 1275, 2826, 2826, 2851
`__enumext_newlabel:nn` 26, 30, 67, 322, 322, 2141, 2249
`\l__enumext_newlabel_arg_one_tl` 26, 30, 66, 68, 148, 2064, 2134, 2142, 2238, 2250, 2285
`\l__enumext_newlabel_arg_two_tl` 26, 30, 65, 148, 2088, 2098, 2112, 2128, 2143, 2225, 2230, 2235, 2251
`__enumext_parse_keys:n` ... 53, 2663, 2688, 2688
`__enumext_parse_keys_vii:n` 53, 3225, 3256, 3256
`__enumext_parse_keys_viii:n` . 3625, 3658, 3658
`__enumext_parse_series:n` .. 53, 78, 1459, 1459, 2696, 3262
`__enumext_parse_store_keys:n` 78, 79, 2704, 2708, 2708
`__enumext_parse_store_keys_vii:n` . 91, 3265, 3269, 3269
`\l__enumext_parsep_i_skip` 919, 921, 1070, 1118
`\l__enumext_parsep_ii_skip` 925, 927, 1076
`\l__enumext_parsep_iii_skip` ... 931, 933, 1082
`\l__enumext_parsep_vii_skip` 3453
`\l__enumext_parsep_viii_skip` 3807
`\l__enumext_partopsep_v_skip` .. 973, 977, 1144, 1148, 1155, 1159, 1175, 1179
`\l__enumext_partopsep_viii_skip` 1220
`__enumext_phantomsection`: 30, 286, 315, 319, 335
`__enumext_print_footnote`: ... 2391, 2414, 3471, 3832
`__enumext_print_keyans_box:NN` 63, 1953, 1953, 1966, 2154, 2336, 2370, 3749, 3764
`\l__enumext_print_keyans_i_tl` ... 3883, 3912
`\l__enumext_print_keyans_ii_tl` ... 3888, 3913
`\l__enumext_print_keyans_iii_tl` .. 3893, 3914
`\l__enumext_print_keyans_iv_tl` ... 3898, 3915
`\l__enumext_print_keyans_vii_tl` .. 3903, 3916
`\l__enumext_print_keyans_X_tl` 113
`__enumext_printkeyans:nnn` 104, 3917, 3920, 3920
`__enumext_redefine_item`: . 73, 2465, 2465, 2615
`\l__enumext_ref_key_arg_tl` 33, 37, 207, 472, 473, 486, 517, 520, 531, 537, 548, 589, 590, 601
`\l__enumext_ref_the_count_tl` . 33, 37, 479, 482, 485, 525, 527, 530, 542, 544, 547, 595, 597, 600
`__enumext_regex_counter_style`: .. 27, 33, 202, 202, 480, 526, 543, 596
`__enumext_register_counter_style:Nn` .. 354, 354, 359, 360, 361, 362, 363
`__enumext_remove_extra_parsep_vii`: .. 3240, 3480, 3480
`__enumext_remove_extra_parsep_viii`: . 3639, 3841, 3841
`__enumext_renew_footnote`: ... 2391, 2395, 3423, 3786
`\l__enumext_renew_the_count_v_tl` 598, 607, 609
`\l__enumext_renew_the_count_vii_tl` 528, 545, 557, 559
`\l__enumext_renew_the_count_viii_tl` 564, 566
`\l__enumext_renew_the_count_X_tl` 37
`\l__enumext_resume_active_bool` 53, 55, 48, 1463, 1583
`\l__enumext_resume_bool` 23
`__enumext_resume_counter`: .. 55, 56, 1581, 1587, 1594
`__enumext_resume_counter:n` . 53, 55, 1552, 1557, 1581, 1581, 1651, 1659
`__enumext_resume_counter_save_ans`: .. 55, 56, 1581, 1592, 1624
`__enumext_resume_counter_series`: 55, 56, 1581, 1590, 1607
`\g__enumext_resume_int` . 23, 48, 1504, 1598, 1599
`__enumext_resume_last:n` .. 53, 1459, 1465, 1478
`\l__enumext_resume_name_tl` 48, 1500, 1508, 1511, 1527, 1535, 1538, 1584, 1585, 1613, 1620
`__enumext_resume_save_counter`: 54, 1491, 1491, 2862, 3311
`__enumext_resume_series:n` . 55, 1423, 1548, 1548
`__enumext_resume_starred`: . 57, 1424, 1645, 1645
`\g__enumext_resume_vii_int` .. 92, 48, 1531, 1603, 1604
`__enumext_safe_exec`: .. 28, 78, 2662, 2679, 2679
`__enumext_safe_exec_vii`: . 28, 3224, 3245, 3245
`__enumext_safe_exec_viii`: ... 3624, 3644, 3644
`\l__enumext_series_name_tl` 55
`\l__enumext_series_str` . 54, 78, 1421, 1461, 1469, 1470, 1472, 1474, 1495, 1498, 1502, 1522, 1525, 1529, 2692, 3260
`__enumext_set_error:nn` 3998, 4008, 4010
`__enumext_set_parse:n` 3981, 3998, 3998
`\l__enumext_setkey_tmpa_int` ... 108, 3974, 3978

```

\l__enumext_setkey_tmpa_seq . . 108, 3972, 3982,
    3988, 3990, 3992, 4005
\l__enumext_setkey_tmpa_tl . . . 108, 3980, 3984
\l__enumext_setkey_tmpb_seq . . 108, 3973, 3976,
    3980, 3981
\l__enumext_setkey_tmpb_tl 108, 4000, 4002, 4003
\l__enumext_show_answer_bool . 124, 1814, 1838,
    2161, 2311, 2325, 3019, 3747
\__enumext_show_length:nnn . . 39, 210, 210, 4050,
    4051, 4052, 4053, 4054, 4055, 4056, 4057, 4058, 4059,
    4065, 4066, 4067, 4068, 4069, 4070, 4071, 4072, 4073,
    4074
\l__enumext_show_position_bool 124, 1817, 1841,
    2165, 2315, 2326, 3020, 3751
\g__enumext_standar_bool . 28, 20, 231, 234, 256,
    1493, 1558, 1570, 1596, 1609, 1647, 1765, 2872
\l__enumext_standar_bool 82, 20, 2093, 2106, 2122,
    2685, 2861
\g__enumext_standar_keyans_pic_star_env_-
    int . . . . . 145
\g__enumext_standar_keyans_star_env_int 144
\l__enumext_standar_level_one_bool 28, 78, 20,
    261, 1480, 1627, 1678, 1688
\__enumext_standar_ref: . . . 33, 470, 490, 2617
\__enumext_standar_ref:n . . . 33, 462, 470, 470
\g__enumext_standar_series_tl . 48, 1482, 1483,
    1649, 1652
\g__enumext_standar_star_env_int . . 141, 235
\g__enumext_standard_bool . . . . . 78
\l__enumext_standard_bool . . . . . 78
\__enumext_standard_item_vii:w 93, 3351, 3353,
    3353
\__enumext_standard_item_viii:w 99, 3695, 3697,
    3697
\g__enumext_starred_bool 28, 90, 92, 20, 242, 245,
    265, 1520, 1563, 1574, 1601, 1616, 1655, 1744, 2084,
    2094, 2124, 2219, 2743, 2756, 2854, 3220, 3502
\l__enumext_starred_bool . 90, 92, 20, 2017, 2025,
    2109, 2150, 3253, 3310
\__enumext_starred_columns_set_vii: . . 3106,
    3106, 3233
\__enumext_starred_columns_set_viii: . 3507,
    3507, 3632
\__enumext_starred_item:nn . . 2442, 2442, 2471
\__enumext_starred_item_exec: . 100, 3740, 3740,
    3790
\__enumext_starred_item_vii:w . 93, 3350, 3369,
    3369
\__enumext_starred_item_vii_aux_i:w . 3369,
    3374, 3377
\__enumext_starred_item_vii_aux_ii:w . 3369,
    3375, 3380, 3382
\__enumext_starred_item_vii_aux_iii:w 3369,
    3385, 3394
\__enumext_starred_item_viii:w . 99, 100, 3694,
    3713, 3713
\__enumext_starred_item_viii_aux_i:w . 3713,
    3718, 3721
\__enumext_starred_item_viii_aux_ii:w 3713,
    3719, 3733, 3735
\__enumext_starred_joined_item_vii:n . 88, 93,
    3125, 3125, 3348
\__enumext_starred_joined_item_viii:n 96, 99,
    3526, 3526, 3692
\g__enumext_starred_keyans_star_env_int 143
\l__enumext_starred_level_one_bool . 20, 270,
    1485, 1636, 1682, 1694
\__enumext_starred_ref: . . . 34, 515, 553, 2645
\__enumext_starred_ref:n . . . 34, 509, 515, 515
\g__enumext_starred_series_tl . 48, 1487, 1488,
    1657, 1660
\g__enumext_starred_star_env_int . . 142, 246
\__enumext_start_from:NNn 36, 612, 612, 625, 647
\l__enumext_start_i_int . . . . 1599, 1611, 1630
\__enumext_start_item_tmp_vii: 90, 3236, 3333,
    3333
\__enumext_start_item_tmp_viii: 97, 3635, 3677,
    3677
\__enumext_start_item_vii:w . 93, 94, 3361, 3366,
    3391, 3398, 3400, 3400
\__enumext_start_item_viii:w . 99, 3705, 3710,
    3738, 3767, 3767
\__enumext_start_list:nn 29, 75, 87, 273, 275, 2666,
    2882, 3033, 3228, 3627
\__enumext_start_mini_vii: . 91, 3174, 3174, 3303
\__enumext_start_mini_viii: 98, 3575, 3575, 3669
\__enumext_start_store_level: . 79, 2665, 2737,
    2737
\__enumext_start_store_level_vii: . 92, 3227,
    3313, 3313
\l__enumext_start_vii_int . . 1604, 1618, 1639
\l__enumext_start_X_int . . . . . 85, 642
\__enumext_stop_item_tmp_vii: . 90, 92, 94, 3235,
    3239, 3335, 3402
\__enumext_stop_item_tmp_viii: . 97, 99, 3634,
    3638, 3679, 3769
\__enumext_stop_item_vii: 94, 95, 3402, 3456, 3456
\__enumext_stop_item_viii: 102, 3769, 3817, 3817
\__enumext_stop_list: . 29, 273, 276, 2675, 2892,
    3046, 3241, 3641
\__enumext_stop_mini_vii: 89, 92, 3193, 3193, 3307
\__enumext_stop_mini_viii: . 98, 3575, 3594, 3673
\__enumext_stop_store_level: . 79, 2676, 2737,
    2766
\__enumext_stop_store_level_vii: . 92, 3242,
    3313, 3323
\l__enumext_store_active_bool 24, 58, 78, 91, 97,
    1628, 1637, 1704, 1979, 2702, 2741, 2754, 2897, 2904,
    2992, 3050, 3263, 3315, 3325, 3652
\__enumext_store_addto_prop:n 61, 67, 1872, 1873,
    1881, 2004, 2200, 3743
\__enumext_store_addto_seq:n 62, 69, 1882, 1882,
    1886, 1893, 1907, 1915, 1924, 1942, 1950, 2068, 2290
\l__enumext_store_ans_bool . 58, 134, 1705, 1726,
    1889, 1913, 1920, 1948, 1992
\l__enumext_store_anskey_arg_tl 24, 64, 65, 97,
    2010, 2019, 2021, 2027, 2035, 2038, 2048, 2053, 2056,
    2062, 2068
\__enumext_store_anskey_code:nnnn . 64, 1998,
    2002, 2002
\__enumext_store_anskey_show_left:n 67, 2009,
    2159, 2159
\__enumext_store_anskey_show_wrap:n 67, 2147,
    2147, 2163, 2178
\l__enumext_store_columns_break_bool . 1973,
    2016
\l__enumext_store_columns_join_int 97, 2024,
    2029
\l__enumext_store_columns_sep_vii_bool 3284

```

`\l__enumext_store_columns_sep_vii_dim` 3289, 3293
`\l__enumext_store_columns_sep_X_bool` .. 113
`\l__enumext_store_columns_sep_X_dim` ... 113
`\l__enumext_store_columns_vii_bool` ... 3271
`\l__enumext_store_columns_vii_int` 3276, 3280
`\l__enumext_store_columns_X_bool` 113
`\l__enumext_store_columns_X_int` 113
`__enumext_store_internal_ref:` .. 64, 65, 2007, 2070, 2070
`\l__enumext_store_item_symbol_sep_dim` 1971, 2045, 2050
`\l__enumext_store_item_symbol_tl` . 1969, 2036, 2040
`\l__enumext_store_keyans_item_opt_sep_-tl` 1803, 2194, 2196, 2267, 2269, 3726, 3728
`\l__enumext_store_keyans_item_opt_tl` ... 97
`\l__enumext_store_keyans_label_tl` 24, 67, 69, 97, 2183, 2186, 2189, 2196, 2198, 2200, 2257, 2260, 2263, 2269, 2271, 2281, 2290, 2291, 3723, 3728, 3729, 3742, 3743, 3745
`__enumext_store_level_close:` . 62, 1887, 1911, 2770
`__enumext_store_level_close_vii:` 1918, 1946, 3329
`__enumext_store_level_open:` .. 61, 62, 79, 1887, 1887, 2749, 2762
`__enumext_store_level_open_vii:` .. 91, 1918, 1918, 3319
`\g__enumext_store_name_tl` 24, 82, 97, 1785, 1788, 2857, 2874, 3504
`\l__enumext_store_name_tl` 24, 57, 97, 1514, 1517, 1541, 1544, 1632, 1641, 1675, 1676, 1691, 1697, 1706, 1708, 1710, 1712, 1714, 1716, 1875, 1877, 1884, 2136, 2137, 2173, 2240, 2241, 2349, 2362, 2857, 3759
`\l__enumext_store_opt_vii_tl` . 1922, 1932, 1938, 1942, 3278, 3291
`\l__enumext_store_opt_X_tl` 113
`\l__enumext_store_ref_key_bool` 64, 1823, 2005, 2059, 2204, 2278
`\l__enumext_store_upper_level_X_bool` .. 113
`\l__enumext_store_write_aux_file_tl` 26, 67, 69, 148, 2139, 2145, 2247, 2253
`__enumext_storing_exec:` 57, 58, 1673, 1692, 1698, 1702
`__enumext_storing_set:n` .. 57, 1668, 1673, 1673
`\l__enumext_the_counter_v_tl` 597
`\l__enumext_the_counter_vii_tl` 527
`\l__enumext_the_counter_viii_tl` 544
`\l__enumext_the_counter_X_tl` 37
`__enumext_tmp:n` 32, 36, 41, 47, 59, 66, 67, 72, 79, 84, 85, 96, 113, 123, 151, 155, 159, 178, 218, 222, 725, 729, 1417, 1428, 1664, 1672, 1719, 1736, 1793, 1828, 1829, 1846, 2072, 2079, 2080, 2101, 2115, 2118, 2130, 2206, 2213, 2587, 2626, 2627, 2659
`__enumext_tmp:nn` 386, 407, 408, 436, 437, 449, 642, 661, 706, 724, 782, 790, 791, 805, 870, 886, 887, 901, 1306, 1322, 1847, 1871, 2375, 2390
`__enumext_tmp:nnn` 450, 466, 467, 468, 469, 497, 513, 514
`__enumext_tmp:nnnnnn` 662, 687, 690, 693, 695, 697, 700, 703
`__enumext_tmp:w` 3866, 3869
`\l__enumext_tmpa_vii_int` 3116, 3119
`\l__enumext_tmpa_viii_int` 3517, 3520
`\l__enumext_tmpa_X_int` 159
`\l__enumext_topsep_v_skip` 959, 963, 1113, 1126, 1134, 1139, 1159, 1163, 3049, 3080
`\l__enumext_topsep_vii_skip` .. 1190, 1199, 1203
`\l__enumext_topsep_viii_skip` . 1212, 1234, 1238
`\l__enumext_vspace_a_star_v_bool` 1355
`\l__enumext_vspace_a_star_vii_bool` ... 1377
`\l__enumext_vspace_a_star_viii_bool` ... 1388
`\l__enumext_vspace_a_star_X_bool` 85
`__enumext_vspace_above:` .. 50, 1323, 1323, 2775
`__enumext_vspace_above_v:` . 51, 1351, 1351, 2920
`\l__enumext_vspace_above_v_skip` .. 1353, 1357, 1359
`__enumext_vspace_above_vii:` .. 51, 1373, 1373, 3300
`\l__enumext_vspace_above_vii_skip` 1375, 1379, 1381
`__enumext_vspace_above_viii:` . 51, 1373, 1384, 3667
`\l__enumext_vspace_above_viii_skip` 1386, 1390, 1392
`\l__enumext_vspace_b_star_v_bool` 1366
`\l__enumext_vspace_b_star_vii_bool` ... 1399
`\l__enumext_vspace_b_star_viii_bool` ... 1410
`\l__enumext_vspace_b_star_X_bool` 85
`__enumext_vspace_below:` .. 51, 1337, 1337, 2860
`__enumext_vspace_below_v:` . 51, 1362, 1362, 2988
`\l__enumext_vspace_below_v_skip` .. 1364, 1368, 1370
`__enumext_vspace_below_vii:` .. 52, 1395, 1395, 3309
`\l__enumext_vspace_below_vii_skip` 1397, 1401, 1403
`__enumext_vspace_below_viii:` . 52, 1395, 1406, 3675
`\l__enumext_vspace_below_viii_skip` 1408, 1412, 1414
`__enumext_widest_from:nnnn` .. 36, 626, 626, 641, 653
`\g__enumext_widest_label_tl` 23, 31, 55, 374, 378, 382
`\l__enumext_wrap_label_opt_v_bool` 2484
`\l__enumext_wrap_label_opt_vii_bool` 93, 3360
`\l__enumext_wrap_label_opt_viii_bool` 99, 3704
`\l__enumext_wrap_label_opt_X_bool` 85
`\l__enumext_wrap_label_v_bool` 2480, 2484, 2492, 2547
`\l__enumext_wrap_label_vii_bool` 93, 3359, 3364, 3372, 3440
`\l__enumext_wrap_label_viii_bool` .. 99, 3703, 3708, 3716, 3794
`\l__enumext_wrap_label_X_bool` 85
`__enumext_wrapper_label_v:n` 2549, 3028
`__enumext_wrapper_label_vii:n` 3443
`__enumext_wrapper_label_viii:n` 3797
`__enumext_zero_count_level:` 216, 216
`__enumext_zero_parsep:` ... 45, 1010, 1065, 1065
`enumext*` 5, 3222
`enumXi` 346
`enumXii` 346
`enumXiii` 346
`enumXiv` 346
`enumXv` 346

enumXvi	346
enumXvii	346
enumXviii	346
Environments provide by enumext :	
enumext* 22, 23, 25–28, 30, 33, 34, 38–41, 47, 48, 51–55, 57, 59–66, 68, 71, 77–79, 90–92, 94, 95, 97, 99, 101, 103, 106, 108	
enumext 22, 23, 25, 27, 28, 30–37, 39–47, 49–55, 57, 59–64, 66, 68, 71–76, 78, 79, 82, 83, 86, 88, 89, 92, 103, 106, 107	
keyans* 22–24, 26, 27, 30, 33–35, 38–41, 47, 48, 51, 52, 58, 61, 68, 71, 77, 98, 106, 108	
keyanspic 22–25, 30, 32, 35, 49, 58, 61, 62, 67–70, 85–87, 107	
keyans 22–25, 27, 30–32, 35–37, 39–41, 43, 46, 47, 49–51, 58, 61, 62, 67–70, 74–76, 82, 83, 85–87, 89, 99, 106, 107	
Environments:	
list	26, 28, 29, 75, 78
lrbox	88, 94, 95, 101, 102
minipage	26, 29, 41, 43, 44, 85–88, 94, 95, 102
multicols	42–44, 49, 80, 81, 83, 84
exp commands:	
\exp_after:wN	3869
\exp_args:Ne	2699, 3857
\exp_not:N . 45, 377, 485, 530, 547, 600, 739, 753, 754, 765, 766, 777, 778, 2064, 2170, 2171, 2283, 2346, 2347, 2359, 2360, 3756, 3757, 3866	
\exp_not:n 485, 486, 530, 531, 547, 548, 600, 601, 740, 1445, 1457, 1855, 1862, 2029, 2040, 2050, 2064, 2065, 2142, 2250, 2285, 2287, 2719, 2732, 3280, 3293	
F	
\fbox	1798
file commands:	
\file_input_stop:	4180
first	791
font	386
\footnote	71
\footnote	71, 2399
\footnotemark	2409
\footnotesize	2171, 2347, 2360, 3757
\footnotetext	2393
G	
\getkeyans	14, 102, 3855
group commands:	
\group_begin: . . 1991, 2169, 2345, 2358, 3419, 3438, 3755, 3782, 3792, 3877, 3911	
\group_end: 2000, 2176, 2352, 2365, 3448, 3460, 3762, 3802, 3821, 3879, 3918	
H	
\hbadness	3467, 3828
hbox commands:	
\hbox_set:Nn	366
\hfill 416, 420, 425, 426, 1277, 1295, 2064, 2283, 3198, 3599	
hook commands:	
\hook_gput_code:nnn	9, 186, 190, 284
\hook_gset_rule:nnnn	285
\hspace	3478, 3839
\hyperlink	65, 69
\hyperlink	2064, 2283
\hypertarget	30
\hypertarget	314
I	
\IfHyperBoolean	292

\IfPackageLoadedTF	11, 288, 302
\ignorespaces	742
\inputlineno	235, 246
int commands:	
\int_add:Nn	3158, 3559
\int_case:nn	916, 1067, 1739, 1762
\int_compare:nNnTF . . 518, 535, 555, 562, 992, 1111, 1256, 1260, 1264, 1782, 1983, 1987, 2184, 2223, 2228, 2233, 2258, 2341, 2683, 2693, 2746, 2759, 2798, 2814, 2828, 2842, 2866, 2905, 2909, 2938, 2963, 2976, 2996, 3000, 3055, 3128, 3138, 3154, 3249, 3317, 3327, 3473, 3482, 3496, 3529, 3539, 3555, 3647, 3653, 3834, 3843, 3978	
\int_compare_p:nNn . . . 232, 243, 257, 258, 266, 267, 1745, 2085, 2095, 2107, 2108, 2123, 2125	
\int_decr:N	3157, 3558
\int_eval:n 1877, 2137, 2171, 2241, 2347, 2360, 2602, 2644, 3146, 3547, 3757	
\int_from_alph:n	620, 634
\int_from_roman:n	622, 636
\int_gadd:Nn	3159, 3560
\int_gdecr:N	1748, 1753, 1756, 1759, 1767
\int_gincr:N 1598, 1603, 1996, 2294, 2430, 2460, 2788, 2929, 3337, 3415, 3681	
\int_gset:Nn	235, 246, 2407
\int_gset_eq:NN 1497, 1504, 1510, 1516, 1524, 1531, 1537, 1543, 2404	
\int_gzero:N 220, 1285, 1302, 1790, 1791, 2847, 2981, 3491, 3852	
\int_if_exist:NTF 1472, 1508, 1514, 1535, 1541, 1714	
\int_incr:N 2682, 2901, 3054, 3248, 3336, 3646, 3680	
\int_mod:nn	3484, 3845
\int_new:N . 20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 110, 118, 131, 132, 139, 140, 141, 142, 143, 144, 145, 156, 162, 163, 164, 165, 166, 1474, 1716	
\int_set:Nn 616, 620, 622, 1611, 1618, 1630, 1639, 1852, 2024, 3093, 3094, 3116, 3127, 3133, 3149, 3467, 3517, 3528, 3534, 3550, 3828, 3974	
\int_set_eq:NN . 1599, 1604, 2714, 3156, 3275, 3557	
\int_step_function:nnN	2101, 2115, 2130
\int_step_inline:nnn	3095, 4001
\int_to_roman:n	194, 2081, 2119
\int_use:N . . 993, 1613, 1620, 1632, 1641, 2602, 2621, 2644, 2700, 2799, 2808, 2823, 2829, 3131, 3132, 3144, 3532, 3533, 3545	
\int_zero:N	3476, 3837
\c_one_int . 3116, 3135, 3141, 3147, 3151, 3154, 3517, 3536, 3542, 3548, 3552, 3555	
\c_zero_int . . 232, 243, 2085, 2095, 2107, 2108, 2123, 2125, 3317, 3327, 3487, 3848	
\item	29, 40, 41, 62, 72, 85, 87, 88, 90, 97
\item 72, 73, 92, 94, 99, 101, 277, 1895, 1901, 1926, 1934, 2021, 2260, 2263, 2467, 2501, 3234, 3236, 3633, 3635, 3745	
\item*	6, 12, 2499
item-pos*	2375
item-sym*	2375
\itemindent	23, 76
\itemindent	75
itemindent	706
\itemsep	86, 87
\itemsep	3069, 3075
\itemwidth	3123, 3167, 3171, 3524, 3568, 3572
K	
keyans	12, 2877

keyans* 12, 3622

keyanspic 13, 3030

Keys for environments provide by **enumext**:

 above* 24, 50, 51

 above 24, 50, 51, 80, 83, 91, 98

 after 39-41, 82, 85, 92, 98

 align 24, 32, 74, 94

 before* 39, 40, 80, 91, 98

 before 39, 40, 83

 below* 24, 50-52

 below 24, 50-52, 82, 85, 92, 98

 check-ans 24, 25, 27, 58, 59, 64, 69, 70, 72, 73, 80, 82, 95, 106

 columns-sep* 25, 61, 79, 91

 columns-sep 41, 62, 79, 81, 84, 91

 columns* 25, 61, 79, 91

 columns 24, 41, 44, 50, 62, 79, 81, 84, 91

 first 39-41, 94

 font 31, 74, 94

 item-pos* 63, 65, 71

 item-sym* 23, 63, 65, 71, 73

 item*-sep 73

 itemindent 24, 37, 38, 74, 94

 itemsep 37, 77

 labelsep 31, 72, 76, 94

 labelwidth 31, 33, 35, 36, 76

 label 23, 31, 32, 36, 88

 lisparindent 77

 list-indent 23, 37, 38, 87

 list-offset 37, 38

 listparindent 37, 94

 mark-ans 25, 60, 67

 mark-pos 60, 61

 mark-ref 25, 60, 65

 mini-env .. 24, 41, 44, 49, 50, 71, 80, 83, 89, 91, 97, 98

 mini-sep 24, 41, 80, 83

 miniright* 24, 41

 miniright 24, 41, 48, 90

 minirigth* 27

 minirigth 27

 no-store 25, 58, 59

 noitemsep 37, 45

 nosep 37, 45

 parindent 77

 parsep 37, 77, 94

 partopsep 37

 ref 23, 27, 32, 33, 35, 106

 resume* 52, 53, 57, 58, 82

 resume 23, 52-58, 82, 92

 rightmargin 37

 save-ans 24, 53-57, 61, 62, 64, 67, 69, 73, 78, 83, 85, 92, 99, 100, 102, 103, 106

 save-key 25, 53, 78

 save-ref 26, 30, 60, 64, 65, 68, 69, 74, 100

 save-sep 60

 series 52-57, 78, 82

 show-ans 25, 60, 61, 63, 64, 67, 74, 100

 show-length 27, 39, 106

 show-pos 25, 60, 61, 63, 64, 67, 70, 74, 100

 start 24, 27, 36, 53

 store-brk 63, 64

 topsep 37

 widest 23, 27, 36

 wrap-ans 60, 63, 67

 wrap-label* 31, 72, 74, 93, 94, 99

wrap-label 31, 74, 93, 94, 99

wrap-opt 60

keys commands:

 \keys_define:nn 388, 410, 439, 452, 499, 570, 644, 664, 708, 727, 784, 793, 872, 889, 1308, 1419, 1666, 1721, 1795, 1831, 1849, 1967, 2377, 3881, 3944

 \l_keys_key_str 4035

 \keys_set:nn . 402, 896, 1313, 1318, 1560, 1565, 1652, 1660, 2013, 2695, 2699, 2916, 3261, 3662, 3946, 3947, 3948, 3949, 3950, 3951, 3952, 3953, 3954, 3955, 3956, 3957, 3995

keyval commands:

 \keyval_parse:NNn 1433

L

label 450, 497, 570

Labels provide by **enumext**:

 \Alph* 31

 \Roman* 31

 \alph* 31

 \arabic* 27, 31

 \roman* 31

 \labelsep 87

 \labelsep 3070, 3073

 labelsep 386

 \labelwidth 31, 87

 \labelwidth 3070, 3071

 labelwidth 386

 \leftmargin 23, 76

 \leftmargin 75, 3070

legacy commands:

 \legacy_if:nTF 3403, 3406, 3770, 3773

 \legacy_if_gset_false:n 986

 \legacy_if_set_false:n 3405, 3772

 \legacy_if_set_true:n 3365, 3390, 3397, 3410, 3709, 3737, 3777

 \linewidth 80, 83

 \linewidth 2783, 2926, 3092, 3119, 3180, 3520, 3581

 \list 29

 \list 275

 list-indent 706

 list-offset 706

 \listparindent 3072

 listparindent 706

 \lrbox 3420, 3783

M

\makebox 88

\makebox .. 1957, 1959, 2521, 3434, 3442, 3446, 3796, 3800

\makelabel 72, 74, 75, 88

\makelabel 74, 75, 2527, 2543

\makesavenoteenv 308

mark-ans 1793

mark-pos 1793, 1829

mark-ref 1793

mini-env 870

mini-sep 870

\minipage 29

\minipage 281

\miniright 10, 49, 1254, 2845, 2979

\miniright* 10

mode commands:

 \mode_if_vertical:TF 941, 969, 1092, 1171

 \mode_leave_vertical: .. 739, 753, 765, 777, 1926, 1934, 1955, 2519, 3432

msg commands:

\msg_error:nn	.. 2907, 2911, 2998, 3057, 3251, 3649, 3655, 3958
\msg_error:nnn	475, 522, 539, 592, 1258, 1262, 1287, 1304, 1572, 1576, 1680, 1684, 3871, 3876, 3941, 4011
\msg_error:nnnn	1981, 1985, 1989, 2899, 2994, 3002
\msg_fatal:nn 2684
\msg_fatal:nnn 340
\msg_info:nnn 13, 16, 290, 304
\msg_line_context:	.. 4039, 4044, 4049, 4064, 4079, 4083, 4088, 4100, 4104, 4108, 4113, 4118, 4123, 4128, 4132, 4137, 4142, 4146, 4151, 4155, 4160, 4165, 4170, 4174, 4178
\msg_new:nnn	4012, 4016, 4020, 4024, 4029, 4033, 4037, 4042, 4047, 4062, 4077, 4081, 4086, 4090, 4097, 4102, 4106, 4111, 4116, 4121, 4126, 4130, 4135, 4140, 4144, 4149, 4153, 4158, 4163, 4168, 4172, 4176
\msg_note:nnnn 1690, 1696
\msg_term:nnn 1785
\msg_term:nnnn 2611, 2621, 2650, 2655
\msg_warning:nn 2844, 2978
\msg_warning:nnn 1788
\msg_warning:nnnn	2301, 2559, 2564, 3130, 3143, 3531, 3544
\multicolsep 81, 84
\multicolsep 2813, 2951

N

\NeedsTeXFormat 3
\newcounter 343
\NewDocumentCommand	1254, 1977, 2990, 3855, 3909, 3965
\NewDocumentEnvironment	.. 2660, 2877, 3030, 3222, 3622
\newlabel 30
\newlabel 326
no-store 1719
\noindent 90, 97
\noindent	.. 2790, 2931, 3189, 3235, 3475, 3590, 3634, 3836
\nointerlineskip 2790, 2931, 3189, 3590
noitemsep 662
\nopagebreak 952, 980, 1103, 1182, 1245, 1251
\normalfont 2170, 2346, 2359, 3756
nosep 662

P

Packages:

enumext 22, 32, 57, 76, 85, 105
enumitem 30, 31
expl3 88
footnotehyper 29
hyperref 25, 26, 29, 30, 65, 69, 94, 105
lua-visual-debug 44
multicol 22, 105
shortlst 88
\par	952, 980, 1103, 1182, 1245, 1251, 1280, 1297, 2149, 2834, 2849, 2968, 2983, 3104, 3207, 3214, 3475, 3489, 3608, 3615, 3836, 3850
\parindent 3452, 3806
\parsep 42, 45, 86, 87
\parsep 1927, 1935, 2641, 3069, 3076, 3081
parsep 662
\parskip 3453, 3807
\partopsep 87
\partopsep 2642, 3074
partopsep 662

peek commands:

\peek_meaning:NTF	3342, 3356, 3373, 3384, 3686, 3700, 3717
\peek_meaning_remove:NTF 3349, 3693
\peek_remove_spaces:n 2505
\phantomsection 30
\phantomsection 315
prg commands:	
\prg_do_nothing: 319
\prg_new_protected_conditional:Npnn	.. 196
\prg_replicate:nn 213, 4095
\prg_return_false: 200
\prg_return_true: 199
\printkeyans 14, 103, 3909
prop commands:	
\prop_count:N	1877, 2137, 2173, 2241, 2349, 2362, 3759
\prop_gput_if_not_in:Nnn 1872, 1875
\prop_if_exist:NTF 1706, 3875
\prop_item:Nn 3878
\prop_new:N 1708
\ProvidesExplPackage 4

R

\raggedcolumns 2822, 2957
\ref 65, 68
ref 450, 497, 570
\refstepcounter 3412, 3779
regex commands:	
\regex_match:nnTF	198, 619, 621, 633, 635, 2712, 2725, 3273, 3286
\regex_replace_once:nnN 206
\renewcommand 485, 530, 547, 600
\RenewDocumentCommand	.. 2399, 2467, 2501, 2527, 2543
\RequirePackage 17
resume 1417
resume* 1417
rightmargin 706
\Roman 31, 36
\Roman 362
\roman 31, 36
\roman 363, 468, 3897

S

save-ans 1664
save-ref 1793
save-sep 1793
scan commands:	
\scan_stop: 87, 3083, 3234, 3633, 3866, 3869
seq commands:	
\seq_clear:N 3972
\seq_const_from_clist:Nn 3960
\seq_count:N 3043, 3976
\seq_gclear:N 2397, 2398
\seq_gput_right:Nn 1884, 2410, 2411
\seq_if_empty:NTF 2416, 3924, 3990
\seq_if_exist:NTF 1710, 3922
\seq_item:Nn 3101
\seq_map_function:NN 3981
\seq_map_inline:Nn	.. 3930, 3935, 3969, 3991, 3992
\seq_map_pairwise_function:NNN 2418
\seq_new:N 111, 112, 129, 157, 158, 1712
\seq_pop_left:NN 3980
\seq_put_right:Nn 3004, 3988, 4005
\seq_set_from_clist:Nn 3973
\seq_set_map_e:NNn 3982

\seq_show:N	3926
series	1417
\setcounter	630, 634, 636, 2602, 2644, 3048
\setenumext	6–9, 104, 3885, 3890, 3895, 3900, 3905, 3965
\setlength	1928, 1936
show-ans	1793, 1829
show-length	782
show-pos	1829
skip commands:	
\skip_add:Nn	921, 927, 933, 943, 947, 971, 975, 1072, 1078, 1084, 1094, 1098, 1120, 1173, 1177, 3069
\skip_eval:n	1927, 1935
\skip_gset:Nn	1193, 1197, 1201
\skip_gzero_new:N	1188, 1189
\skip_horizontal:N	754, 766, 778, 3435, 3449, 3803
\skip_horizontal:n	740, 1956, 1964, 2520, 2522, 3433, 3811
\skip_if_eq:nnTF	919, 925, 931, 995, 1029, 1070, 1076, 1082, 1113, 1118, 1139, 1190, 1212, 1325, 1339, 1353, 1364, 1375, 1386, 1397, 1408
\skip_new:N	69, 70, 74, 75, 76, 77, 78, 133, 176
\skip_set:Nn	904, 908, 957, 961, 998, 1002, 1006, 1013, 1017, 1021, 1032, 1037, 1041, 1047, 1052, 1057, 1115, 1116, 1117, 1124, 1128, 1132, 1141, 1146, 1150, 1153, 1157, 1161, 1192, 1196, 1214, 1218, 1222, 1228, 1232, 1236, 3063, 3077
\skip_set_eq:NN	2600, 2640, 2641, 3452, 3453, 3806, 3807
\skip_use:N	906, 910, 945, 949, 953, 973, 977, 996, 1015, 1024, 1030, 1035, 1039, 1050, 1054, 1055, 1060, 1096, 1100, 1126, 1326, 1330, 1333, 1340, 1344, 1347, 2834
\skip_zero:N	2642, 2813, 2951, 3074, 3075
\skip_zero_new:N	1108, 1109, 1110, 1187, 1209, 1210, 1211
\c_zero_skip	919, 925, 931, 996, 1030, 1070, 1076, 1082, 1113, 1118, 1139, 1190, 1212, 1326, 1340, 1353, 1364, 1375, 1386, 1397, 1408
\small	3887, 3892, 3897, 3902, 3907
\star	2381
start	642
\stepcounter	2403, 3011
str commands:	
\c_backslash_str	4104, 4113, 4114, 4118, 4119, 4123, 4124, 4155, 4156, 4160, 4165, 4166
\c_colon_str	2136, 2240, 3866
\str_case:nn	226
\str_case:nnTF	1440, 1449
\str_clear:N	2692, 3260
\str_count:n	213, 4095
\str_if_empty:N	1461, 1502, 1529
\str_if_eq:nnTF	2603, 2646
\str_if_in:nnTF	3862
\str_new:N	128, 171
\str_set:Nn	442, 443, 444, 1810, 1811, 1834, 1835
\string	308
\strutbox	1000, 1004, 1008, 1019, 1023, 1034, 1043, 1049, 1059, 1072, 1078, 1084, 1115, 1116, 1117, 1120, 1130, 1134, 1143, 1150, 1155, 1163, 1192, 1193, 1196, 1203, 1216, 1224, 1230, 1238, 3079

T

TeX and \TeX 2_ε commands:

\@auxout	324
\@currentenv	226
\protected@write	324

text commands:

\text_expand:n	3858
\textasteriskcentered	1807, 1821
\thepage	330
tl commands:	
\c_space_tl	2328, 4049, 4064
\tl_clear:N	415, 421, 2010, 2183, 2257, 3723
\tl_clear_new:N	372
\tl_const:Nn	37, 356
\tl_gclear:N	1482, 1487, 2303, 2538, 2874, 3218, 3436, 3504, 3619
\tl_gclear_new:N	1469
\tl_gput_right:Nn	357
\tl_greplace_all:Nnn	378
\tl_gset:Nn	1470, 1483, 1488, 2291, 2857, 3379
\tl_gset_eq:NN	374, 2448, 3429
\tl_if_blank:nTF	3427
\tl_if_empty:N	473, 492, 520, 537, 557, 564, 590, 607, 1495, 1500, 1522, 1527, 1585, 1649, 1657, 1676, 1891, 1922, 2036, 2194, 2267, 2299, 2322, 2517, 3726, 4003
\tl_if_empty:nTF	1550
\tl_if_exist:N	1555
\tl_if_novalue:nTF	2011, 2022, 2191, 2265, 2307, 2401, 2426, 2444, 2449, 2478, 2690, 3041, 3258, 3660, 3724, 3967
\tl_map_inline:Nn	204, 375
\tl_new:N	34, 39, 40, 43, 44, 50, 52, 53, 54, 56, 57, 90, 91, 92, 98, 99, 100, 102, 103, 104, 105, 106, 108, 109, 115, 116, 126, 127, 138, 148, 149, 150, 153, 170, 173
\tl_put_left:Nn	1899, 1932, 2019, 2334, 2368, 3742, 3745
\tl_put_right:Nn	373, 483, 528, 545, 598, 1853, 1860, 1903, 1938, 2021, 2027, 2035, 2038, 2048, 2053, 2056, 2062, 2088, 2098, 2112, 2128, 2134, 2139, 2186, 2189, 2196, 2198, 2225, 2230, 2235, 2238, 2247, 2260, 2263, 2269, 2271, 2281, 2717, 2730, 3278, 3291, 3728, 3729, 3883, 3888, 3893, 3898, 3903
\tl_remove_all:Nn	4002
\tl_remove_once:Nn	2076, 2210
\tl_replace_all:Nnn	377
\tl_reverse:N	2075, 2077, 2209, 2211
\tl_set:Nn	45, 342, 416, 420, 425, 426, 472, 517, 589, 737, 751, 763, 775, 1584, 1675, 2167, 2309, 2343, 2356, 2446, 3731, 3753, 4000
\tl_set_eq:NN	383, 478, 481, 525, 527, 542, 544, 595, 597, 2074, 2208, 2221, 2490, 2494, 3022, 3024
\tl_to_str:n	1555, 1561, 1566, 3858
\tl_trim_spaces:n	373, 3988, 4000, 4006
\tl_use:N	379, 382, 494, 559, 566, 609, 808, 812, 816, 820, 824, 828, 832, 836, 840, 844, 848, 852, 856, 860, 864, 868, 1961, 2081, 2089, 2100, 2114, 2119, 2131, 2433, 2439, 2463, 2481, 2485, 2493, 2529, 2530, 2537, 2545, 2546, 2552, 2667, 2883, 3027, 3212, 3439, 3450, 3454, 3613, 3793, 3804, 3810, 3814, 3912, 3913, 3914, 3915, 3916, 3984

token commands:

\token_to_str:N	326
\topsep	1928, 1936
topsep	662
\typeout	236, 247, 294, 297, 307, 308, 1749, 1768

U

\u	207
use commands:	
\use:N	214, 2534, 2669

\use:n	1431, 3864	1935, 3038, 3049, 3490, 3851
\use_none:nn	318	
\usecounter	2601, 2643	
V		
\value	1498, 1504, 1511, 1517, 1525, 1531, 1538, 1544	
\vspace	987, 1330, 1333, 1344, 1347, 1357, 1359, 1368, 1370, 1379, 1381, 1390, 1392, 1401, 1403, 1412, 1414, 1927,	
W		
widest		642
wrap-ans		1793
wrap-label		386
wrap-label*		386
wrap-opt		1793