

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	109
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/l^{pp}l.txt>). The software has the status “maintained”.

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

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

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

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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

(A) value

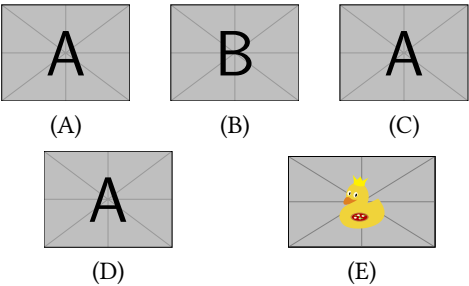
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
-

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

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

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

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

1.1 Description and usage

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

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

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

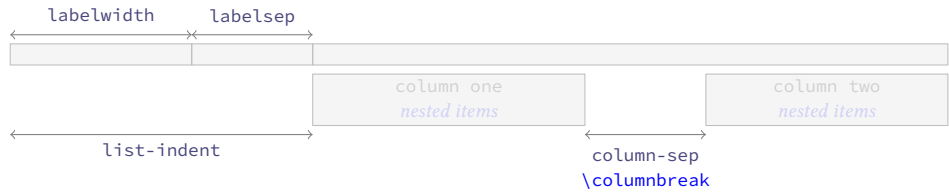


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

1.3.5 Support for \footnote

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

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

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

2 The environment enumext

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

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

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

Example

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

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

2.1 The `\item*` in `enumext`

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

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

3.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

3.5 Keys for multicols

`columns = {⟨integer⟩}` default: 1

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

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

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

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

3.6 Keys for minipage

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

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

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

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

3.6.1 The command `\miniright`

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

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

3.6.2 The key `miniright`

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

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

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

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

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

4 The storage system

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

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

4.1 Keys for storage

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

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

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

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

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

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

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

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

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

4.2 Keys for internal label and ref

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

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

4.3 Keys for debugging and checking

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

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

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

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

4.4 The command \anskey

`\anskey {content}`

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

Example

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

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

4.5 The environment keyans

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

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

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

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

4.5.1 The `\item*` in `keyans`

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

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

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

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

Example

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

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

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

1. Text containing a question.

(A) Choice

* (B) Correct choice

(C) Choice

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

(A) Choice

(B) Choice

(C) Choice

(D) Choice

* (E) [note] Correct choice



4.6 The environment keyanspic

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

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

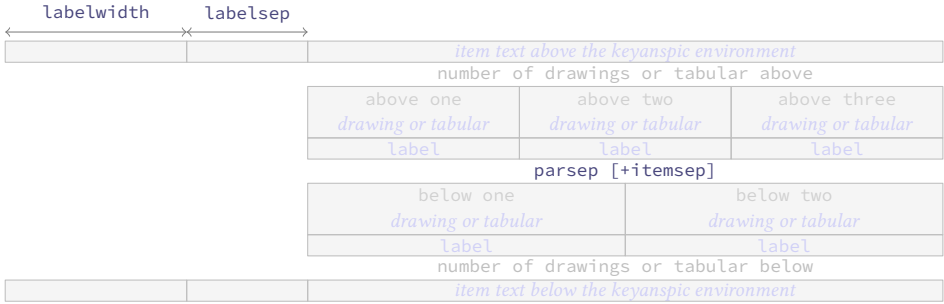


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

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

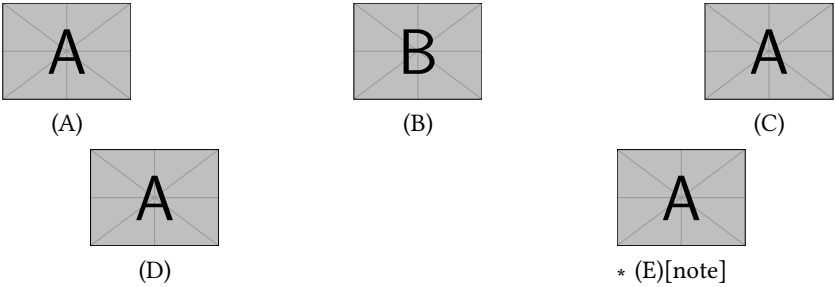
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

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

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

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

(a) ~~LaTeXe~~ is cool?

[2] Very True!
3. Related to Linux

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

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

5 Full examples

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

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

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

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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


C

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

D

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

Example 2

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

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

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

☒ A

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

B

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

C

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

D

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

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

☒ A

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

B

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

C

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

D

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

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

©2024 by Pablo González L

15 / 121

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

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

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

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

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

A

 value

B

 value

C

 value

D

 value

E

 value
4. Question with image and label below:



A



B



C



D



E

5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

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

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

 True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?
- 4

 Related to Linux
- (a)

 You use linux?
- (b)

 Usually uses the package manager?
- (c)

 Rate the following package and class
- i.

 xsim-exam
- ii.

 xsim
- iii.

 exsheets

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

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

Example 5

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

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

1. A) 2. C) 3. B) 4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

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

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

Fake description environment

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

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

Something A short one-line description text.

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

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

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

Something A short one-line description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

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

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

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

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

7 References

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

8 Change history

v1.0 2024-05-21 – 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 / 121

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	\topsep	8
\partopsep	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
\l__enumext_check_starred_cmd_int
\g__enumext_check_starred_cmd_int
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int
```

Internal variables used by “check answer” mechanism (§10.23) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans` environments and `\anspic*` in `keyanspic` environment.

```
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 \tl_new:N \l__enumext_check_start_line_env_tl
140 \int_new:N \l__enumext_check_starred_cmd_int
141 \int_new:N \g__enumext_check_starred_cmd_int
142 \int_new:N \g__enumext_count_item_anskey_int
143 \int_new:N \g__enumext_count_item_number_int
144 \int_new:N \g__enumext_standar_star_env_int
145 \int_new:N \g__enumext_starred_star_env_int
146 \int_new:N \g__enumext_starred_keyans_star_env_int
147 \int_new:N \g__enumext_standar_keyans_star_env_int
148 \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`.

```
149 \bool_new:N \l__enumext_hyperref_bool
150 \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.

```
151 \tl_new:N \l__enumext_newlabel_arg_one_tl
152 \tl_new:N \l__enumext_newlabel_arg_two_tl
153 \tl_new:N \l__enumext_store_write_aux_file_tl
154 \cs_set_protected:Npn \__enumext_tmp:n #1
155 {
156   \tl_new:c { l__enumext_label_copy_#1_tl }
157 }
158 \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`.

```
159 \int_new:N \g__enumext_footnote_int
160 \seq_new:N \g__enumext_footnote_arg_seq
161 \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.

```
162 \cs_set_protected:Npn \__enumext_tmp:n #1
163 {
164   \bool_new:c { l__enumext_item_starred_#1_bool }
165   \int_new:c { l__enumext_item_column_pos_#1_int }
166   \int_new:c { g__enumext_item_count_all_#1_int }
167   \int_new:c { l__enumext_joined_item_#1_int }
168   \int_new:c { l__enumext_joined_item_aux_#1_int }
169   \int_new:c { l__enumext_tmpa_#1_int }
170   \box_new:c { l__enumext_item_text_#1_box }
171   \dim_new:c { l__enumext_joined_width_#1_dim }
172   \dim_new:c { l__enumext_item_width_#1_dim }
173   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
174   \str_new:c { l__enumext_align_label_#1_str }
175   \bool_new:c { g__enumext_minipage_active_#1_bool }
176   \tl_new:c { g__enumext_miniright_code_#1_tl }
177   \bool_new:c { g__enumext_minipage_center_#1_bool }
178   \dim_new:c { g__enumext_minipage_right_#1_dim }
179   \skip_new:c { g__enumext_minipage_right_#1_skip }
180 }
181 \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`.

```
182 \clist_const:Nn \c__enumext_all_envs_clist
183 {
184   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
185   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
186 }
```

(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.

```
187 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
188 {
189   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
190 }
```

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

`__enumext_after_env:nn`

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

```
191 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
192 {
193   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
194 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
195 \cs_new:Nn \__enumext_level:
196 {
197   \int_to_roman:n { \__enumext_level_int }
198 }
```

(End of definition for `__enumext_level:.`)

`__enumext_if_is_int:nT`

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

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?](#).

```
199 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
200 {
201   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
202   { \prg_return_true: }
203   { \prg_return_false: }
204 }
```

(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.

```
205 \cs_new_protected:Nn \__enumext_regex_counter_style:
206 {
207   \tl_map_inline:Nn \c__enumext_counter_style_tl
208   {
209     \regex_replace_once:nnN { \c{##1}\* }
210     { \c{##1}\cB{\u{\__enumext_ref_the_count_tl}\cE} } \__enumext_ref_key_arg_tl
211   }
212 }
```

(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.

```
213 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
214 {
215   * ~ #2
216   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
217   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
218 }
```

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

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

```

219 \cs_set_protected:Nn \__enumext_zero_count_level:
220 {
221   \cs_set_protected:Npn \__enumext_tmp:n ##1
222   {
223     \int_gzero:c { g__enumext_count_level_##1_int }
224   }
225   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
226 }

```

(End of definition for `__enumext_zero_count_level:`.)

`__enumext_current_env_set_bool:` The function `__enumext_current_env_set_bool:` will set the global variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` with which we will distinguish whether the environments `enumext` and `enumext*` are nested in each other.

`__enumext_check_first_level:`

```

227 \cs_new_protected:Nn \__enumext_current_env_set_bool:
228 {
229   \str_case:en { \@currenvir }
230   {
231     {enumext}
232     {
233       \bool_lazy_and:nnT
234       { \bool_not_p:n { \g__enumext_standar_bool } }
235       { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
236       {
237         \bool_gset_true:N \g__enumext_standar_bool
238         \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
239         \typeout{working-on-enumext}
240       }
241     }
242     {enumext*}
243     {
244       \bool_lazy_and:nnT
245       { \bool_not_p:n { \g__enumext_starred_bool } }
246       { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
247       {
248         \bool_gset_true:N \g__enumext_starred_bool
249         \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
250         \typeout{working-on-enumext*}
251       }
252     }
253   }
254 }

```

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.

```

255 \cs_new_protected:Nn \__enumext_check_first_level:
256 {
257   \bool_lazy_all:nT
258   {
259     { \bool_if_p:N \g__enumext_standar_bool }
260     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
261     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
262   }
263   {
264     \bool_set_true:N \l__enumext_standar_level_one_bool
265   }
266   \bool_lazy_all:nT
267   {
268     { \bool_if_p:N \g__enumext_starred_bool }
269     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
270     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
271   }
272   {
273     \bool_set_true:N \l__enumext_starred_level_one_bool
274   }
275 }

```

(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 78) and to the `__enumext_safe_exec_vii:` function in the definition of the `enumext*` environment (pag 90).

`__enumext_keyans_save_start_line:`

The function `__enumext_keyans_save_start_line:` 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.

```

276 \cs_new_protected:Nn \__enumext_keyans_save_start_line:
277 {
278   \str_case:en { \@currenvir }
279   {
280     {keyans}
281     {
282       \tl_set:Nc \l__enumext_check_start_line_env_tl
283       {
284         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
285       }
286       \typeout{working-on-keyans}
287     }
288     {keyans*}
289     {
290       \tl_set:Nc \l__enumext_check_start_line_env_tl
291       {
292         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
293       }
294       \typeout{working-on-keyans*}
295     }
296     {keyanspic}
297     {
298       \tl_set:Nc \l__enumext_check_start_line_env_tl
299       {
300         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
301       }
302       \typeout{working-on-keyanspic}
303     }
304   }
305 }

```

(End of definition for `__enumext_keyans_save_start_line:`.)

10.6 Copying list and minipage environments

The `list` environment provided by \TeX 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.

```

306 \__enumext_at_begin_document:n
307 {
308   \cs_new_eq:NN \__enumext_start_list:nn \list
309   \cs_new_eq:NN \__enumext_stop_list: \endlist
310   \cs_new_eq:NN \__enumext_item_std:w \item
311 }

```

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

The `minipage` environment provided by \TeX 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.

```

312 \__enumext_at_begin_document:n

```

```

313 {
314   \cs_new_eq:NN \__enumext_minipage:w \minipage
315   \cs_new_eq:NN \__enumext_endminipage: \endminipage
316 }

```

(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.

```

317 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
318 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

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

```

319 \cs_new_protected:Nn \__enumext_after_hyperref:
320 {
321   \IfPackageLoadedTF { hyperref }
322   {
323     \msg_info:nnn { enumext } { package-load } { hyperref }
324     \bool_set_true:N \l__enumext_hyperref_bool
325     \IfHyperBoolean{hyperfootnotes}
326     {
327       \typeout{hyperfootnotes=true}
328       \bool_set_true:N \l__enumext_footnotes_key_bool
329     }
330     { \typeout{hyperfootnotes=false} }
331   }
332   { }

```

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`.

```

333   \bool_if:NT \l__enumext_footnotes_key_bool
334   {
335     \IfPackageLoadedTF { footnotehyper }
336     {
337       \msg_info:nnn { enumext } { package-load } { footnotehyper }
338     }
339     {
340       \typeout{No ~ footnotehyper ~ load}
341       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
342       \bool_set_false:N \l__enumext_footnotes_key_bool
343     }
344   }

```

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.

```

345   \bool_if:NTF \l__enumext_hyperref_bool
346   {
347     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
348     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
349   }
350   {
351     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
352     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
353   }
354 }

```

(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.

```

355 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
356 {
357   \protected@write \@auxout { }
358   {
359     \token_to_str:N \newlabel {#1}
360     {
361       {#2}
362       \bool_if:NT \l__enumext_hyperref_bool
363       { { \thepage } {#2} {#1} }
364       { }
365     }
366   }
367   \__enumext_hypertarget:nn {#1} { }
368   \__enumext_phantomsection:
369 }

```

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

10.8 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

```

370 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
371 {
372   \cs_if_exist:cTF { c@ #2 }
373   { \msg_fatal:nnn { enumext } { counters } { #2 } }
374   {
375     \tl_set:Nn #1 { #2 }
376     \newcounter { #2 }
377   }
378 }

```

(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      379 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     380 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    381 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv     382 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXv      383 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXvi     384 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXvii    385 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXviii   386 \__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.

```

387 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
388 {
389   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
390   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
391 }
392 \__enumext_register_counter_style:Nn \arabic { 0 }
393 \__enumext_register_counter_style:Nn \Alph { M }
394 \__enumext_register_counter_style:Nn \alph { m }
395 \__enumext_register_counter_style:Nn \Roman { VIII }
396 \__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.

```
397 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
398 {
399   \hbox_set:Nn \__enumext_label_width_by_box {#2}
400   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
401 }
402 \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.

```
403 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
404 {
405   \tl_clear_new:N #1
406   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
407   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
408   \tl_map_inline:Nn \g__enumext_counter_styles_tl
409   {
410     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
411     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
412     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
413   }
414   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
415   { \tl_use:N \g__enumext_widest_label_tl }
416   \tl_set_eq:cN { the #2 } #1
417 }
418 \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.

```
419 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
420 {
421   \keys_define:nn { enumext / #1 }
422   {
423     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
424     font      .value_required:n = true,
425     labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
426     labelsep   .initial:n   = {0.3333em},
427     labelsep   .value_required:n = true,
428     labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
429     labelwidth .value_required:n = true,
430     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
431     wrap-label .initial:n   = {##1},
432     wrap-label .value_required:n = true,
433     wrap-label* .code:n = {
434       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
435       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
436     },
437     wrap-label* .value_required:n = true,
438   }
439 }
440 \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.

```

441 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
442 {
443   \keys_define:nn { enumext / #1 }
444   {
445     align .choice:,
446     align / left .code:n =
447       {
448         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
449         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
450       },
451     align / right .code:n =
452       {
453         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
454         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
455       },
456     align / center .code:n =
457       {
458         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
459         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
460       },
461     align .initial:n = left,
462     align .value_required:n = true,
463   }
464 }
465 \clist_map_inline:nn
466 {
467   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
468 }
469 { \__enumext_tmp:nn #1 }

470 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
471 {
472   \keys_define:nn { enumext / #1 }
473   {
474     align .choice:,
475     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
476     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
477     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
478     align .initial:n = left,
479     align .value_required:n = true,
480   }
481 }
482 \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 $\langle label \rangle$, 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 $\langle labels \rangle$ of the *four levels* of `enumext` environment, along with the default value for `labelwidth` key and `ref` key.

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

483 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
484 {
485   \keys_define:nn { enumext / #1 }
486   {
487     label .code:n = {
488       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
489       { l__enumext_counter_#2_tl } {##1}
490       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
491       \l__enumext_current_widest_dim
492     },
493     label .initial:n = #3,
494     label .value_required:n = true,
495     ref .code:n = \__enumext_standar_ref:n {##1},
496     ref .value_required:n = true,
497   }

```

```

498   }
499   \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
500   \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
501   \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
502   \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

The `__enumext_standar_ref:n` first we will pass the key argument to `\l__enumext_ref_key_arg_tl` and we will analyze its state, if it is not *empty* we will make a copy of the current counter in `\l__enumext_ref_the_count_tl` and we will execute the function `__enumext_regex_counter_style:` which will return the modified `\l__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.

```

503 \cs_new_protected:Npn \__enumext_standar_ref:n #1
504 {
505   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
506   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
507   {
508     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
509   }
510   {
511     \tl_set_eq:Nc
512     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
513     \__enumext_regex_counter_style:
514     \tl_set_eq:Nc
515     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
516     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
517     {
518       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
519       { \exp_not:V \l__enumext_ref_key_arg_tl }
520     }
521   }
522 }

```

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

```

523 \cs_new_protected:Nn \__enumext_standar_ref:
524 {
525   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
526   {
527     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
528   }
529 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
530 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
531 {
532   \keys_define:nn { enumext / #1 }
533   {
534     label .code:n = {
535       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
536       { \l__enumext_counter_#2_tl } {##1}
537       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
538       \l__enumext_current_widest_dim
539     },
540     label .initial:n = #3,
541     label .value_required:n = true,
542     ref .code:n = \__enumext_starred_ref:n {##1},
543     ref .value_required:n = true,
544   }
545 }
546 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
547 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

_enumext_starred_ref:n
_enumext_starred_ref:

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

```

548 \cs_new_protected:Npn \\_enumext_starred_ref:n #1
549 {
550   \tl_set:Nn \\_enumext_ref_key_arg_tl {#1}
551   \int_compare:nNnT { \\_enumext_level_h_int } = { 1 }
552   {
553     \tl_if_empty:NTF \\_enumext_ref_key_arg_tl
554     {
555       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
556     }
557     {
558       \tl_set_eq:NN \\_enumext_ref_the_count_tl \\_enumext_counter_vii_tl
559       \\_enumext_regex_counter_style:
560       \tl_set_eq:NN \\_enumext_ref_the_count_tl \\_enumext_the_counter_vii_tl
561       \tl_put_right:Ne \\_enumext_renew_the_count_vii_tl
562       {
563         \exp_not:N \renewcommand { \exp_not:V \\_enumext_ref_the_count_tl }
564         { \exp_not:V \\_enumext_ref_key_arg_tl }
565       }
566     }
567   }
568   \int_compare:nNnT { \\_enumext_keyans_level_h_int } = { 1 }
569   {
570     \tl_if_empty:NTF \\_enumext_ref_key_arg_tl
571     {
572       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
573     }
574     {
575       \tl_set_eq:NN \\_enumext_ref_the_count_tl \\_enumext_counter_viii_tl
576       \\_enumext_regex_counter_style:
577       \tl_set_eq:NN \\_enumext_ref_the_count_tl \\_enumext_the_counter_viii_tl
578       \tl_put_right:Ne \\_enumext_renew_the_count_viii_tl
579       {
580         \exp_not:N \renewcommand { \exp_not:V \\_enumext_ref_the_count_tl }
581         { \exp_not:V \\_enumext_ref_key_arg_tl }
582       }
583     }
584   }
585 }

```

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.

```

586 \cs_new_protected:Nn \\_enumext_starred_ref:
587 {
588   \int_compare:nNnT { \\_enumext_level_h_int } = { 1 }
589   {
590     \tl_if_empty:NF \\_enumext_renew_the_count_vii_tl
591     {
592       \tl_use:N \\_enumext_renew_the_count_vii_tl
593     }
594   }
595   \int_compare:nNnT { \\_enumext_keyans_level_h_int } = { 1 }
596   {
597     \tl_if_empty:NF \\_enumext_renew_the_count_viii_tl
598     {
599       \tl_use:N \\_enumext_renew_the_count_viii_tl
600     }
601   }
602 }

```

(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.

```

\\_enumext_label_v_tl 603 \keys_define:nn { enumext / keyans }
\\_enumext_label_vi_tl 604 {
605   label .code:n = {
606     \\_enumext_label_style:cvn { \\_enumext_label_v_tl }
607     { \\_enumext_counter_v_tl } {#1}

```

```

608         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
609         \l__enumext_current_widest_dim
610         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
611         { \l__enumext_counter_vi_tl } {#1}
612         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
613         \l__enumext_current_widest_dim
614     },
615     label .initial:n = (\Alph*),
616     label .value_required:n = true,
617     ref .code:n = \__enumext_keyans_ref:n {#1},
618     ref .value_required:n = true,
619 }

```

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:

```

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

```

620 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
621 {
622     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
623     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
624     {
625         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
626     }
627     {
628         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
629         \__enumext_regex_counter_style:
630         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
631         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
632         {
633             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
634             { \exp_not:V \l__enumext_ref_key_arg_tl }
635         }
636     }
637 }

```

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

```

638 \cs_new_protected:Nn \__enumext_keyans_ref:
639 {
640     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
641     {
642         \tl_use:N \l__enumext_renew_the_count_v_tl
643     }
644 }

```

(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.

```

645 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
646 {
647     \__enumext_if_is_int:nTF { #3 }
648     {
649         \int_set:Nn #2 {#3}
650     }
651     {
652         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
653         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
654         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
655         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
656     }
657 }
658 \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: $\langle integer \text{ or } string \rangle$

The second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an $\langle integer \rangle$ or $\langle string \rangle$ 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.

```

659 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
660 {
661   \__enumext_if_is_int:nTF {#4}
662   {
663     \setcounter{enumX#1} { #4 }
664   }
665   {
666     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
667     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
668     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
669     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
670   }
671   \__enumext_label_width_by_box:cv
672   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
673 }
674 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

`start`
`widest`

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

`\l__enumext_start_X_int`

```

675 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
676 {
677   \keys_define:nn { enumext / #1 }
678   {
679     start .code:n = {
680       \__enumext_start_from:ccn
681       { \l__enumext_label_#2_tl }
682       { \l__enumext_start_#2_int } {##1}
683     },
684     start .initial:n = 1,
685     widest .code:n = {
686       \__enumext_widest_from:nccn {#2}
687       { \l__enumext_label_#2_tl }
688       { \l__enumext_labelwidth_#2_dim } {##1}
689     },
690     widest .value_required:n = true,
691     start .value_required:n = true,
692   }
693 }
694 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.13 Setting keys for vertical spaces

`topsep`
`partopsep`
`parsep`
`noitemsep`
`nosep`

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

```

695 \cs_set_protected:Npn \__enumext_tmp:nnnnn #1 #2 #3 #4 #5 #6
696 {
697   \keys_define:nn { enumext / #1 }
698   {
699     topsep .skip_set:c = { \l__enumext_topsep_#2_skip },
700     topsep .initial:n = {#3},
701     topsep .value_required:n = true,
702     partopsep .skip_set:c = { \l__enumext_partopsep_#2_skip },
703     partopsep .initial:n = {#4},
704     partopsep .value_required:n = true,
705     parsep .skip_set:c = { \l__enumext_parsep_#2_skip },
706     parsep .initial:n = {#5},

```

```

707     parsep      .value_required:n = true,
708     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
709     itemsep     .initial:n = {#6},
710     itemsep     .value_required:n = true,
711     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
712     noitemsep   .value_forbidden:n = true,
713     nosepe     .meta:n = {
714         itemsep = 0pt, parsep= 0pt,
715         topsep = 0pt, partopsep = 0pt,
716     },
717     nosepe     .value_forbidden:n = true,
718 }
719 }

```

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

```

720 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
721 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
722 { 4.0pt plus 2.0pt minus 1.0pt }
723 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
724 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
725 { 2.0pt plus 1.0pt minus 1.0pt }
726 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
727 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
728 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
729 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
730 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
731 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
732 { 2.0pt plus 1.0pt minus 1.0pt }
733 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
734 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
735 { 4.0pt plus 2.0pt minus 1.0pt }
736 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
737 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
738 { 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.

```

739 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
740 {
741     \keys_define:nn { enumext / #1 }
742     {
743         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
744         itemindent .value_required:n = true,
745         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
746         rightmargin .value_required:n = true,
747         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
748         listparindent .value_required:n = true,
749         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
750         list-offset .value_required:n = true,
751         list-indent .code:n =
752             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
753             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
754         list-indent .value_required:n = true,
755     }
756 }
757 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

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

```

758 \cs_set_protected:Npn \__enumext_tmp:n #1
759 {
760     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
761 }
762 \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 `\opt`. 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.

```

763 \cs_set_protected:Nn \__enumext_fake_item:
764 {
765   \dim_compare:nNnT
766     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
767     >
768     { \c_zero_dim }
769   {
770     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
771     {
772       \exp_not:N \mode_leave_vertical:
773       \exp_not:n { \skip_horizontal:n }
774       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
775       \ignorespaces
776     }
777   }
778 }
779 \cs_set_protected:Nn \__enumext_keyans_fake_item:
780 {
781   \dim_compare:nNnT
782     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
783   {
784     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
785     {
786       \exp_not:N \mode_leave_vertical:
787       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
788     }
789   }
790 }
791 \cs_set_protected:Nn \__enumext_fake_item_vii:
792 {
793   \dim_compare:nNnT
794     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
795   {
796     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
797     {
798       \exp_not:N \mode_leave_vertical:
799       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
800     }
801   }
802 }
803 \cs_set_protected:Nn \__enumext_fake_item_viii:
804 {
805   \dim_compare:nNnT
806     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
807   {
808     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
809     {
810       \exp_not:N \mode_leave_vertical:
811       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
812     }
813   }
814 }

```

(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.

```

815 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
816 {
817   \keys_define:nn { enumext / #1 }
818   {
819     show-length .bool_set:c = { l__enumext_show_length_#2_bool },

```



```

820         show-length .initial:n = false,
821     }
822 }
823 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

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

```

before* 824 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    825 {
first    826     \keys_define:nn { enumext / #1 }
          827     {
            828         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
            829         before .value_required:n = true,
            830         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
            831         before* .value_required:n = true,
            832         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
            833         after .value_required:n = true,
            834         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
            835         first .value_required:n = true,
            836     }
          837 }
838 \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

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

```

839 \cs_new_protected:Nn \__enumext_before_args_exec:
840 {
841     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
842 }

```

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

```

843 \cs_new_protected:Nn \__enumext_before_keys_exec:
844 {
845     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
846 }

```

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

```

847 \cs_new_protected:Nn \__enumext_after_stop_list:
848 {
849     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
850 }

```

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

```

851 \cs_new_protected:Nn \__enumext_after_args_exec:
852 {
853     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
854 }

```

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

10.16.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_args_exec_v: 855 \cs_new_protected:Nn \__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v: 856 {
\__enumext_after_stop_list_v: 857     \tl_use:N \l__enumext_before_starred_key_v_tl
\__enumext_after_args_exec_v: 858 }

```

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⟩`.

```
859 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
860 {
861     \tl_use:N \l__enumext_before_no_starred_key_v_tl
862 }
```

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

```
863 \cs_new_protected:Nn \__enumext_after_stop_list_v:
864 {
865     \tl_use:N \l__enumext_after_stop_list_v_tl
866 }
```

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`.

```
867 \cs_new_protected:Nn \__enumext_after_args_exec_v:
868 {
869     \tl_use:N \l__enumext_after_list_args_v_tl
870 }
```

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

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

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

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

```
871 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
872 {
873     \tl_use:N \l__enumext_before_starred_key_vii_tl
874 }
875 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
876 {
877     \tl_use:N \l__enumext_before_starred_key_viii_tl
878 }
```

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⟩`.

```
879 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
880 {
881     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
882 }
883 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
884 {
885     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
886 }
```

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

```
887 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
888 {
889     \tl_use:N \l__enumext_after_stop_list_vii_tl
890 }
891 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
892 {
893     \tl_use:N \l__enumext_after_stop_list_viii_tl
894 }
```

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`.

```
895 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
896 {
897     \tl_use:N \l__enumext_after_list_args_vii_tl
898 }
899 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
900 {
901     \tl_use:N \l__enumext_after_list_args_viii_tl
902 }
```

(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

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

columns

```

903 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
904 {
905   \keys_define:nn { enumext / #1 }
906   {
907     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
908     mini-env .value_required:n = true,
909     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
910     mini-sep .initial:n = 0.3333em,
911     mini-sep .value_required:n = true,
912     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
913     columns-sep .value_required:n = true,
914     columns .int_set:c = { \__enumext_columns_#2_int },
915     columns .initial:n = 1,
916     columns .value_required:n = true,
917   }
918 }
919 \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`.

```

920 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
921 {
922   \keys_define:nn { enumext / #1 }
923   {
924     columns .initial:n = 2,
925     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
926     miniright .value_required:n = true,
927     miniright* .code:n = {
928       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
929       \keys_set:nn { enumext / #1 } { miniright = {##1} }
930     },
931     miniright* .value_required:n = true,
932   }
933 }
934 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for multicols

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

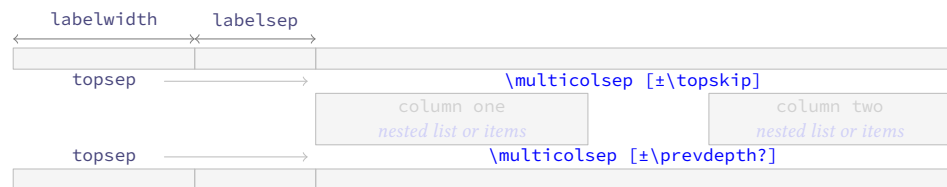


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

10.18.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

935 \cs_new_protected:Nn \__enumext_multi_set_vskip:
936 {
937   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
938   {
939     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
940   }
941   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
942   {
943     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
944   }
945   \__enumext_add_pre_parsep:
946 }
```

(End of definition for `__enumext_multi_set_vskip:`.)

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

```

947 \cs_new_protected:Nn \__enumext_add_pre_parsep:
948 {
949   \int_case:nn { \l__enumext_level_int }
950   {
951     { 2 }{
952       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
953       {
954         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
955       }
956     }
957     { 3 }{
958       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
959       {
960         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
961       }
962     }
963     { 4 }{
964       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
965       {
966         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
967       }
968     }
969   }
970 }
```

(End of definition for `__enumext_add_pre_parsep:`.)

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

```

971 \cs_new_protected:Nn \__enumext_multi_addvspace:
972 {
973   \__enumext_multi_set_vskip:
974   \mode_if_vertical:T
975   {
976     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
977     {
978       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
979     }
980     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
981     {
982       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
983     }
984   }
```

```

985 \par\nopagebreak
986 \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: \skip } }
987 }

```

(End of definition for `\l__enumext_multi_addvspace:`.)

10.18.2 Adjustment of vertical spaces for multicols in keyans

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

```

988 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
989 {
990   \skip_set:Nn \l__enumext_multicols_above_v_skip
991   {
992     \l__enumext_topsep_v_skip
993   }
994   \skip_set:Nn \l__enumext_multicols_below_v_skip
995   {
996     \l__enumext_topsep_v_skip
997   }
998 }
999 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
1000 {
1001   \l__enumext_keyans_multi_set_vskip:
1002   \mode_if_vertical:T
1003   {
1004     \skip_add:Nn \l__enumext_multicols_above_v_skip
1005     {
1006       \skip_use:N \l__enumext_partopsep_v_skip
1007     }
1008     \skip_add:Nn \l__enumext_multicols_below_v_skip
1009     {
1010       \skip_use:N \l__enumext_partopsep_v_skip
1011     }
1012   }
1013   \par\nopagebreak
1014   \addvspace{ \l__enumext_multicols_above_v_skip }
1015 }

```

(End of definition for `\l__enumext_keyans_multi_set_vskip:` and `\l__enumext_keyans_multi_addvspace:`.)

10.19 Adjustment of vertical spaces for minipage

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

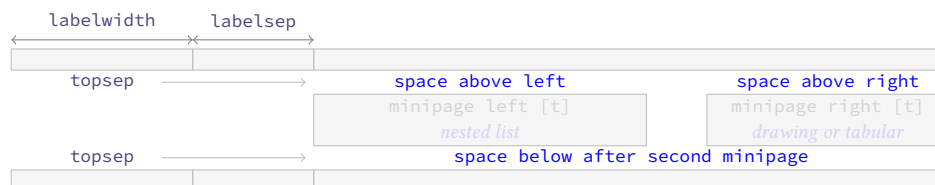


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

`\l__enumext_mini_env*` Creates a `\l__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.

```

1016 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
1017 {
1018     \__enumext_minipage:w [ t ] { #1 }
1019     \legacy_if_gset_false:n { @minipage }
1020     \vspace { 0pt }
1021 }
1022 { \__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 T_EX 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`.

```

1023 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1024 {
1025     \int_compare:nNnTF
1026     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1027     {

```

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

```

1028     \skip_if_eq:nnTF
1029     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1030     {
1031         \skip_set:Nn \l__enumext_minipage_left_skip
1032         {
1033             -0.150\box_dp:N \strutbox
1034         }
1035         \skip_set:Nn \l__enumext_minipage_right_skip
1036         {
1037             0.695\box_dp:N \strutbox
1038         }
1039         \skip_set:Nn \l__enumext_minipage_after_skip
1040         {
1041             \box_dp:N \strutbox
1042         }
1043         \__enumext_zero_parsep:
1044     }
1045     {
1046         \skip_set:Nn \l__enumext_minipage_left_skip
1047         {
1048             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1049         }
1050         \skip_set:Nn \l__enumext_minipage_right_skip
1051         {
1052             0.695\box_dp:N \strutbox
1053         }
1054         \skip_set:Nn \l__enumext_minipage_after_skip
1055         {
1056             1.85\box_dp:N \strutbox
1057             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1058         }
1059     }
1060 }
1061 {

```

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.

```

1062     \skip_if_eq:nnTF
1063     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1064     {
1065         \skip_set:Nn \l__enumext_minipage_left_skip

```

```

1066         {
1067             0.5\box_dp:N \strutbox
1068             - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1069         }
1070     \skip_set:Nn \l__enumext_minipage_right_skip
1071     {
1072         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1073     }
1074     \skip_set:Nn \l__enumext_minipage_after_skip
1075     {
1076         1.6\box_dp:N \strutbox
1077     }
1078 }
1079 {
1080     \skip_set:Nn \l__enumext_minipage_left_skip
1081     {
1082         0.5875\box_dp:N \strutbox
1083         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1084     }
1085     \skip_set:Nn \l__enumext_minipage_right_skip
1086     {
1087         + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1088         + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1089     }
1090     \skip_set:Nn \l__enumext_minipage_after_skip
1091     {
1092         0.325\box_dp:N \strutbox
1093         + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1094     }
1095 }
1096 }
1097 }

```

(End of definition for `\l__enumext_mini_set_vskip:`)

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

```

1098 \cs_new_protected:Nn \l__enumext_zero_parsep:
1099 {
1100     \int_case:nn { \l__enumext_level_int }
1101     {
1102         { 2 } {
1103             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1104             {
1105                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1106             }
1107         }
1108         { 3 } {
1109             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1110             {
1111                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1112             }
1113         }
1114         { 4 } {
1115             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1116             {
1117                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1118             }
1119         }
1120     }
1121 }

```

(End of definition for `\l__enumext_zero_parsep:`)

`\l__enumext_mini_addvspace:` The function `\l__enumext_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\l__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*.

```

1122 \cs_new_protected:Nn \l__enumext_mini_addvspace:
1123 {

```



```

1124 \__enumext_mini_set_vskip:
1125 \mode_if_vertical:T
1126 {
1127   \skip_add:Nn \l__enumext_minipage_left_skip
1128   {
1129     \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1130   }
1131   \skip_add:Nn \l__enumext_minipage_after_skip
1132   {
1133     \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1134   }
1135 }
1136 \par\nopagebreak
1137 \addvspace { \l__enumext_minipage_left_skip }
1138 }

```

(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`.

```

1139 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1140 {
1141   \skip_zero_new:N \l__enumext_minipage_after_skip
1142   \skip_zero_new:N \l__enumext_minipage_left_skip
1143   \skip_zero_new:N \l__enumext_minipage_right_skip
1144   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1145   {
1146     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1147     {
1148       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1149       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1150       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1151       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1152       {
1153         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1154       }
1155     }
1156     {
1157       \skip_set:Nn \l__enumext_minipage_left_skip
1158       {
1159         \skip_use:N \l__enumext_topsep_v_skip
1160       }
1161       \skip_set:Nn \l__enumext_minipage_right_skip
1162       {
1163         0.705\box_dp:N \strutbox
1164       }
1165       \skip_set:Nn \l__enumext_minipage_after_skip
1166       {
1167         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1168       }
1169     }
1170   }
1171   {
1172     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1173     {
1174       \skip_set:Nn \l__enumext_minipage_left_skip
1175       {
1176         0.5\box_dp:N \strutbox
1177         + \l__enumext_partopsep_v_skip
1178       }
1179       \skip_set:Nn \l__enumext_minipage_right_skip
1180       {
1181         \l__enumext_partopsep_v_skip
1182       }
1183       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1184     }
1185     {
1186       \skip_set:Nn \l__enumext_minipage_left_skip
1187       {

```

```

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

```

(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`.

```

1201 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1202 {
1203     \__enumext_keyans_mini_set_vskip:
1204     \mode_if_vertical:T
1205     {
1206         \skip_add:Nn \l__enumext_minipage_left_skip
1207         {
1208             \l__enumext_partopsep_v_skip
1209         }
1210         \skip_add:Nn \l__enumext_minipage_after_skip
1211         {
1212             \l__enumext_partopsep_v_skip
1213         }
1214     }
1215     \par\nopagebreak
1216     \addvspace { \l__enumext_minipage_left_skip }
1217 }

```

(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*`.

```

1218 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1219 {
1220     \skip_zero_new:N \l__enumext_minipage_left_skip
1221     \skip_gzero_new:N \g__enumext_minipage_right_skip
1222     \skip_gzero_new:N \g__enumext_minipage_after_skip
1223     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1224     {
1225         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1226         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1227     }
1228     {
1229         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1230         \skip_gset:Nn \g__enumext_minipage_right_skip
1231         {
1232             \l__enumext_topsep_vii_skip
1233         }
1234         \skip_gset:Nn \g__enumext_minipage_after_skip
1235         {
1236             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1237         }
1238     }
1239 }
1240 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1241 {
1242     \skip_zero_new:N \l__enumext_minipage_after_skip

```

```

1243 \skip_zero_new:N \l__enumext_minipage_left_skip
1244 \skip_zero_new:N \l__enumext_minipage_right_skip
1245 \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1246 {
1247   \skip_set:Nn \l__enumext_minipage_left_skip
1248   {
1249     0.5\box_dp:N \strutbox
1250   }
1251   \skip_set:Nn \l__enumext_minipage_right_skip
1252   {
1253     \l__enumext_partopsep_viii_skip
1254   }
1255   \skip_set:Nn \l__enumext_minipage_after_skip
1256   {
1257     1.6\box_dp:N \strutbox
1258   }
1259 }
1260 {
1261   \skip_set:Nn \l__enumext_minipage_left_skip
1262   {
1263     0.5875\box_dp:N \strutbox
1264   }
1265   \skip_set:Nn \l__enumext_minipage_right_skip
1266   {
1267     \l__enumext_topsep_viii_skip
1268   }
1269   \skip_set:Nn \l__enumext_minipage_after_skip
1270   {
1271     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1272   }
1273 }
1274 }

```

(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 \TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1275 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1276 {
1277   \__enumext_mini_set_vskip_vii:
1278   \par\nopagebreak
1279   \addvspace { \l__enumext_minipage_left_skip }
1280 }
1281 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1282 {
1283   \__enumext_mini_set_vskip_viii:
1284   \par\nopagebreak
1285   \addvspace { \l__enumext_minipage_left_skip }
1286 }

```

(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 \TeX 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.

```

1287 \NewDocumentCommand \miniright { s }
1288 {
1289   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1290   {
1291     \msg_error:nnn { enumext } { wrong-miniright-place }

```

```

1292     }
1293     \int_compare:nNt { \__enumext_level_int } = { 0 }
1294     {
1295         \msg_error:nnn { enumext } { wrong-miniright-place }
1296     }
1297     \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1298     {
1299         \__enumext_keyans_mini_right_cmd:n {#1}
1300     }
1301     { \__enumext_mini_right_cmd:n {#1} }
1302 }

```

(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.

```

1303 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1304 {
1305     \dim_compare:nNtF
1306     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1307     {
1308         \__enumext_multicols_stop:
1309         \end{__enumext_mini_env*}
1310         \hfill
1311         \begin{__enumext_mini_env*}
1312         { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1313         \par\addvspace { \__enumext_minipage_right_skip }
1314         \bool_if:nF {#1}
1315         {
1316             \centering
1317         }
1318         \int_gzero:N \g__enumext_minipage_stat_int
1319     }
1320     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1321 }

```

(End of definition for __enumext_mini_right_cmd:n.)

__enumext_keyans_mini_right_cmd:n

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

```

1322 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1323 {
1324     \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1325     {
1326         \__enumext_keyans_multicols_stop:
1327         \end{__enumext_mini_env*}
1328         \hfill
1329         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1330         \par\addvspace { \__enumext_minipage_right_skip }
1331         \bool_if:nF {#1}
1332         {
1333             \centering
1334         }
1335         \int_gzero:N \g__enumext_minipage_stat_int
1336     }
1337     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1338 }

```

(End of definition for __enumext_keyans_mini_right_cmd:n.)

10.20 Setting above and below keys

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

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

```

above* 1339 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1340 {
below* 1341 \keys_define:nn { enumext / #1 }
1342 {
1343   above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1344   above .value_required:n = true,
1345   above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1346             \keys_set:nn { enumext / #1 } { above = {##1} },
1347   above* .value_required:n = true,
1348   below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1349   below .value_required:n = true,
1350   below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1351             \keys_set:nn { enumext / #1 } { below = {##1} },
1352   below* .value_required:n = true,
1353 }
1354 }
1355 \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.

```

1356 \cs_new_protected:Nn \__enumext_vspace_above:
1357 {
1358   \skip_if_eq:nnF
1359   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1360   {
1361     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1362     {
1363       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1364     }
1365     {
1366       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1367     }
1368   }
1369 }

```

(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.

```

1370 \cs_new_protected:Nn \__enumext_vspace_below:
1371 {
1372   \skip_if_eq:nnF
1373   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1374   {
1375     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1376     {
1377       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1378     }
1379     {
1380       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1381     }
1382   }
1383 }

```

(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.

```

1384 \cs_new_protected:Nn \__enumext_vspace_above_v:
1385 {
1386   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1387   {
1388     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1389     {
1390       \vspace*{ \l__enumext_vspace_above_v_skip }
1391     }
1392     { \vspace { \l__enumext_vspace_above_v_skip } }
1393   }
1394 }

```

(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.

```

1395 \cs_new_protected:Nn \__enumext_vspace_below_v:
1396 {
1397   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1398   {
1399     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1400     {
1401       \vspace*{ \l__enumext_vspace_below_v_skip }
1402     }
1403     { \vspace { \l__enumext_vspace_below_v_skip } }
1404   }
1405 }

```

(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:`

```

1406 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1407 {
1408   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1409   {
1410     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1411     {
1412       \vspace*{ \l__enumext_vspace_above_vii_skip }
1413     }
1414     { \vspace { \l__enumext_vspace_above_vii_skip } }
1415   }
1416 }
1417 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1418 {
1419   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1420   {
1421     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1422     {
1423       \vspace*{ \l__enumext_vspace_above_viii_skip }
1424     }
1425     { \vspace { \l__enumext_vspace_above_viii_skip } }
1426   }
1427 }

```

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

`__enumext_vspace_below_vii:`

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.

`__enumext_vspace_below_viii:`

```

1428 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1429 {
1430   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1431   {
1432     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1433     {
1434       \vspace*{ \l__enumext_vspace_below_vii_skip }

```

```

1435     }
1436     { \vspace { \l__enumext_vspace_below_vii_skip } }
1437   }
1438 }
1439 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1440 {
1441   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1442   {
1443     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1444     {
1445       \vspace*{ \l__enumext_vspace_below_viii_skip }
1446     }
1447     { \vspace { \l__enumext_vspace_below_viii_skip } }
1448   }
1449 }

```

(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 `<keys>` passed to the optional argument of the “first level” of the environments `enumext` and `enumext*`, but, discarding some specific `<keys>`.

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

```

series
resume
resume*
1450 \cs_set_protected:Npn \__enumext_tmp:n #1
1451 {
1452   \keys_define:nn { enumext / #1 }
1453   {
1454     series .str_set:N = \l__enumext_series_str,
1455     series .value_required:n = true,
1456     resume .code:n = \__enumext_resume_series:n {##1},
1457     resume* .code:n = \__enumext_resume_starred:,
1458     resume* .value_forbidden:n = true,
1459   }
1460 }
1461 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.21.1 Internal functions for series key

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

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1462 \cs_new:Npn \__enumext_filter_series:n #1
1463 {
1464   \use:e
1465   {
1466     \keyval_parse:NNn
1467     \__enumext_filter_series_key:n
1468     \__enumext_filter_series_pair:nn {#1}
1469   }
1470 }

```

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.

```

1471 \cs_new:Npn \__enumext_filter_series_key:n #1
1472 {
1473   \str_case:nnF {#1}
1474   {
1475     { resume } {}
1476     { resume* } {}
1477   }
1478   { , { \exp_not:n {#1} } }
1479 }

```

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.

```

1480 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1481 {
1482   \str_case:nnF {#1}
1483   {
1484     { series } {}

```



```

1485     { resume } {}
1486     { start } {}
1487     { save-ans } {}
1488     { save-key } {}
1489   }
1490   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1491 }

```

(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).

```

1492 \cs_new_protected:Npn \__enumext_parse_series:n #1
1493 {
1494   \str_if_empty:NTF \l__enumext_series_str
1495   {
1496     \bool_if:NF \l__enumext_resume_active_bool
1497     {
1498       \__enumext_resume_last:n {#1}
1499     }
1500   }
1501   {
1502     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1503     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1504       { \__enumext_filter_series:n {#1} }
1505     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1506     {
1507       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1508     }
1509   }
1510 }

```

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.

```

1511 \cs_new_protected:Npn \__enumext_resume_last:n #1
1512 {
1513   \bool_if:NT \l__enumext_standar_level_one_bool
1514   {
1515     \tl_gclear:N \g__enumext_standar_series_tl
1516     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1517   }
1518   \bool_if:NT \l__enumext_starred_level_one_bool
1519   {
1520     \tl_gclear:N \g__enumext_starred_series_tl
1521     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1522   }
1523 }

```

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

10.21.2 Internal function to save counter value

```
\__enumext_resume_save_counter:
```

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same *{⟨series name⟩}* but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when

execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§10.32) and the `enumext*` environment definition (§10.35).

```

1524 \cs_new_protected:Nn \__enumext_resume_save_counter:
1525 {
1526   \bool_if:NT \__enumext_standar_bool
1527   {
1528     \tl_if_empty:NF \l__enumext_series_str
1529     {
1530       \int_gset_eq:cN
1531       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1532     }
1533     \tl_if_empty:NTF \l__enumext_resume_name_tl
1534     {
1535       \str_if_empty:NT \l__enumext_series_str
1536       {
1537         \int_gset_eq:NN \__enumext_resume_int \value{enumXi}
1538       }
1539     }
1540     {
1541       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1542       {
1543         \int_gset_eq:cN
1544         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1545       }
1546     }
1547     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1548     {
1549       \int_gset_eq:cN
1550       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1551     }
1552   }
1553   \bool_if:NT \__enumext_starred_bool
1554   {
1555     \tl_if_empty:NF \l__enumext_series_str
1556     {
1557       \int_gset_eq:cN
1558       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1559     }
1560     \tl_if_empty:NTF \l__enumext_resume_name_tl
1561     {
1562       \str_if_empty:NT \l__enumext_series_str
1563       {
1564         \int_gset_eq:NN \__enumext_resume_vii_int \value{enumXvii}
1565       }
1566     }
1567     {
1568       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1569       {
1570         \int_gset_eq:cN
1571         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1572       }
1573     }
1574     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1575     {
1576       \int_gset_eq:cN
1577       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1578     }
1579   }
1580 }

```

(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.

```

1581 \cs_new_protected:Npn \__enumext_resume_series:n #1
1582 {
1583   \tl_if_empty:NTF {#1}
1584   {
1585     \__enumext_resume_counter:n { }
1586   }
1587   {
1588     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1589     {
1590       \__enumext_resume_counter:n {#1}
1591       \bool_if:NT \g__enumext_standar_bool
1592       {
1593         \keys_set:nv { enumext / level-1 }
1594         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1595       }
1596       \bool_if:NT \g__enumext_starred_bool
1597       {
1598         \keys_set:nv { enumext / enumext* }
1599         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1600       }
1601     }
1602     {
1603       \bool_if:NT \g__enumext_standar_bool
1604       {
1605         \msg_error:nnn { enumext } { unknown-series } {#1}
1606       }
1607       \bool_if:NT \g__enumext_starred_bool
1608       {
1609         \msg_error:nnn { enumext } { unknown-series } {#1}
1610       }
1611     }
1612   }
1613 }

```

(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`.

```

1614 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1615 {
1616   \bool_set_true:N \l__enumext_resume_active_bool
1617   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1618   \tl_if_empty:NTF \l__enumext_resume_name_tl
1619   {
1620     \__enumext_resume_counter:
1621   }
1622   {
1623     \__enumext_resume_counter_series:
1624   }
1625   \__enumext_resume_counter_save_ans:
1626 }

```

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.

```

1627 \cs_new_protected:Nn \__enumext_resume_counter:
1628 {
1629   \bool_if:NT \g__enumext_standar_bool
1630   {
1631     \int_gincr:N \g__enumext_resume_int
1632     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1633   }
1634   \bool_if:NT \g__enumext_starred_bool
1635   {
1636     \int_gincr:N \g__enumext_resume_vii_int

```

```

1637         \int_set_eq:Nn \l__enumext_start_vii_int \g__enumext_resume_vii_int
1638     }
1639 }

```

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.

```

1640 \cs_new_protected:Nn \__enumext_resume_counter_series:
1641 {
1642     \bool_if:NT \g__enumext_standar_bool
1643     {
1644         \int_set:Nn \l__enumext_start_i_int
1645         {
1646             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1647         }
1648     }
1649     \bool_if:NT \g__enumext_starred_bool
1650     {
1651         \int_set:Nn \l__enumext_start_vii_int
1652         {
1653             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1654         }
1655     }
1656 }

```

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.

```

1657 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1658 {
1659     \bool_lazy_and:nnT
1660     { \bool_if_p:N \l__enumext_standar_level_one_bool }
1661     { \bool_if_p:N \l__enumext_store_active_bool }
1662     {
1663         \int_set:Nn \l__enumext_start_i_int
1664         {
1665             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1666         }
1667     }
1668     \bool_lazy_and:nnT
1669     { \bool_if_p:N \l__enumext_starred_level_one_bool }
1670     { \bool_if_p:N \l__enumext_store_active_bool }
1671     {
1672         \int_set:Nn \l__enumext_start_vii_int
1673         {
1674             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1675         }
1676     }
1677 }

```

(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.

```

1678 \cs_new_protected:Nn \__enumext_resume_starred:
1679 {
1680     \bool_if:NT \g__enumext_standar_bool
1681     {
1682         \tl_if_empty:NF \g__enumext_standar_series_tl
1683         {
1684             \__enumext_resume_counter:n { }
1685             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1686         }
1687     }
1688     \bool_if:NT \g__enumext_starred_bool
1689     {
1690         \tl_if_empty:NF \g__enumext_starred_series_tl
1691         {

```

```

1692         \__enumext_resume_counter:n { }
1693         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1694     }
1695 }
1696 }

```

(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*`.

```

1697 \cs_set_protected:Npn \__enumext_tmp:n #1
1698 {
1699     \keys_define:nn { enumext / #1 }
1700     {
1701         save-ans .code:n = \__enumext_storing_set:n {##1},
1702         save-ans .value_required:n = true,
1703     }
1704 }
1705 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

10.22.1 Internal functions for `save-ans` key

__enumext_storing_set:n
__enumext_storing_exec:

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

```

1706 \cs_new_protected:Npn \__enumext_storing_set:n #1
1707 {
1708     \tl_set:Nx \l__enumext_store_name_tl {#1}
1709     \tl_if_empty:NTF \l__enumext_store_name_tl
1710     {
1711         \bool_if:NT \__enumext_standar_level_one_bool
1712         {
1713             \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1714         }
1715         \bool_if:NT \__enumext_starred_level_one_bool
1716         {
1717             \msg_error:nnn { enumext* } { save-ans-empty } { enumext* }
1718         }
1719     }
1720     {
1721         \bool_if:NT \__enumext_standar_level_one_bool
1722         {
1723             \msg_note:nnnV
1724             { enumext } { save-ans-ok } { enumext } \l__enumext_store_name_tl
1725             \__enumext_storing_exec:
1726         }
1727         \bool_if:NT \__enumext_starred_level_one_bool
1728         {
1729             \msg_note:nnnV
1730             { enumext* } { save-ans-ok } { enumext* } \l__enumext_store_name_tl
1731             \__enumext_storing_exec:
1732         }
1733     }
1734 }

```

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

```

1735 \cs_new_protected:Nn \__enumext_storing_exec:
1736 {
1737     \bool_set_true:N \l__enumext_store_active_bool

```

```

1738   \bool_set_true:N \l__enumext_store_ans_bool
1739   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1740   {
1741     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1742   }
1743   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1744   {
1745     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1746   }
1747   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1748   {
1749     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1750   }
1751 }

```

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

10.23 The check answer mechanism

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

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

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

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

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

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

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

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

10.23.1 Setting check-ans key

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

check-ans

no-store

```

1752 \cs_set_protected:Npn \__enumext_tmp:n #1
1753 {
1754   \keys_define:nn { enumext / #1 }
1755   {
1756     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1757     check-ans .initial:n = false,
1758     no-store .code:n = {
1759       \bool_set_false:N \l__enumext_store_ans_bool
1760       \bool_set_false:N \l__enumext_check_ans_bool
1761     },
1762     no-store .value_forbidden:n = true,
1763   }
1764 }
1765 \clist_map_inline:nn
1766 {
1767   level-1, level-2, level-3, level-4, enumext*
1768 }
1769 { \__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.

```

1770 \cs_new_protected:Nn \__enumext_check_ans_set:
1771 {
1772   \int_case:nn { \l__enumext_level_int }
1773   {
1774     { 1 }{
1775       \bool_lazy_all:nT
1776       {
1777         { \bool_if_p:N \g__enumext_starred_bool }
1778         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1779       }
1780       {
1781         \int_gdecr:N \g__enumext_count_item_number_int
1782         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1783       }
1784     }
1785     { 2 }{
1786       \int_gdecr:N \g__enumext_count_item_number_int
1787     }
1788     { 3 }{
1789       \int_gdecr:N \g__enumext_count_item_number_int
1790     }
1791     { 4 }{
1792       \int_gdecr:N \g__enumext_count_item_number_int
1793     }
1794   }
1795   \int_case:nn { \l__enumext_level_h_int }
1796   {
1797     { 1 }{
1798       \bool_if:NT \g__enumext_standar_bool
1799       {
1800         \int_gdecr:N \g__enumext_count_item_number_int
1801         \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1802       }
1803     }
1804   }
1805 }

```

(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.

```

1806 \cs_new_protected:Nn \__enumext_check_ans_exec:
1807 {
1808   \bool_if:NT \l__enumext_check_ans_bool
1809   {
1810     \__enumext_check_ans_set:
1811   }
1812 }

```

(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.

```

1813 \cs_new_protected:Nn \__enumext_check_ans_show:
1814 {
1815   \int_compare:nNnTF
1816   { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1817   {
1818     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1819   }
1820   {
1821     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1822   }
1823   \int_gzero:N \g__enumext_count_item_number_int

```



```

1824 \int_gzero:N \g__enumext_count_item_anskey_int
1825 }

```

(End of definition for `__enumext_check_ans_show:`.)

10.23.3 Check for `\item*` and `\anspic*` commands

`__enumext_check_starred_cmd:n`

The function `__enumext_check_starred_cmd:n` performs an extra check for the `keyans`, `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.

```

1826 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1827 {
1828   \int_compare:nNtT
1829   { \l__enumext_check_starred_cmd_int } = { 0 }
1830   {
1831     \msg_warning:nnnV { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1832   }
1833   \int_compare:nNtT
1834   { \l__enumext_check_starred_cmd_int } > { 1 }
1835   {
1836     \msg_warning:nnnV { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1837   }
1838   \tl_clear:N \l__enumext_check_start_line_env_tl
1839   %% \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1840   %% {
1841   %%   \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1842   %% }
1843   %% { \tl_gclear:N \g__enumext_check_ans_item_tl }
1844 }

```

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

10.24 Keys and functions associated with storage

wrap-ans
wrap-opt
save-sep
mark-ans
mark-pos
show-ans
mark-ref
save-ref

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*`.

```

1845 \cs_set_protected:Npn \__enumext_tmp:n #1
1846 {
1847   \keys_define:nn { enumext / #1 }
1848   {
1849     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1850     wrap-ans .initial:n = \fbox{##1},
1851     wrap-ans .value_required:n = true,
1852     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1853     wrap-opt .initial:n = [{##1}],
1854     wrap-opt .value_required:n = true,
1855     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1856     save-sep .initial:n = { , ~ },
1857     save-sep .value_required:n = true,
1858     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1859     mark-ans .initial:n = \textasteriskcentered,
1860     mark-ans .value_required:n = true,
1861     mark-pos .choice:,
1862     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1863     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1864     mark-pos .initial:n = right,
1865     mark-pos .value_required:n = true,
1866     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1867     show-ans .initial:n = false,
1868     show-ans .value_required:n = true,
1869     show-pos .bool_set:N = \l__enumext_show_position_bool,
1870     show-pos .initial:n = false,
1871     show-pos .value_required:n = true,
1872     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
1873     mark-ref .initial:n = \textasteriskcentered,
1874     mark-ref .value_required:n = true,
1875     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1876     save-ref .initial:n = false,
1877     save-ref .value_required:n = true,
1878   }
1879 }
1880 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1881 \cs_set_protected:Npn \__enumext_tmp:n #1
1882 {
1883   \keys_define:nn { enumext / #1 }
1884   {
1885     mark-pos .choice:,
1886     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1887     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1888     mark-pos .initial:n = right,
1889     mark-pos .value_required:n = true,
1890     show-ans .bool_set:N = \__enumext_show_answer_bool,
1891     show-ans .initial:n = false,
1892     show-ans .value_required:n = true,
1893     show-pos .bool_set:N = \__enumext_show_position_bool,
1894     show-pos .initial:n = false,
1895     show-pos .value_required:n = true,
1896   }
1897 }
1898 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1899 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1900 {
1901   \keys_define:nn { enumext / #1 }
1902   {
1903     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1904               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1905               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1906               {
1907                 columns = \exp_not:v { l__enumext_store_columns_#2_int },
1908               },
1909     columns* .value_required:n = true,
1910     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1911               \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1912               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1913               {
1914                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1915               },
1916     columns-sep* .value_required:n = true,
1917   }
1918 }
1919 \clist_map_inline:nn
1920 {
1921   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1922 }
1923 { \__enumext_tmp:nn #1 }

```

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

10.24.1 Function for storing content in prop list

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

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

```

1924 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1925 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1926 {
1927   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
1928   {
1929     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
1930   }

```

```

1931         { #1 }
1932     }
1933 \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` 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`.

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

`__enumext_store_addto_seq:V`

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

```

1934 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1935 {
1936     \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
1937 }
1938 \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:` 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`.

```

1939 \cs_new_protected:Npn \__enumext_store_level_open:
1940 {
1941     \bool_if:NT \l__enumext_store_ans_bool
1942     {
1943         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1944         {
1945             \__enumext_store_addto_seq:n
1946             {
1947                 \item \begin{enumext}
1948             }
1949         }
1950         {
1951             \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1952             {
1953                 \item \begin{enumext} [
1954             }
1955             \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1956             {
1957                 ]
1958             }
1959             \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1960         }
1961     }
1962 }
1963 \cs_new_protected:Npn \__enumext_store_level_close:
1964 {
1965     \bool_if:NT \l__enumext_store_ans_bool
1966     {
1967         \__enumext_store_addto_seq:n { \end{enumext} }
1968     }
1969 }

```

(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.

```

1970 \cs_new_protected:Npn \__enumext_store_level_open_vii:
1971 {
1972     \bool_if:NT \l__enumext_store_ans_bool

```

```

1973 {
1974   \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1975   {
1976     \__enumext_store_addto_seq:n
1977     {
1978       \item \mode_leave_vertical:
1979       \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1980       \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1981     }
1982   }
1983   {
1984     \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1985     {
1986       \item \mode_leave_vertical:
1987       \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1988       \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1989     }
1990     \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1991     {
1992     ]
1993     }
1994     \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1995   }
1996 }
1997 }
1998 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1999 {
2000   \bool_if:NT \l__enumext_store_ans_bool
2001   {
2002     \__enumext_store_addto_seq:n { \end{enumext*} }
2003   }
2004 }

```

(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

```

2005 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2006 {
2007   \mode_leave_vertical:
2008   \skip_horizontal:n { -\dim_use:N #2 }
2009   \makebox[\opt][ r ]
2010   {
2011     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2012     {
2013       \tl_use:N \l__enumext_mark_answer_sym_tl
2014     }
2015   }
2016   \skip_horizontal:n { \dim_use:N #2 }
2017 }
2018 \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.

```

2019 \keys_define:nn { enumext / anskey }
2020 {
2021   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,

```

```

2022     item-sym* .value_required:n = true,
2023     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2024     item-pos* .value_required:n = true,
2025     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2026     store-brk .default:n = true,
2027     store-brk .value_forbidden:n = true,
2028 }

```

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>`.

```

2029 \NewDocumentCommand \anskey { d() s o +m }
2030 {
2031   \bool_if:NF \l__enumext_store_active_bool
2032   {
2033     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2034   }
2035   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2036   {
2037     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2038   }
2039   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2040   {
2041     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2042   }
2043   \group_begin:
2044     \bool_if:NT \l__enumext_store_ans_bool
2045     {
2046       \bool_if:NT \l__enumext_check_ans_bool
2047       {
2048         \int_gincr:N \g__enumext_count_item_anskey_int
2049       }
2050       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2051     }
2052   \group_end:
2053 }

```

(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.

```

2054 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2055 {
2056   \__enumext_store_addto_prop:n {#4}
2057   \bool_if:NT \l__enumext_store_ref_key_bool
2058   {
2059     \__enumext_store_internal_ref:
2060   }
2061   \__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`.

```

2062 \tl_clear:N \l__enumext_store_anskey_arg_tl
2063 \tl_if_novalue:nF {#3}
2064 {
2065   \keys_set:nn { enumext / anskey } {#3}
2066 }
2067 \bool_lazy_and:nnT
2068 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2069 { \bool_not_p:n { \l__enumext_starred_bool } }
2070 {
2071   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2072 }
2073 \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).

```

2074 \tl_if_novalue:nF {#1}
2075 {
2076   \int_set:Nn \l__enumext_store_columns_join_int {#1}
2077   \bool_if:NT \l__enumext_starred_bool
2078   {
2079     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2080     {
2081       ( \exp_not:V \l__enumext_store_columns_join_int )
2082     }
2083   }
2084 }

```

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`.

```

2085 \bool_if:nTF {#2}
2086 {
2087   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2088   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2089   {
2090     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2091     {
2092       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2093     }
2094   }
2095   \dim_compare:nT
2096   {
2097     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2098   }
2099   {
2100     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2101     {
2102       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2103     }
2104   }
2105   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2106 }
2107 {
2108   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2109 }

```

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 ($\langle sequence \rangle$).

```

2110 \bool_lazy_and:nnT
2111 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2112 { \bool_if_p:N \l__enumext_hyperref_bool }
2113 {
2114   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2115   {
2116     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2117     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2118   }
2119 }
2120 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2121 }

```

(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`.

```

2122 \cs_new_protected:Nn \l__enumext_store_internal_ref:
2123 {
2124   \cs_set_protected:Npn \l__enumext_tmp:n ##1
2125   {
2126     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2127     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2128     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }

```

```

2129     \tl_reverse:c { l__enumext_label_copy_##1_tl }
2130   }
2131   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2132   \cs_set:Npn \__enumext_tmp:n ##1
2133     { . \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.

```

2134   \bool_lazy_all:nT
2135   {
2136     { \bool_if_p:N \g__enumext_starred_bool }
2137     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2138   }
2139   {
2140     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2141       { \tl_use:N \l__enumext_label_copy_vii_tl }
2142   }
2143   \bool_lazy_all:nT
2144   {
2145     { \bool_if_p:N \l__enumext_standar_bool }
2146     { \bool_if_p:N \g__enumext_starred_bool }
2147     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2148   }
2149   {
2150     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2151       {
2152         \tl_use:N \l__enumext_label_copy_vii_tl
2153         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2154       }
2155   }

```

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.

```

2156   \bool_lazy_all:nT
2157   {
2158     { \bool_if_p:N \l__enumext_standar_bool }
2159     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2160     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2161     { \bool_not_p:n { \l__enumext_starred_bool } }
2162   }
2163   {
2164     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2165       {
2166         \tl_use:N \l__enumext_label_copy_i_tl
2167         \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2168       }
2169   }
2170   \cs_set:Npn \__enumext_tmp:n ##1
2171     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2172   \bool_lazy_all:nT
2173   {
2174     { \bool_if_p:N \l__enumext_standar_bool }
2175     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2176     { \bool_not_p:n { \g__enumext_starred_bool } }
2177     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2178   }
2179   {
2180     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2181       {
2182         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2183         . \tl_use:N \l__enumext_label_copy_vii_tl
2184       }
2185   }

```

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

```

2186   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2187   {
2188     \l__enumext_store_name_tl \c_colon_str
2189     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2190   }

```


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.

```

2191 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2192 {
2193   \__enumext_newlabel:n
2194   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2195   { \l__enumext_newlabel_arg_two_tl }
2196 }
2197 \l__enumext_store_write_aux_file_tl
2198 }

```

(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.

```

2199 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2200 {
2201   \par
2202   \bool_if:NT \l__enumext_starred_bool
2203   {
2204     \cs_set:Nn \__enumext_level: { vii }
2205   }
2206   \__enumext_print_keyans_box:cc
2207   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2208   { \l__enumext_labelsep_ \__enumext_level: _dim }
2209   \__enumext_anskey_wrapper:n { #1 }
2210 }

```

(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.

```

2211 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2212 {
2213   \bool_if:NT \l__enumext_show_answer_bool
2214   {
2215     \__enumext_store_anskey_show_wrap:n { #1 }
2216   }
2217   \bool_if:NT \l__enumext_show_position_bool
2218   {
2219     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2220     {
2221       \group_begin:
2222       \exp_not:N \normalfont
2223       \exp_not:N \footnotesize [ \int_eval:n
2224         {
2225           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2226         }
2227       ]
2228       \group_end:
2229     }
2230     \__enumext_store_anskey_show_wrap:n { #1 }
2231   }
2232 }

```

(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`.

```

2233 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2234 {

```

```

2235 \tl_clear:N \__enumext_store_keyans_label_tl
2236 \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2237 {
2238   \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_vi_tl }
2239 }
2240 {
2241   \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_v_tl }
2242 }
2243 \tl_if_novalue:nF { #1 }
2244 {
2245   % Set save-sep
2246   \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
2247   {
2248     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
2249   }
2250   \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
2251 }
2252 \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
2253 }

```

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

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

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

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

```

2254 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2255 {
2256   \bool_if:NT \__enumext_store_ref_key_bool
2257   {
2258     \cs_set_protected:Npn \__enumext_tmp:n ##1
2259     {
2260       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2261       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2262       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2263       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2264     }
2265     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2266     \__enumext_keyans_store_ref_aux_i:
2267   }
2268 }

```

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

```

2269 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2270 {
2271   \bool_if:NT \g__enumext_starred_bool
2272   {
2273     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2274   }
2275   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2276   {
2277     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2278     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2279   }
2280   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2281   {
2282     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2283     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2284   }
2285   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2286   {
2287     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2288     { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
2289   }

```

```

2290 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2291 {
2292     \l__enumext_store_name_tl \c_colon_str
2293     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2294 }
2295 \__enumext_keyans_store_ref_aux_ii:
2296 }

```

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.

```

2297 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2298 {
2299     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2300     {
2301         \__enumext_newlabel:nn
2302         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2303         { \l__enumext_newlabel_arg_two_tl }
2304     }
2305     \l__enumext_store_write_aux_file_tl
2306 }

```

(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.

```

2307 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2308 {
2309     \tl_clear:N \l__enumext_store_keyans_label_tl
2310     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2311     {
2312         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2313     }
2314     {
2315         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2316     }
2317     \tl_if_novalue:nF { #1 }
2318     {
2319         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2320         {
2321             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep }
2322         }
2323         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2324     }
2325     \__enumext_keyans_addto_seq_link:
2326 }

```

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_check_starred_cmd:n` and increment the value of the integer variable `\g__enumext_count_item_anskey_int` handled by the `check-ans` key.

```

2327 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2328 {
2329     \bool_lazy_and:nnT
2330     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2331     { \bool_if_p:N \l__enumext_hyperref_bool }
2332     {
2333         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2334         {
2335             \hfill \exp_not:N \hyperlink
2336             {
2337                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2338             }
2339             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2340         }
2341     }

```

```

2341     }
2342     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2343     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2344     \bool_if:NT \l__enumext_check_ans_bool
2345     {
2346         \int_gincr:N \g__enumext_count_item_anskey_int
2347     }
2348 }

```

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

10.26.4 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:
2349 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2350 {
2351     \tl_if_novalue:nF { #1 }
2352     {
2353         \tl_set:Nc \l__enumext_keyans_item_opt_tl { #1 }
2354     }
2355     \bool_if:NT \l__enumext_show_answer_bool
2356     {
2357         \__enumext_keyans_show_ans:
2358     }
2359     \bool_if:NT \l__enumext_show_position_bool
2360     {
2361         \__enumext_keyans_show_pos:
2362     }
2363 }
2364 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2365 {
2366     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2367     {
2368         \bool_lazy_or:nnT
2369         { \bool_if_p:N \l__enumext_show_answer_bool }
2370         { \bool_if_p:N \l__enumext_show_position_bool }
2371         {
2372             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2373         }
2374     }
2375 }
2376 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2377 {
2378     \tl_put_left:Nn \l__enumext_label_v_tl
2379     {
2380         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2381     }
2382 }
2383 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2384 {
2385     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2386     {
2387         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2388         {
2389             \group_begin:
2390             \exp_not:N \normalfont
2391             \exp_not:N \footnotesize [ \int_eval:n
2392             {
2393                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2394             }
2395             ]
2396             \group_end:
2397         }
2398     }
2399     {
2400         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2401         {

```

```

2402         \group_begin:
2403         \exp_not:N \normalfont
2404         \exp_not:N \footnotesize [ \int_eval:n
2405         {
2406             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2407         }
2408         ]
2409         \group_end:
2410     }
2411 }
2412 \tl_put_left:Nn \l__enumext_label_v_tl
2413 {
2414     \__enumext_print_keyans_box:NN
2415     \l__enumext_labelwidth_i_dim
2416     \l__enumext_labelsep_i_dim
2417 }
2418 }

```

(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* 2419 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2420 {
2421     \keys_define:nn { enumext / #1 }
2422     {
2423         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2424         item-sym* .value_required:n = true,
2425         item-sym* .initial:n = { $\star$ },
2426         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2427         item-pos* .value_required:n = true,
2428     }
2429 }
2430 \clist_map_inline:nn
2431 {
2432     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2433 }
2434 { \__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](#).

```

2435 \cs_new_protected:Nn \__enumext_footnotetext:nn
2436 {
2437     \footnotetext[#1]{#2}
2438 }
2439 \cs_new_protected:Nn \__enumext_renew_footnote:
2440 {
2441     \seq_gclear:N \g__enumext_footnote_arg_seq
2442     \seq_gclear:N \g__enumext_footnote_int_seq
2443     \RenewDocumentCommand \footnote { o +m }
2444     {
2445         \tl_if_novalue:nTF {##1}
2446         {
2447             \stepcounter{footnote}
2448             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2449         }
2450         {
2451             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2452         }
2453         \footnotemark [ \g__enumext_footnote_int ]
2454         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2455         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2456     }

```

```

2457 }
2458 \cs_new_protected:Nn \__enumext_print_footnote:
2459 {
2460   \seq_if_empty:NF \g__enumext_footnote_int_seq
2461   {
2462     \seq_map_pairwise_function:NNN
2463     \g__enumext_footnote_int_seq
2464     \g__enumext_footnote_arg_seq
2465     \__enumext_footnotetext:nn
2466   }
2467 }

```

(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).

```

2468 \cs_new_protected:Npn \__enumext_default_item:n #1
2469 {
2470   \tl_if_novalue:nTF {#1}
2471   {
2472     \bool_if:NT \l__enumext_check_ans_bool
2473     {
2474       \int_gincr:N \g__enumext_count_item_number_int
2475     }
2476     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2477     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2478   }
2479   {
2480     \bool_set_eq:cc
2481     { \l__enumext_wrap_label_ \__enumext_level: _bool }
2482     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2483     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2484   }
2485 }

```

(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).

```

2486 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2

```

```

2487 {
2488   \tl_if_novalue:nF {#1}
2489   {
2490     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2491   }
2492   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2493   \tl_if_novalue:nTF {#2}
2494   {
2495     \dim_set_eq:cc
2496     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2497     { \__enumext_labelsep_ \__enumext_level: _dim }
2498   }
2499   {
2500     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2501   }
2502   \bool_if:NT \l__enumext_check_ans_bool
2503   {
2504     \int_gincr:N \g__enumext_count_item_number_int
2505   }
2506   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2507   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2508 }

```

(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).

```

2509 \cs_new_protected:Nn \__enumext_redefine_item:
2510 {
2511   \RenewDocumentCommand \item { s o o }
2512   {
2513     \bool_if:nTF {##1}
2514     {
2515       \__enumext_starred_item:nn {##2} {##3}
2516     }
2517     { \__enumext_default_item:n {##2} }
2518   }
2519 }

```

(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`.

```

2520 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2521 {
2522   \tl_if_novalue:nTF { #1 }
2523   {
2524     \bool_set_true:N \l__enumext_wrap_label_v_bool
2525     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2526   }
2527   {
2528     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2529     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2530   }
2531 }

```

(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`.

```

2532 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2533 {

```



```

2534      \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2535      \__enumext_keyans_show_left:n { #1 }
2536      \bool_set_true:N \l__enumext_wrap_label_v_bool
2537      \__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⟩*.

```

2538      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2539      \__enumext_keyans_addto_prop:n { #1 }
2540      \__enumext_keyans_store_ref:
2541      \__enumext_keyans_addto_seq:n { #1 }
2542      \int_incr:N \l__enumext_check_starred_cmd_int
2543  }

```

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

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

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

```

2544 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2545 {
2546   \RenewDocumentCommand \item { s o }
2547   {
2548     \bool_if:nTF {##1}
2549     {
2550       \peek_remove_spaces:n
2551       {
2552         \__enumext_keyans_starred_item:n {##2}
2553       }
2554     }
2555     {
2556       \__enumext_keyans_default_item:n {##2}
2557     }
2558   }
2559 }

```

(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.

```

2560 \cs_new_protected:Nn \__enumext_item_starred:
2561 {
2562   \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2563   {
2564     \mode_leave_vertical:
2565     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2566     \makebox[ \opt ][ r ] { \g__enumext_item_symbol_tl }
2567     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2568   }
2569 }

```

(End of definition for `__enumext_item_starred:`.)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makeLabel` for the `enumext` environment. This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§10.31.2).

```

2570 \cs_new_protected:Nn \__enumext_make_label:
2571 {
2572   \RenewDocumentCommand \makeLabel { m }
2573   {
2574     \tl_use:c { \l__enumext_label_fill_left_ \l__enumext_level: _tl }

```

```

2575 \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2576 \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2577 {
2578   \__enumext_item_starred:
2579   \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2580 }
2581 { ##1 }
2582 \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2583 \tl_gclear:N \g__enumext_item_symbol_tl
2584 }
2585 }

```

(End of definition for `__enumext_make_label:`)

10.30.2 Redefining `\makeLabel` for keyans

`__enumext_keyans_make_label:`

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

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

```

2586 \cs_new_protected:Nn \__enumext_keyans_make_label:
2587 {
2588   \RenewDocumentCommand \makeLabel { m }
2589   {
2590     \tl_use:N \l__enumext_label_fill_left_v_tl
2591     \tl_use:N \l__enumext_label_font_style_v_tl
2592     \bool_if:NTF \l__enumext_wrap_label_v_bool
2593     {
2594       \__enumext_wrapper_label_v:n { ##1 }
2595     }
2596     { ##1 }
2597     \tl_use:N \l__enumext_label_fill_right_v_tl
2598   }
2599 }

```

(End of definition for `__enumext_keyans_make_label:`)

10.31 Second argument of the lists

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

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

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

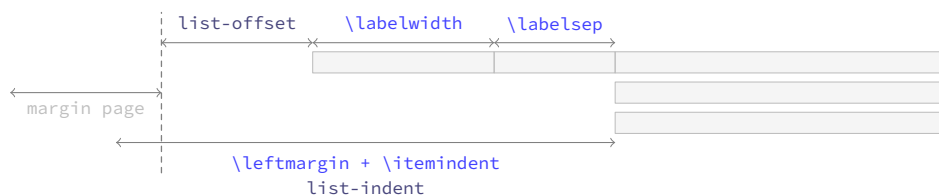


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

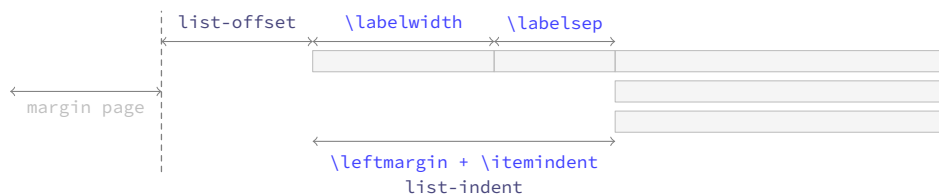
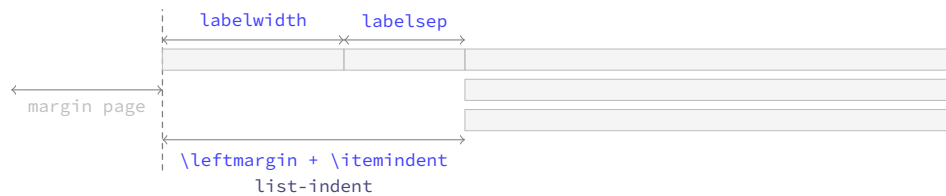


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

`__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:

Figure 11: Default horizontal lengths in `enumext`.

```

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

```

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

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

```

2600 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2601 {
2602   \dim_compare:nNt { #1 } < { \c_zero_dim }
2603   {
2604     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2605     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2606   }
2607   \dim_compare:nNt { #2 } < { \c_zero_dim }
2608   {
2609     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2610     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2611   }

```

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

```

2612   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2613   \dim_compare:nNtF { #4 } < { \c_zero_dim }
2614   {
2615     \dim_set:Nn #6 { #1 + #2 - #4 }
2616     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2617   }
2618   {
2619     \dim_compare:nNt { #4 } = { #1 + #2 }
2620     { \dim_set:Nn #6 { \c_zero_dim } }
2621     \dim_compare:nNt { #4 } < { #1 + #2 }
2622     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2623     \dim_compare:nNt { #4 } > { #1 + #2 }
2624     {
2625       \dim_set:Nn #6 { -#1 - #2 + #4 }
2626       \dim_set:Nn #6 { #6*-1 }
2627     }
2628     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2629   }
2630 }
2631 \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.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
2632 \cs_set_protected:Npn \__enumext_tmp:n #1
2633 {
2634   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2635   {
2636     \__enumext_calc_hspace:ccccccc
2637     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2638     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2639     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2640     { \__enumext_leftmargin_tmp_#1_bool }

```

```

2641 \clist_map_inline:nn
2642 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2643 { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2644 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2645 { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2646 \usecounter { enumX#1 }
2647 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2648 \str_if_eq:nnTF {#1} { v }
2649 {
2650   \__enumext_keyans_redefine_item:
2651   \__enumext_keyans_make_label:
2652   \__enumext_keyans_ref:
2653   \__enumext_keyans_fake_item:
2654   \bool_if:cT { l__enumext_show_length_#1_bool }
2655   {
2656     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2657   }
2658 }
2659 {
2660   \__enumext_redefine_item:
2661   \__enumext_make_label:
2662   \__enumext_standar_ref:
2663   \__enumext_fake_item:
2664   \bool_if:cT { l__enumext_show_length_#1_bool }
2665   {
2666     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N l__enumext_level_#1_int }
2667   }
2668 }
2669 }
2670 }
2671 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

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

```

2672 \cs_set_protected:Npn \__enumext_tmp:n #1
2673 {
2674   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2675   {
2676     \__enumext_calc_hspace:ccccc
2677     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2678     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2679     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2680     { l__enumext_leftmargin_tmp_#1_bool }
2681   \clist_map_inline:nn
2682   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2683   { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2684   \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2685   { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2686   \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
2687   \skip_zero:N \partopsep
2688   \usecounter { enumX#1 }
2689   \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2690   \__enumext_starred_ref:
2691   \str_if_eq:nnTF {#1} { vii }
2692   {
2693     \__enumext_fake_item_vii:
2694     %\__enumext_starred_ref:
2695     \bool_if:cT { l__enumext_show_length_vii_bool }
2696     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2697   }
2698   {
2699     \__enumext_fake_item_viii:
2700     %\__enumext_starred_ref:
2701     \bool_if:cT { l__enumext_show_length_#1_bool }
2702     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2703   }
2704 }

```

```

2705 }
2706 \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.

```

2707 \NewDocumentEnvironment{enumext}{0}{}
2708 {
2709   \__enumext_safe_exec:
2710   \__enumext_parse_keys:n {#1}
2711   \__enumext_before_list:
2712   \__enumext_start_store_level:
2713   \__enumext_start_list:nn
2714   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2715   {
2716     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2717     \__enumext_before_keys_exec:
2718   }
2719   \__enumext_after_args_exec:
2720 }
2721 {
2722   \__enumext_stop_list:
2723   \__enumext_stop_store_level:
2724   \__enumext_after_list:
2725 }

```

(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:**.

```

2726 \cs_new_protected:Nn \__enumext_safe_exec:
2727 {
2728   \__enumext_current_env_set_bool:
2729   \int_incr:N \l__enumext_level_int
2730   \int_compare:nNt { \l__enumext_level_int } > { 4 }
2731   { \msg_fatal:nn { enumext } { list-too-deep } }
2732   \bool_set_true:N \l__enumext_standard_bool
2733   \__enumext_check_first_level:
2734 }

```

(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**.

```

2735 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2736 {
2737   \tl_if_novalue:nF {#1}
2738   {
2739     \str_clear:N \l__enumext_series_str
2740     \int_compare:nNtF { \l__enumext_level_int } = { 1 }
2741     {
2742       \keys_set:nn { enumext / level-1 } {#1}
2743       \__enumext_parse_series:n {#1}
2744     }
2745     {
2746       \exp_args:Ne \keys_set:nn
2747       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2748     }
2749     \bool_if:NT \l__enumext_store_active_bool
2750     {
2751       \__enumext_parse_store_keys:n {#1}

```

```

2752     }
2753   }
2754 }

```

(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:`.

```

2755 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2756 {
2757   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2758   {
2759     \regex_match:nnT { \b columns\b } {#1}
2760     {
2761       \int_set_eq:cc
2762       { \l__enumext_store_columns_ \__enumext_level: _int }
2763       { \l__enumext_columns_ \__enumext_level: _int }
2764       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2765       {
2766         columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2767       }
2768     }
2769   }
2770   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2771   {
2772     \regex_match:nnT { \b columns-sep\b } {#1}
2773     {
2774       \dim_set_eq:cc
2775       { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2776       { \l__enumext_columns_sep_ \__enumext_level: _dim }
2777       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2778       {
2779         columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim },
2780       }
2781     }
2782   }
2783 }

```

(End of definition for `__enumext_parse_store_keys:n`)

`__enumext_start_store_level:`

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

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

```

2784 \cs_new_protected:Npn \__enumext_start_store_level:
2785 {
2786   \bool_lazy_all:nT
2787   {
2788     { \bool_if_p:N \l__enumext_store_active_bool }
2789     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2790     { \bool_not_p:n { \g__enumext_starred_bool } }
2791   }
2792   {
2793     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2794     {
2795       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2796       \__enumext_store_level_open:
2797     }
2798   }
2799   \bool_lazy_all:nT
2800   {
2801     { \bool_if_p:N \l__enumext_store_active_bool }
2802     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2803     { \bool_if_p:N \g__enumext_starred_bool }
2804   }
2805   {
2806     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2807     {
2808       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }

```

```

2809         \__enumext_store_level_open:
2810     }
2811 }
2812 }
2813 \cs_new_protected:Nn \__enumext_stop_store_level:
2814 {
2815     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2816     {
2817         \__enumext_store_level_close:
2818     }
2819 }

```

(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.

```

2820 \cs_new_protected:Nn \__enumext_before_list:
2821 {
2822     \__enumext_vspace_above:
2823     \__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.

```

2824     \__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.

```

2825     \dim_compare:nNnT
2826     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2827     {
2828         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2829         {
2830             \linewidth
2831             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2832             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2833         }

```

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*.

```

2834         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2835         \int_gincr:N \g__enumext_minipage_stat_int
2836         \__enumext_mini_addvspace:
2837         \nointerlineskip\noindent
2838         \begin{\__enumext_mini_env*}
2839             { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2840     }
2841     \__enumext_multicols_start:
2842 }

```

(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.

```

2843 \cs_new_protected:Nn \__enumext_multicols_start:
2844 {
2845     \int_compare:nNnT
2846     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2847     {
2848         \dim_compare:nNnT
2849         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }

```



```

2850         {
2851             \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
2852             {
2853                 ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
2854                   + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
2855                   ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
2856                   - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
2857             }
2858         }
2859         \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
2860         \skip_zero:N \multicolsep
2861         \int_compare:nNnT { \__enumext_level_int } > { 1 }
2862         {
2863             \dim_zero:N \columnseprule
2864         }

```

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.

```

2865         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2866         {
2867             \__enumext_multi_addvspace:
2868         }
2869         \raggedcolumns
2870         \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2871     }
2872 }

```

(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.

```

2873 \cs_new_protected:Nn \__enumext_multicols_stop:
2874 {
2875     \int_compare:nNnT
2876     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2877     {
2878         \end{multicols}
2879         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2880         {
2881             \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2882         }
2883     }
2884 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_after_list:`

The function `__enumext_after_list:` will 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.

```

2885 \cs_new_protected:Nn \__enumext_after_list:
2886 {
2887     \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
2888     {
2889         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2890         {
2891             \msg_warning:nn { enumext } { missing-miniright }
2892             \miniright
2893         }
2894         \int_gzero:N \g__enumext_minipage_stat_int
2895         \end{__enumext_mini_env*}
2896         \par\addvspace { \__enumext_minipage_after_skip }
2897     }
2898     { \__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`.

```

2899     \bool_lazy_and:nnT

```

```

2900     { \bool_if_p:N \l__enumext_check_ans_bool }
2901     { \bool_not_p:n { \g__enumext_starred_bool } }
2902     {
2903         \bool_gset_true:N \g__enumext_check_ans_show_bool
2904         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2905     }

```

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.

```

2906     \__enumext_after_stop_list:
2907     \__enumext_vspace_below:
2908     \bool_set_false:N \l__enumext_standar_bool
2909     \__enumext_resume_save_counter:
2910 }

```

(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`.

```

2911 \__enumext_after_env:nn {enumext}
2912 {
2913     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2914     {
2915         \bool_if:NT \g__enumext_check_ans_show_bool
2916         {
2917             \__enumext_check_ans_show:
2918         }
2919         \bool_gset_false:N \g__enumext_standar_bool
2920         \bool_gset_false:N \g__enumext_check_ans_show_bool
2921         \tl_gclear:N \g__enumext_store_name_tl
2922     }
2923 }

```

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.

```

2924 \NewDocumentEnvironment{keyans}{0}{}
2925 {
2926     \__enumext_keyans_safe_exec:
2927     \__enumext_keyans_parse_keys:n {#1}
2928     \__enumext_before_list_v:
2929     \__enumext_start_list:nn
2930     { \tl_use:N \l__enumext_label_v_tl }
2931     {
2932         \__enumext_list_arg_two_v:
2933         \__enumext_before_keys_exec_v:
2934     }
2935     \__enumext_after_args_exec_v:
2936 }
2937 {
2938     \__enumext_check_starred_cmd:n { item }
2939     \__enumext_stop_list:
2940     \__enumext_after_list_v:
2941 }

```

(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.

```

2942 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2943 {
2944     \bool_if:NF \l__enumext_store_active_bool
2945     {
2946         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
2947     }
2948     \int_incr:N \l__enumext_keyans_level_int

```

```

2949 \bool_set_true:N \l__enumext_keyans_env_bool
2950 \__enumext_keyans_save_start_line:
2951 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2952 \bool_set_false:N \l__enumext_store_active_bool
2953 \int_compare:nNtT { \l__enumext_keyans_level_int } > { 1 }
2954 {
2955   \msg_error:nn { enumext } { keyans-nested }
2956 }
2957 \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2958 {
2959   \msg_error:nn { enumext } { keyans-wrong-level }
2960 }
2961 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

```

\__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
2962 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2963 {
2964   \keys_set:nn { enumext / keyans } {#1}
2965 }

```

(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.

```

2966 \cs_new_protected:Nn \__enumext_before_list_v:
2967 {
2968   \__enumext_vspace_above_v:
2969   \__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.

```

2970 \dim_compare:nNtT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2971 {
2972   \dim_set:Nn \l__enumext_minipage_left_v_dim
2973   {
2974     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2975   }

```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniiright` command will be incremented, then the function `__enumext_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*.

```

2976 \bool_set_true:N \l__enumext_minipage_active_v_bool
2977 \int_gincr:N \g__enumext_minipage_stat_int
2978 \__enumext_keyans_mini_addvspace:
2979 \nointerlineskip\noindent
2980 \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2981 }

```

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

```

2982 \__enumext_keyans_multicols_start:
2983 }

```

(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.

```

2984 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2985 {
2986   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
2987   {

```

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

```

2988 \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2989 {
2990   \dim_set:Nn \l__enumext_columns_sep_v_dim
2991   {
2992     (
2993       \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2994     ) / \l__enumext_columns_v_int
2995     - \l__enumext_listoffset_v_dim
2996   }
2997 }
2998 \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim

```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```

2999 \skip_zero:N \multicolsep
3000 \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.

```

3001 \bool_if:NF \l__enumext_minipage_active_v_bool
3002 {
3003   \__enumext_keyans_multi_addvspace:
3004 }
3005 \raggedcolumns
3006 \begin{multicols}{ \l__enumext_columns_v_int }
3007 }
3008 }

```

(End of definition for `__enumext_keyans_multicols_start:`)

`__enumext_keyans_multicols_stop:`

The function `__enumext_keyans_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

3009 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3010 {
3011   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3012   {
3013     \end{multicols}
3014     \bool_if:NF \l__enumext_minipage_active_v_bool
3015     {
3016       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3017     }
3018   }
3019 }

```

(End of definition for `__enumext_keyans_multicols_stop:`)

`__enumext_after_list_v:`

The function `__enumext_after_list_v:` will check the state of the boolean variable `\l__enumext_minipage_active_v_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini-env*` environment has not been closed), then close `__enumext_mini-env*` and add the vertical adjustment space `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3020 \cs_new_protected:Nn \__enumext_after_list_v:
3021 {
3022   \bool_if:NTF \l__enumext_minipage_active_v_bool
3023   {
3024     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3025     {
3026       \msg_warning:nn { enumext } { missing-miniright }
3027       \miniright
3028     }
3029     \int_gzero:N \g__enumext_minipage_stat_int
3030     \end{__enumext_mini-env*}
3031     \par\addvspace{ \l__enumext_minipage_after_skip }
3032   }
3033   { \__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.

```

3034 \bool_set_false:N \l__enumext_keyans_env_bool
3035 \__enumext_after_stop_list_v:
3036 \__enumext_vspace_below_v:
3037 }

```

(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.

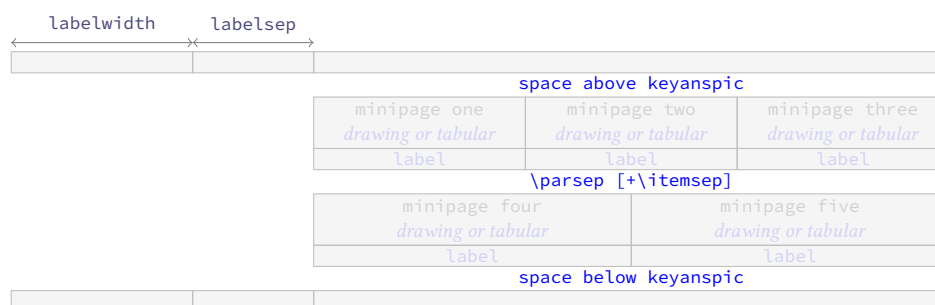


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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.

```

3038 \NewDocumentCommand \anspic { s o +m }
3039 {

```

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

```

3040 \bool_if:NF \l__enumext_store_active_bool
3041 {
3042 \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3043 }
3044 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3045 {
3046 \msg_error:nn { enumext } { keyanspic-wrong-level }
3047 }
3048 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3049 {
3050 \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3051 }

```

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.

```

3052 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3053 {
3054 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3055 }
3056 }

```

(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.

```

3057 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3058 {
3059 \stepcounter { enumXvi }

```

```

3060 #3 \\
3061 \bool_if:nT { #1 }
3062 {
3063   \__enumext_keyans_addto_prop:n { #2 }
3064   \__enumext_keyans_store_ref:
3065   \__enumext_keyans_addto_seq:n { #2 }
3066   \int_incr:N \__enumext_check_starred_cmd_int
3067   \bool_lazy_or:nnT
3068   { \bool_if_p:N \__enumext_show_answer_bool }
3069   { \bool_if_p:N \__enumext_show_position_bool }
3070   {
3071     \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3072     \__enumext_keyans_show_left:n { #2 }
3073     \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3074   }
3075 }
3076 \tl_use:N \__enumext_label_font_style_v_tl
3077 \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3078 }

```

(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.

```

3079 \NewDocumentEnvironment{keyanspic}{o}
3080 {
3081   \__enumext_keyans_pic_safe_exec:
3082   \__enumext_start_list:nn
3083   { }
3084   {
3085     \__enumext_keyans_pic_arg_two:
3086   }

```

We apply the “adjusted” vertical spacing above the environment

```

3087   \vspace { \__enumext_keyans_pic_above_skip }
3088 }

```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `__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.

```

3089 {
3090   \tl_if_novalue:nTF { #1 }
3091   {
3092     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
3093   }
3094   { \__enumext_keyans_pic_do:n { #1 } }
3095   \__enumext_stop_list:
3096   \__enumext_check_starred_cmd:n { anspic }
3097   \setcounter { enumXvi } { 0 }
3098   \vspace { \__enumext_topsep_v_skip }
3099   %\bool_set_false:N \__enumext_store_active_bool
3100 }

```

(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.

```

3101 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3102 {
3103   \int_incr:N \__enumext_keyans_pic_level_int
3104   \int_compare:nNtT { \__enumext_keyans_pic_level_int } > { 1 }
3105   {
3106     \msg_error:nn { enumext } { keyanspic-nested }
3107   }
3108   \__enumext_keyans_save_start_line:
3109 }

```

(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`.

```

3110 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3111 {
3112   \dim_compare:nNnT { #1 } < { 0pt }
3113   { \skip_set:Nn #1 { -#1 } }
3114 }

```

(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.

```

3115 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3116 {

```

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.

```

3117   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3118   \__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.

```

3119   \skip_add:Nn \parsep { \itemsep }
3120   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3121   \dim_zero:N \labelwidth
3122   \dim_zero:N \listparindent
3123   \dim_zero:N \labelsep
3124   \skip_zero:N \partopsep
3125   \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.

```

3126   \__enumext_keyans_pic_skip_abs:N \parsep
3127   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3128   {
3129     \box_dp:N \strutbox
3130     + \l__enumext_topsep_v_skip
3131     - \parsep
3132   }
3133   \__enumext_item_std:w \scan_stop:
3134 }

```

(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`.

```

3135 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3136 {
3137   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3138 }
3139 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

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

```

3140 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3141 {
3142   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3143   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3144   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3145   \int_step_inline:nnn
3146   { \l__enumext_keyans_pic_above_int + 1 }
3147   { \l__enumext_keyans_pic_below_int }
3148   {
3149     \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3150     \centering

```



```

3151         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3152     \__enumext_endminipage:
3153 }
3154 \par
3155 }

```

(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.

```

3156 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3157 {
3158     \dim_compare:nNt { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3159     {
3160         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3161         {
3162             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3163             / \l__enumext_columns_vii_int
3164         }
3165     }
3166     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3167     \dim_set:Nn \l__enumext_item_width_vii_dim
3168     {
3169         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3170         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3171         - \l__enumext_labelsep_vii_dim
3172     }
3173     \dim_zero_new:N \itemwidth
3174 }

```

(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`.

```

3175 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3176 {
3177     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3178     \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3179     {
3180         \msg_warning:nnee { enumext } { item-joined }
3181         { \int_use:N \l__enumext_joined_item_vii_int }
3182         { \int_use:N \l__enumext_columns_vii_int }
3183         \int_set:Nn \l__enumext_joined_item_vii_int
3184         {
3185             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3186         }
3187     }
3188     \int_compare:nNt
3189     { \l__enumext_joined_item_vii_int }
3190     >
3191     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3192     {
3193         \msg_warning:nnee { enumext } { item-joined-columns }
3194         { \int_use:N \l__enumext_joined_item_vii_int }
3195         {
3196             \int_eval:n
3197             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```

```

3198     }
3199     \int_set:Nn \l__enumext_joined_item_vii_int
3200     {
3201         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3202     }
3203 }

```

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

```

3204     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3205     {
3206         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3207         \int_decr:N \l__enumext_joined_item_aux_vii_int
3208         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3209         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3210         \dim_set:Nn \l__enumext_joined_width_vii_dim
3211         {
3212             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3213             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3214               + \l__enumext_columns_sep_vii_dim
3215               ) * \l__enumext_joined_item_aux_vii_int
3216         }
3217         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3218     }
3219     {
3220         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3221         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3222     }
3223 }

```

(End of definition for __enumext_starred_joined_item_vii:n.)

__enumext_start_mini_vii:

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

3224 \cs_new_protected:Nn \__enumext_start_mini_vii:
3225 {
3226     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3227     {
3228         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3229         {
3230             \linewidth
3231             - \l__enumext_minipage_right_vii_dim
3232             - \l__enumext_minipage_hsep_vii_dim
3233         }
3234         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3235         \dim_gset_eq:NN
3236             \g__enumext_minipage_right_vii_dim
3237             \l__enumext_minipage_right_vii_dim
3238         \__enumext_mini_addvspace_vii:
3239         \nointerlineskip\noindent
3240         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3241     }
3242 }

```

(End of definition for __enumext_start_mini_vii:.)

__enumext_stop_mini_vii:

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_env*` environment on the left side, applies `\hfill` and sets the value of the variable `\g__enumext_minipage_active_vii_bool` to true which will be used in the function `__enumext_after_star_env:nn` to execute the `__enumext-mini_env*` on the “right side”.

```

3243 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3244 {
3245     \bool_if:NT \l__enumext_minipage_active_vii_bool
3246     {
3247         \end{__enumext_mini_env*}
3248         \hfill
3249         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3250     }
3251 }

```

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”.

```

3252 \__enumext_after_env:nn {enumext*}
3253 {
3254   \bool_if:NT \g__enumext_minipage_active_vii_bool
3255   {
3256     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3257     \par\addvspace { \g__enumext_minipage_right_skip }
3258     \bool_if:NF \g__enumext_minipage_center_vii_bool
3259     {
3260       \centering
3261     }
3262     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3263     \end{__enumext_mini_env*}
3264     \par\addvspace{ \g__enumext_minipage_after_skip }
3265   }
3266   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3267   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3268   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3269   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3270   \bool_gset_false:N \g__enumext_starred_bool
3271 }

```

(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.

```

3272 \NewDocumentEnvironment{enumext*}{ o }
3273 {
3274   \__enumext_safe_exec_vii:
3275   \__enumext_parse_keys_vii:n {#1}
3276   \__enumext_before_list_vii:
3277   \__enumext_start_store_level_vii:
3278   \__enumext_start_list:nn { }
3279   {
3280     \__enumext_list_arg_two_vii:
3281     \__enumext_before_keys_exec_vii:
3282   }
3283   \__enumext_starred_columns_set_vii:
3284   \item[] \scan_stop:
3285   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3286   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3287 }
3288 {
3289   \__enumext_stop_item_tmp_vii:
3290   \__enumext_remove_extra_parsep_vii:
3291   \__enumext_stop_list:
3292   \__enumext_stop_store_level_vii:
3293   \__enumext_after_list_vii:
3294 }

```

(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`.

```

3295 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3296 {
3297   \__enumext_current_env_set_bool:
3298   \int_incr:N \l__enumext_level_h_int
3299   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3300   {
3301     \msg_error:nn { enumext } { nested }
3302   }
3303   \bool_set_true:N \l__enumext_starred_bool
3304   \__enumext_check_first_level:
3305 }

```

(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.

```

3306 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3307 {
3308     \tl_if_novalue:nF {#1}
3309     {
3310         \str_clear:N \__enumext_series_str
3311         \keys_set:nn { enumext / enumext* } {#1}
3312         \__enumext_parse_series:n {#1}
3313         \bool_if:NT \__enumext_store_active_bool
3314         {
3315             \__enumext_parse_store_keys_vii:n {#1}
3316         }
3317     }
3318 }

```

(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:`.

```

3319 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3320 {
3321     \bool_if:NF \__enumext_store_columns_vii_bool
3322     {
3323         \regex_match:nnT { \b columns\b } {#1}
3324         {
3325             \int_set_eq:NN
3326             \__enumext_store_columns_vii_int
3327             \__enumext_columns_vii_int
3328             \tl_put_right:Ne \__enumext_store_opt_vii_tl
3329             {
3330                 columns = \exp_not:V \__enumext_store_columns_vii_int ,
3331             }
3332         }
3333     }
3334     \bool_if:NF \__enumext_store_columns_sep_vii_bool
3335     {
3336         \regex_match:nnT { \b columns-sep\b } {#1}
3337         {
3338             \dim_set_eq:NN
3339             \__enumext_store_columns_sep_vii_dim
3340             \__enumext_columns_sep_vii_dim
3341             \tl_put_right:Ne \__enumext_store_opt_vii_tl
3342             {
3343                 columns-sep = \exp_not:V \__enumext_store_columns_sep_vii_dim,
3344             }
3345         }
3346     }
3347 }

```

(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`.

```

3348 \cs_new_protected:Nn \__enumext_before_list_vii:
3349 {
3350     \__enumext_vspace_above_vii:
3351     \__enumext_check_ans_exec: % need by chek-ans
3352     \__enumext_before_args_exec_vii:
3353     \__enumext_start_mini_vii:
3354 }

```

(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.

```

3355 \cs_new_protected:Nn \__enumext_after_list_vii:
3356 {
3357   \__enumext_stop_mini_vii:
3358   \__enumext_after_stop_list_vii:
3359   \__enumext_vspace_below_vii:
3360   \bool_set_false:N \l__enumext_starred_bool
3361   \__enumext_resume_save_counter:
3362 }

```

(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:`

```

3363 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3364 {
3365   \bool_if:NT \l__enumext_store_active_bool
3366   {
3367     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3368     {
3369       \__enumext_store_level_open_vii:
3370     }
3371   }
3372 }
3373 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3374 {
3375   \bool_if:NT \l__enumext_store_active_bool
3376   {
3377     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3378     {
3379       \__enumext_store_level_close_vii:
3380     }
3381   }
3382 }

```

(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`.

```

3383 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3384 {
3385   \__enumext_stop_item_tmp_vii:
3386   \int_incr:N \l__enumext_item_column_pos_vii_int
3387   \int_gincr:N \g__enumext_item_count_all_vii_int
3388   \__enumext_item_peek_args_vii:
3389 }

```

(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).

```

3390 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3391 {
3392   \peek_meaning:NTF (
3393     { \__enumext_joined_item_vii:w }
3394     { \__enumext_joined_item_vii:w (1) }
3395   }

```

(End of definition for __enumext_item_peek_args_vii:.)

__enumext_joined_item_vii:w The function __enumext_joined_item_vii:w will first call the function __enumext_starred_joined_item_vii:n in charge of setting the *width* of the box that will store the content passed to \item. Then we will look for the argument “*”, if it is present we will call the function __enumext_starred_item_vii:w otherwise we will call the function __enumext_standard_item_vii:w.

```

3396 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3397 {
3398   \__enumext_starred_joined_item_vii:n {#1}
3399   \peek_meaning_remove:NTF *
3400   { \__enumext_starred_item_vii:w }
3401   { \__enumext_standard_item_vii:w }
3402 }

```

(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].

```

3403 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3404 {
3405   \bool_set_false:N \l__enumext_item_starred_vii_bool
3406   \peek_meaning:NTF [
3407   {
3408     \bool_set_eq:NN
3409     \l__enumext_wrap_label_vii_bool
3410     \l__enumext_wrap_label_opt_vii_bool
3411     \__enumext_start_item_vii:w
3412   }
3413   {
3414     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3415     \legacy_if_set_true:n { @noitemarg }
3416     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3417   }
3418 }

```

(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*⟩].

```

3419 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3420 {
3421   \bool_set_true:N \l__enumext_item_starred_vii_bool
3422   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3423   \peek_meaning:NTF [
3424   { \__enumext_starred_item_vii_aux_i:w }
3425   { \__enumext_starred_item_vii_aux_ii:w }
3426 }
3427 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3428 {
3429   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3430   \__enumext_starred_item_vii_aux_ii:w
3431 }
3432 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3433 {
3434   \peek_meaning:NTF [
3435   { \__enumext_starred_item_vii_aux_iii:w }
3436   {
3437     \dim_set_eq:NN
3438     \l__enumext_item_symbol_sep_vii_dim
3439     \l__enumext_labelsep_vii_dim
3440     \legacy_if_set_true:n { @noitemarg }
3441     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3442   }
3443 }
3444 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]

```

```

3445 {
3446   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3447   \legacy_if_set_true:n { @noitemarg }
3448   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3449 }

```

(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.

```

3450 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3451 {
3452   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3453   \legacy_if:nT { @noitemarg }
3454   {
3455     \legacy_if_set_false:n { @noitemarg }
3456     \legacy_if:nT { @nmbrlist }
3457     {
3458       \bool_if:NT \l__enumext_hyperref_bool
3459       {
3460         \legacy_if_set_true:n { @hyper@item }
3461       }
3462       \refstepcounter{enumXvii}
3463       \bool_if:NT \l__enumext_check_ans_bool
3464       {
3465         \int_gincr:N \g__enumext_count_item_number_int
3466       }
3467     }
3468   }

```

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.

```

3469   \group_begin:
3470   \lrbox{ \l__enumext_item_text_vii_box }
3471   \bool_if:NF \l__enumext_footnotes_key_bool
3472   {
3473     \__enumext_renew_footnote:
3474   }
3475   \bool_if:NT \l__enumext_item_starred_vii_bool
3476   {
3477     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3478     {
3479       \tl_gset_eq:NN
3480       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3481     }
3482     \mode_leave_vertical:
3483     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3484     \makebox[ 0pt ]{ r }{ \g__enumext_item_symbol_aux_vii_tl }
3485     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3486     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3487   }
3488   \group_begin:
3489   \tl_use:N \l__enumext_label_font_style_vii_tl
3490   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3491   {
3492     \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }
3493     { \l__enumext_wrapper_label_vii:n {#1} }
3494   }
3495   {

```

```

3496         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3497     }
3498     \group_end:
3499     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3500     \tl_use:N \l__enumext_after_list_args_vii_tl
3501     \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3502     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3503     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3504     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3505 }

```

(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.

```

3506 \cs_new_protected_nopar:Nn \l__enumext_stop_item_vii:
3507 {
3508     \l__enumext_endminipage:
3509     \endlrbox
3510     \group_end:
3511     \box_set_wd:Nn \l__enumext_item_text_vii_box
3512     {
3513         \l__enumext_joined_width_vii_dim
3514         + \l__enumext_labelwidth_vii_dim
3515         + \l__enumext_labelsep_vii_dim
3516     }
3517     \int_set:Nn \hbadness { 10000 }
3518     \box_use:N \l__enumext_item_text_vii_box
3519     \bool_if:NF \l__enumext_footnotes_key_bool
3520     {
3521         \l__enumext_print_footnote:
3522     }
3523     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3524     {
3525         \par\noindent
3526         \int_zero:N \l__enumext_item_column_pos_vii_int
3527     }
3528     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3529 }

```

(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.

```

3530 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
3531 {
3532     \int_compare:nNnT
3533     {
3534         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3535     }
3536     =
3537     { \c_zero_int }
3538     {
3539         \par
3540         \vspace{ -\l__enumext_itemsep_vii_skip }
3541         \int_gzero:N \g__enumext_item_count_all_vii_int
3542     }
3543 }

```

(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`.

```

3544 \l__enumext_after_env:nn {enumext*}
3545 {
3546     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3547     {
3548         \bool_if:NT \g__enumext_check_ans_show_h_bool
3549         {
3550             \l__enumext_check_ans_show:

```



```

3551     }
3552     \bool_gset_false:N \g__enumext_starred_bool
3553     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3554     \tl_gclear:N \g__enumext_store_name_tl
3555   }
3556 }

```

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.

```

3557 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3558 {
3559   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3560   {
3561     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3562     {
3563       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3564       / \l__enumext_columns_viii_int
3565     }
3566   }
3567   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3568   \dim_set:Nn \l__enumext_item_width_viii_dim
3569   {
3570     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3571     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3572     - \l__enumext_labelsep_viii_dim
3573   }
3574   \dim_zero_new:N \itemwidth
3575 }

```

(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`.

```

3576 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3577 {
3578   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3579   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3580   {
3581     \msg_warning:nnee { enumext } { item-joined }
3582     { \int_use:N \l__enumext_joined_item_viii_int }
3583     { \int_use:N \l__enumext_columns_viii_int }
3584     \int_set:Nn \l__enumext_joined_item_viii_int
3585     {
3586       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3587     }
3588   }
3589   \int_compare:nNnT
3590   { \l__enumext_joined_item_viii_int }
3591   >
3592   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3593   {
3594     \msg_warning:nnee { enumext } { item-joined-columns }
3595     { \int_use:N \l__enumext_joined_item_viii_int }
3596     {
3597       \int_eval:n
3598       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3599     }
3600     \int_set:Nn \l__enumext_joined_item_viii_int
3601     {
3602       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3603     }
3604   }
3605   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3606   {
3607     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3608     \int_decr:N \l__enumext_joined_item_aux_viii_int
3609     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3610     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }

```

```

3611     \dim_set:Nn \l__enumext_joined_width_viii_dim
3612     {
3613         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3614         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3615             + \l__enumext_columns_sep_viii_dim
3616             )*\l__enumext_joined_item_aux_viii_int
3617     }
3618     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3619 }
3620 {
3621     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3622     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3623 }
3624 }

```

(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:`

```

3625 \cs_new_protected:Nn \__enumext_start_mini_viii:
3626 {
3627     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3628     {
3629         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3630         {
3631             \linewidth
3632             - \l__enumext_minipage_right_viii_dim
3633             - \l__enumext_minipage_hsep_viii_dim
3634         }
3635         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3636         \dim_gset_eq:NN
3637             \g__enumext_minipage_right_viii_dim
3638             \l__enumext_minipage_right_viii_dim
3639         \__enumext_mini_addvspace_viii:
3640         \nointerlineskip\noindent
3641         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3642     }
3643 }
3644 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3645 {
3646     \bool_if:NT \l__enumext_minipage_active_viii_bool
3647     {
3648         \end{__enumext_mini_env*}
3649         \hfill
3650         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3651     }
3652 }
3653 \__enumext_after_env:nn {keyans*}
3654 {
3655     \bool_if:NT \g__enumext_minipage_active_viii_bool
3656     {
3657         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3658         \par\addvspace { \g__enumext_minipage_right_skip }
3659         \bool_if:NF \g__enumext_minipage_center_viii_bool
3660         {
3661             \centering
3662         }
3663         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3664         \end{__enumext_mini_env*}
3665         \par\addvspace{ \g__enumext_minipage_after_skip }
3666     }
3667     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3668     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3669     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3670     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3671 }

```

(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.

```

3672 \NewDocumentEnvironment{keyans*}{ o }
3673 {
3674   \__enumext_safe_exec_viii:
3675   \__enumext_parse_keys_viii:n {#1}
3676   \__enumext_before_list_viii:
3677   \__enumext_start_list:nn { }
3678   {
3679     \__enumext_list_arg_two_viii:
3680     \__enumext_before_keys_exec_viii:
3681   }
3682   \__enumext_starred_columns_set_viii:
3683   \item[] \scan_stop:
3684   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3685   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3686 }
3687 {
3688   \__enumext_stop_item_tmp_viii:
3689   \__enumext_remove_extra_parsep_viii:
3690   \__enumext_check_starred_cmd:n { item }
3691   \__enumext_stop_list:
3692   \__enumext_after_list_viii:
3693 }

```

(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.

```

3694 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3695 {
3696   \int_incr:N \__enumext_keyans_level_h_int
3697   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
3698   {
3699     \msg_error:nn { enumext } { nested }
3700   }
3701   \__enumext_keyans_save_start_line:
3702   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3703   \bool_set_false:N \__enumext_store_active_bool
3704   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3705   {
3706     \msg_error:nn { enumext } { keyans-wrong-level }
3707   }
3708 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

3709 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3710 {
3711   \tl_if_novalue:nF {#1}
3712   {
3713     \keys_set:nn { enumext / keyans* } {#1}
3714   }
3715 }

```

(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`.

```

3716 \cs_new_protected:Nn \__enumext_before_list_viii:
3717 {
3718   \__enumext_vspace_above_viii:
3719   \__enumext_before_args_exec_viii:
3720   \__enumext_start_mini_viii:
3721 }

```

(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.

```

3722 \cs_new_protected:Nn \__enumext_after_list_viii:

```

```

3723 {
3724   \__enumext_stop_mini_viii:
3725   \__enumext_after_stop_list_viii:
3726   \__enumext_vspace_below_viii:
3727 }

```

(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`.

```

3728 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3729 {
3730   \__enumext_stop_item_tmp_viii:
3731   \int_incr:N \l__enumext_item_column_pos_viii_int
3732   \int_gincr:N \g__enumext_item_count_all_viii_int
3733   \__enumext_item_peek_args_viii:
3734 }

```

(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)`.

```

3735 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3736 {
3737   \peek_meaning:NTF (
3738     { \__enumext_joined_item_viii:w }
3739     { \__enumext_joined_item_viii:w (1) }
3740 }

```

(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`.

```

3741 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3742 {
3743   \__enumext_starred_joined_item_viii:n {#1}
3744   \peek_meaning_remove:NTF *
3745     { \__enumext_starred_item_viii:w }
3746     { \__enumext_standard_item_viii:w }
3747 }

```

(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]`.

```

3748 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3749 {
3750   \bool_set_false:N \l__enumext_item_starred_viii_bool
3751   \peek_meaning:NTF [

```

```

3752     {
3753         \bool_set_eq:NN
3754         \l__enumext_wrap_label_viii_bool
3755         \l__enumext_wrap_label_opt_viii_bool
3756         \__enumext_start_item_viii:w
3757     }
3758     {
3759         \bool_set_true:N \l__enumext_wrap_label_viii_bool
3760         \legacy_if_set_true:n { @noitemarg }
3761         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3762     }
3763 }

```

(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]`.

```

3764 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3765 {
3766     \bool_set_true:N \l__enumext_item_starred_viii_bool
3767     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3768     \peek_meaning:NTF [
3769     { \__enumext_starred_item_viii_aux_i:w }
3770     { \__enumext_starred_item_viii_aux_ii:w }
3771 }

```

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

```

3772 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3773 {
3774     \tl_clear:N \l__enumext_store_keyans_label_tl
3775     \tl_if_novalue:nF { #1 }
3776     {
3777         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3778         {
3779             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3780             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3781         }
3782         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3783     }
3784     \__enumext_starred_item_viii_aux_ii:w
3785 }
3786 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3787 {
3788     \legacy_if_set_true:n { @noitemarg }
3789     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3790 }

```

(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.

```

3791 \cs_new_protected:Nn \__enumext_starred_item_exec:
3792 {
3793     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3794     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3795     \__enumext_keyans_store_ref:
3796     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3797     \__enumext_keyans_addto_seq_link:
3798     \int_incr:N \l__enumext_check_starred_cmd_int
3799     \bool_if:NT \l__enumext_show_answer_bool
3800     {
3801         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3802     }
3803     \bool_if:NT \l__enumext_show_position_bool
3804     {
3805         \tl_set:Ne \l__enumext_mark_answer_sym_tl

```

```

3806         {
3807             \group_begin:
3808             \exp_not:N \normalfont
3809             \exp_not:N \footnotesize [ \int_eval:n
3810             {
3811                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3812             }
3813             ]
3814             \group_end:
3815         }
3816     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3817 }
3818 }

```

(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.

```

3819 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3820 {
3821     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3822     \legacy_if:nT { @noitemarg }
3823     {
3824         \legacy_if_set_false:n { @noitemarg }
3825         \legacy_if:nT { @nmbrlist }
3826         {
3827             \bool_if:NT \l__enumext_hyperref_bool
3828             {
3829                 \legacy_if_set_true:n { @hyper@item }
3830             }
3831             \refstepcounter{enumXviii}
3832         }
3833     }

```

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

```

3834     \group_begin:
3835     \lrbox{ \l__enumext_item_text_viii_box }
3836     \bool_if:NF \l__enumext_footnotes_key_bool
3837     {
3838         \__enumext_renew_footnote:
3839     }
3840     \bool_if:NT \l__enumext_item_starred_viii_bool
3841     {
3842         \__enumext_starred_item_exec:
3843     }
3844     \group_begin:
3845     \tl_use:N \l__enumext_label_font_style_viii_tl
3846     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3847     {
3848         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3849         { \__enumext_wrapper_label_viii:n {#1} }
3850     }
3851     {
3852         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3853     }
3854     \group_end:
3855     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3856     \tl_use:N \l__enumext_after_list_args_viii_tl
3857     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3858     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3859     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3860     \bool_if:NT \l__enumext_item_starred_viii_bool
3861     {
3862         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3863         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3864     }
3865     {
3866         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3867     }
3868 }

```

(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.

```

3869 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3870 {
3871     \__enumext_endminipage:
3872     \endlrbox
3873     \group_end:
3874     \box_set_wd:Nn \l__enumext_item_text_viii_box
3875     {
3876         \l__enumext_joined_width_viii_dim
3877         + \l__enumext_labelwidth_viii_dim
3878         + \l__enumext_labelsep_viii_dim
3879     }
3880     \int_set:Nn \hbadness { 10000 }
3881     \box_use:N \l__enumext_item_text_viii_box
3882     \bool_if:NF \l__enumext_footnotes_key_bool
3883     {
3884         \__enumext_print_footnote:
3885     }
3886     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3887     {
3888         \par\noindent
3889         \int_zero:N \l__enumext_item_column_pos_viii_int
3890     }
3891     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3892 }

```

(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.

```

3893 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3894 {
3895     \int_compare:nNnT
3896     {
3897         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3898     }
3899     =
3900     { \c_zero_int }
3901     {
3902         \par
3903         \vspace{ -\l__enumext_itemsep_viii_skip }
3904         \int_gzero:N \g__enumext_item_count_all_viii_int
3905     }
3906 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`.)

10.37 The command `\getkeyans`

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form $\langle \textit{store name} : \textit{position} \rangle$. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from *prop list* defined by `save-ans` key.

```

3907 \NewDocumentCommand \getkeyans { m }
3908 {
3909     \exp_args:Ne \__enumext_getkeyans_aux:n
3910     { \tl_to_str:e { \text_expand:n {#1} } }
3911 }

```

(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.

```

3912 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3913 {
3914     \str_if_in:nnTF {#1} { : }
3915     {
3916         \use:e
3917         {
3918             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3919             { {##1} {##2} }

```

```

3920     }
3921     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3922   }
3923   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3924 }

```

(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⟩*.

```

3925 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3926 {
3927   \prop_if_exist:cF { g__enumext_#1_prop }
3928   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3929   \group_begin:
3930     \prop_item:cn { g__enumext_#1_prop }{#2}
3931   \group_end:
3932 }

```

(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.

```

3933 \keys_define:nn { keyanskey / print }
3934 {
3935   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3936                   {
3937                     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3938                   },
3939   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3940   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3941                   {
3942                     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3943                   },
3944   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3945   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3946                   {
3947                     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3948                   },
3949   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3950   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3951                   {
3952                     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3953                   },
3954   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3955   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3956                   {
3957                     \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3958                   },
3959   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3960 }

```

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

```

3961 \NewDocumentCommand \printkeyans { s O{} m }
3962 {
3963   \group_begin:
3964     \tl_use:N \l__enumext_print_keyans_i_tl
3965     \tl_use:N \l__enumext_print_keyans_ii_tl
3966     \tl_use:N \l__enumext_print_keyans_iii_tl
3967     \tl_use:N \l__enumext_print_keyans_iv_tl
3968     \tl_use:N \l__enumext_print_keyans_vii_tl
3969     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3970   \group_end:
3971 }

```

(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

```

3972 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3973 {
3974   \seq_if_exist:cTF { g__enumext_#3_seq }
3975   {
3976     \seq_if_empty:cF { g__enumext_#3_seq }
3977     {
3978       %%\seq_show:c { g__enumext_#3_seq }
3979       \bool_if:nTF {#1}
3980       {
3981         \begin{enumext*}[#2]
3982         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3983         \end{enumext*}
3984       }
3985       {
3986         \begin{enumext}[#2]
3987         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3988         \end{enumext}
3989       }
3990     }
3991   }
3992   {
3993     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3994   }
3995 }

```

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

10.39 The command `\setenumext`

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

```

3996 \keys_define:nn { enumext / meta-families }
3997 {
3998   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3999   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4000   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4001   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4002   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4003   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4004   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4005   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
4006   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
4007   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
4008   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
4009   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
4010   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4011 }

```

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

```

4012 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4013 {
4014   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
4015   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
4016 }

```

`\setenumext`

Now we define the user command `\setenumext`.

```

4017 \NewDocumentCommand \setenumext { o +m }
4018 {
4019   \tl_if_novalue:nTF {#1}
4020   {
4021     \seq_map_inline:Nn \c__enumext_all_families_seq
4022     {
4023       \seq_clear:N \l__enumext_setkey_tmpa_seq
4024       \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4025       \int_set:Nn \l__enumext_setkey_tmpa_int

```

```

4027     {
4028       \seq_count:N \l__enumext_setkey_tmpb_seq
4029     }
4030   \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4031   {
4032     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4033     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4034     \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4035     {
4036       \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4037     }
4038   }
4039   {
4040     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4041   }
4042   \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4043   { \seq_map_inline:Nn \c__enumext_all_families_seq }
4044   { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4045 }
4046 {
4047   \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4048 }
4049 }

```

(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.

```

4050 \cs_new_protected:Npn \__enumext_set_parse:n #1
4051 {
4052   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4053   \int_step_inline:nnn { 0 } { 4 } %<- max level
4054   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4055   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4056   {
4057     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4058     { \tl_trim_spaces:n {#1} }
4059   }
4060   { \__enumext_set_error:nn {#1} { } }
4061 }
4062 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4063 { \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.

```

4064 \msg_new:nnn { enumext } { package-load }
4065 {
4066   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4067 }

4068 \msg_new:nnn { enumext } { package-not-load }
4069 {
4070   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4071 }

4072 \msg_new:nnn { enumext } { package-load-foot }
4073 {
4074   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4075 }

```

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

```

4076 \msg_new:nnn { enumext } { counters }
4077 {
4078   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
4079   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4080 }

```

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

```

4081 \msg_new:nnn { enumext } { invalid-key }
4082 {
4083   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.

```

```

4084     }
4085     \msg_new:nnn { enumext } { unknown-key-family }
4086     {
4087         Unknown~key~family~`\l_keys_key_str'~for~enumext.
4088     }

```

Messages used in length calculation.

```

4089     \msg_new:nnn { enumext } { width-negative }
4090     {
4091         Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4092         The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4093     }
4094     \msg_new:nnn { enumext } { width-zero }
4095     {
4096         Invalid ~ '#1=#2' ~ \msg_line_context:.\
4097         The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4098     }

```

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

```

4099     \msg_new:nnn { enumext } { list-lengths }
4100     {
4101         **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4102         \__enumext_show_length:nnn { dim } { labelsep } {#1}
4103         \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4104         \__enumext_show_length:nnn { dim } { itemindent } {#1}
4105         \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4106         \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4107         \__enumext_show_length:nnn { dim } { listparindent } {#1}
4108         \__enumext_show_length:nnn { skip } { topsep } {#1}
4109         \__enumext_show_length:nnn { skip } { parsep } {#1}
4110         \__enumext_show_length:nnn { skip } { partopsep } {#1}
4111         \__enumext_show_length:nnn { skip } { itemsep } {#1}
4112         ****~
4113     }

```

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

```

4114     \msg_new:nnn { enumext } { list-lengths-not-nested }
4115     {
4116         **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4117         \__enumext_show_length:nnn { dim } { labelsep } {#1}
4118         \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4119         \__enumext_show_length:nnn { dim } { itemindent } {#1}
4120         \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4121         \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4122         \__enumext_show_length:nnn { dim } { listparindent } {#1}
4123         \__enumext_show_length:nnn { skip } { topsep } {#1}
4124         \__enumext_show_length:nnn { skip } { parsep } {#1}
4125         \__enumext_show_length:nnn { skip } { partopsep } {#1}
4126         \__enumext_show_length:nnn { skip } { itemsep } {#1}
4127         ****~
4128     }

```

Messages used by `ref` key.

```

4129     \msg_new:nnn { enumext } { key-ref-empty }
4130     {
4131         Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4132     }

```

Messages used by `save-ans` key.

```

4133     \msg_new:nnn { enumext } { save-ans-empty }
4134     {
4135         Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4136     }
4137
4138     \msg_new:nnn { enumext } { save-ans-ok }
4139     {
4140         Set ~ 'save-ans=#2' ~ in ~ '#1' ~ \msg_line_context:.
4141     }

```

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

```

4142     \msg_new:nnn { enumext } { items-same-answer }
4143     {
4144         *****~Checking~answers~on~'#1'~OK~*****\
4145         **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \

```

```

4146      *****
4147      \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4148    }
4149    \msg_new:nnn { enumext } { item-different-answer }
4150    {
4151      Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4152      answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4153    }

```

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

```

4154    \msg_new:nnn { enumext } { missing-starred }
4155    {
4156      Missing ~ '\c_backslash_str #1*' ~ #2.
4157    }
4158    \msg_new:nnn { enumext } { many-starred }
4159    {
4160      Many ~ '\c_backslash_str #1*' ~ #2.
4161    }

```

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

```

4162    \msg_new:nnn { enumext } { list-too-deep }
4163    {
4164      Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4165      The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4166    }

```

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

```

4167    \msg_new:nnn { enumext } { anskey-wrong-place }
4168    {
4169      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4170      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4171    }
4172    \msg_new:nnn { enumext } { anspic-wrong-place }
4173    {
4174      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4175      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4176    }
4177    \msg_new:nnn { enumext } { command-wrong-place }
4178    {
4179      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4180      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4181    }

```

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

```

4182    \msg_new:nnn { enumext } { keyans-nested }
4183    {
4184      The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4185    }
4186    \msg_new:nnn { enumext } { keyans-wrong-level }
4187    {
4188      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4189      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4190    }
4191    \msg_new:nnn { enumext } { wrong-place }
4192    {
4193      Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4194      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4195    }
4196    \msg_new:nnn { enumext } { keyanspic-nested }
4197    {
4198      The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4199    }
4200    \msg_new:nnn { enumext } { keyanspic-wrong-level }
4201    {
4202      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4203      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4204    }

```

Messages used by `\getkeyans` command.

```

4205    \msg_new:nnn { enumext } { undefined-storage-anskey }
4206    {
4207      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4208    }

```

Messages used by `\miniright` command.

```

4209 \msg_new:nnn { enumext } { missing-miniright }
4210 {
4211   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
4212   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4213 }
4214 \msg_new:nnn { enumext } { wrong-miniright-place }
4215 {
4216   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
4217   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4218 }
4219 \msg_new:nnn { enumext } { wrong-miniright-use }
4220 {
4221   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
4222   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4223 }

```

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

```

4224 \msg_new:nnn { enumext } { nested }
4225 {
4226   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4227 }
4228 \msg_new:nnn { enumext } { item-joined }
4229 {
4230   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4231 }
4232 \msg_new:nnn { enumext } { item-joined-columns }
4233 {
4234   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
4235 }

```

10.41 Finish package

Finish package implementation.

```

4236 \file_input_stop:
4237 </package>

```

11 Index of Implementation

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

Symbols	
<code>*</code>	209
<code>\+</code>	201
<code>\-</code>	201
<code>\\</code> 217, 3060, 4078, 4091, 4096, 4101, 4116, 4144, 4145, 4164, 4169, 4174, 4179, 4188, 4193, 4202, 4211, 4216, 4221	
A	
<code>above</code>	<u>1339</u>
<code>above*</code>	<u>1339</u>
<code>\addvspace</code> . 986, 1014, 1137, 1216, 1279, 1285, 1313, 1330, 2881, 2896, 3016, 3031, 3257, 3264, 3658, 3665	
<code>after</code>	<u>824</u>
<code>align</code>	<u>441</u>
<code>\Alph</code>	32, 36, 37
<code>\Alph</code>	393, 502, 547, 615, 3954
<code>\alph</code>	32, 36, 37
<code>\alph</code>	394, 500, 3944
<code>\anskey</code>	11, 64, <u>2019</u>
<code>\anspic</code>	13, 86, <u>3038</u>
<code>\anspic*</code>	61
<code>\arabic</code>	27, 32
<code>\arabic</code>	392, 499, 546, 3939, 3959
B	
<code>\b</code>	2759, 2772, 3323, 3336
<code>\baselineskip</code>	44
<code>\baselineskip</code>	1979, 1987
<code>before</code>	<u>824</u>
<code>before*</code>	<u>824</u>
<code>below</code>	<u>1339</u>
<code>below*</code>	<u>1339</u>
bool commands:	
<code>\bool_gset_false:N</code> . . 2919, 2920, 3266, 3270, 3552, 3553, 3667	
<code>\bool_gset_true:N</code> 237, 248, 928, 2903, 3249, 3267, 3650, 3668	
<code>\bool_if:NTF</code> . 333, 345, 362, 1361, 1375, 1388, 1399, 1410, 1421, 1432, 1443, 1496, 1513, 1518, 1526, 1553, 1591, 1596, 1603, 1607, 1629, 1634, 1642, 1649, 1680, 1688, 1711, 1715, 1721, 1727, 1798, 1808, 1941, 1965, 1972, 2000, 2031, 2044, 2046, 2057, 2077, 2202, 2213, 2217, 2256, 2271, 2344, 2355, 2359, 2472, 2502, 2576, 2592, 2654, 2664, 2695, 2701, 2749, 2757, 2770, 2815, 2865, 2879, 2887, 2915, 2944, 3001, 3014, 3022, 3040, 3245, 3254, 3258, 3313, 3321, 3334, 3365, 3375, 3458, 3463, 3471, 3475, 3490, 3519, 3548, 3646, 3655, 3659, 3799, 3803, 3827, 3836, 3840, 3846, 3860, 3882	
<code>\bool_if:nTF</code> 1314, 1331, 2085, 2513, 2548, 2612, 3061, 3979	
<code>\bool_if_p:N</code> . 259, 268, 1660, 1661, 1669, 1670, 1777, 2068, 2111, 2112, 2136, 2145, 2146, 2158, 2174, 2330, 2331, 2369, 2370, 2788, 2801, 2803, 2900, 3068, 3069	
<code>\bool_lazy_all:nTF</code> 257, 266, 1775, 2134, 2143, 2156, 2172, 2786, 2799	
<code>\bool_lazy_and:nnTF</code> . . 233, 244, 1659, 1668, 2067, 2110, 2329, 2899	
<code>\bool_lazy_or:nnTF</code>	2368, 3067
<code>\bool_new:N</code> 25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88, 93, 94, 97, 117, 119, 121, 124, 125, 134, 135, 136, 137, 149, 150, 164, 175, 177	
<code>\bool_not_p:n</code> 234, 245, 2069, 2161, 2176, 2789, 2790, 2802, 2901	
<code>\bool_set_eq:NN</code>	2480, 2528, 3408, 3753
<code>\bool_set_false:N</code> 342, 1759, 1760, 2908, 2952, 3034, 3099, 3117, 3360, 3405, 3703, 3750	
<code>\bool_set_true:N</code> 264, 273, 324, 328, 434, 752, 1345, 1350, 1616, 1737, 1738, 1903, 1910, 2476, 2506, 2524, 2536, 2732, 2795, 2808, 2834, 2949, 2976, 3234, 3303, 3414, 3421, 3422, 3635, 3759, 3766, 3767	
box commands:	
<code>\box_dp:N</code> . . 1033, 1037, 1041, 1052, 1056, 1067, 1076, 1082, 1092, 1105, 1111, 1117, 1148, 1149, 1150, 1153, 1163, 1167, 1176, 1183, 1188, 1196, 1225, 1226, 1229, 1236, 1249, 1257, 1263, 1271, 3129	
<code>\box_new:N</code>	58, 170
<code>\box_set_wd:Nn</code>	3511, 3874
<code>\box_use:N</code>	3518, 3881
<code>\box_wd:N</code>	400
C	
<code>\c</code>	209, 210, 652, 654, 666, 668
<code>\cB</code>	210
<code>\cE</code>	210
<code>\centering</code>	1316, 1333, 3150, 3260, 3661
<code>check-ans</code>	<u>1752</u>
Document class:	
<code>article</code>	38
clist commands:	
<code>\clist_const:Nn</code>	182
<code>\clist_map_function:nN</code>	3137
<code>\clist_map_inline:Nn</code> . 440, 694, 757, 823, 838, 919, 1355	
<code>\clist_map_inline:nn</code> 36, 47, 66, 72, 84, 96, 123, 158, 181, 225, 465, 482, 762, 934, 1461, 1705, 1765, 1880, 1898, 1919, 2131, 2265, 2430, 2641, 2644, 2671, 2681, 2684, 2706	
<code>\columnbreak</code>	65
<code>\columnbreak</code>	2071
<code>columns</code>	<u>903</u>
<code>columns*</code>	<u>1899</u>
<code>columns-sep</code>	<u>903</u>
<code>columns-sep*</code>	<u>1899</u>
<code>\columnsep</code>	81, 85
<code>\columnsep</code>	2859, 2998
<code>\columnseprule</code>	81, 85
<code>\columnseprule</code>	2863, 3000
Commands provide by enumext :	
<code>\anskey</code> . 24, 25, 58, 59, 62–69, 71, 80, 93, 103, 104, 108	
<code>\anspic*</code>	24, 25, 61, 68–70, 86–88, 103, 104
<code>\anspic</code>	62, 63, 86–88, 108
<code>\getkeyans</code>	62, 103, 108
<code>\item*</code> . . 24, 25, 61–63, 68–70, 73, 74, 94, 101, 103, 104	
<code>\itemwidth</code>	89, 97
<code>\item</code>	73, 74, 89, 93–95, 97, 100
<code>\miniright</code>	24, 42, 49, 50, 81, 82, 84, 85, 109
<code>\printkeyans</code>	25, 63, 104
<code>\setenumext</code>	25, 105, 106
Counters defined by enumext :	
<code>enumXiii</code>	23, 31

enumXii 23, 31
enumXiv 23, 31
enumXi 23, 31
enumXviii 23, 31
enumXvii 23, 31, 95
enumXvi 23, 31
enumXv 23, 31

cs commands:

\cs_generate_variant:Nn 402, 418, 658, 674, 1924,
1933, 1938, 2018, 2631, 3139
\cs_if_exist:NTF 372
\cs_new:Nn 195
\cs_new:Npn 213, 1462, 1471, 1480
\cs_new_eq:NN 308, 309, 310, 314, 315, 347, 348, 351,
352
\cs_new_protected:Nn . 205, 227, 255, 276, 319, 523,
586, 638, 839, 843, 847, 851, 855, 859, 863, 867, 871,
875, 879, 883, 887, 891, 895, 899, 935, 947, 971, 988,
999, 1023, 1098, 1122, 1139, 1201, 1218, 1240, 1275,
1281, 1356, 1370, 1384, 1395, 1406, 1417, 1428, 1439,
1524, 1627, 1640, 1657, 1678, 1735, 1770, 1806, 1813,
1939, 1963, 1970, 1998, 2005, 2122, 2254, 2269, 2297,
2327, 2364, 2376, 2383, 2435, 2439, 2458, 2509, 2544,
2560, 2570, 2586, 2726, 2784, 2813, 2820, 2843, 2873,
2885, 2942, 2966, 2984, 3009, 3020, 3057, 3101, 3115,
3135, 3140, 3156, 3224, 3243, 3295, 3348, 3355, 3363,
3373, 3390, 3530, 3557, 3625, 3644, 3694, 3716, 3722,
3735, 3791, 3893
\cs_new_protected:Npn 187, 191, 355, 370, 387, 397,
403, 503, 548, 620, 645, 659, 1303, 1322, 1492, 1511,
1581, 1614, 1706, 1826, 1925, 1934, 2054, 2199, 2211,
2233, 2307, 2349, 2468, 2486, 2520, 2532, 2600, 2634,
2674, 2735, 2755, 2962, 3110, 3175, 3306, 3319, 3396,
3403, 3419, 3427, 3432, 3444, 3576, 3709, 3741, 3748,
3764, 3772, 3786, 3912, 3925, 3972, 4050, 4062
\cs_new_protected_nopar:Nn . . . 3383, 3506, 3728,
3869
\cs_new_protected_nopar:Npn 3450, 3819
\cs_set:Nn 2204
\cs_set:Npn 2132, 2170, 3918
\cs_set_eq:NN . . 3285, 3286, 3452, 3684, 3685, 3821
\cs_set_protected:Nn 219, 763, 779, 791, 803
\cs_set_protected:Npn . 32, 41, 59, 67, 79, 85, 113,
154, 162, 221, 419, 441, 470, 483, 530, 675, 695, 739,
758, 815, 824, 903, 920, 1339, 1450, 1697, 1752, 1845,
1881, 1899, 2124, 2258, 2419, 2632, 2672
\cs_to_str:N 389, 412

D

\d 201
\DeclareDocumentEnvironment 1016

dim commands:

\dim_abs:n 2605, 2610
\dim_add:Nn 3120
\dim_compare:nNnTF . 765, 781, 793, 805, 1305, 1324,
2602, 2607, 2613, 2619, 2621, 2623, 2825, 2848, 2970,
2988, 3112, 3158, 3226, 3559, 3627
\dim_compare:nTF 2095
\dim_gset_eq:NN 3235, 3636
\dim_gzero:N 3269, 3670
\dim_new:N . 55, 62, 63, 64, 81, 107, 120, 130, 171, 172,
178
\dim_set:Nn . . 400, 753, 1911, 2500, 2605, 2610, 2612,
2615, 2616, 2620, 2622, 2625, 2626, 2628, 2828, 2851,
2972, 2990, 3142, 3160, 3167, 3210, 3228, 3446, 3561,
3568, 3611, 3629

\dim_set_eq:NN 490, 537, 608, 612, 2495, 2643, 2683,
2774, 2859, 2998, 3217, 3220, 3221, 3338, 3437, 3618,
3621, 3622
\dim_use:N 766, 774, 1306, 1312, 2008, 2011, 2016, 2565,
2567, 2826, 2831, 2832, 2839, 2849, 2853, 2854, 2856
\dim_zero:N 2863, 3000, 3121, 3122, 3123
\dim_zero_new:N 3173, 3574
\c_zero_dim 768, 782, 794, 806, 1306, 1324, 2097, 2602,
2607, 2613, 2620, 2826, 2849, 2970, 2988, 3158, 3226,
3559, 3627

E

\end . . 1309, 1327, 1967, 2002, 2878, 2895, 3013, 3030, 3247,
3263, 3648, 3664, 3983, 3988

\endlist 29
\endlist 309
\endlrbox 3509, 3872
\endminipage 29
\endminipage 315

enumext 5, 2707

enumext internal commands:

\l__enumext__ref_the_count_tl 34
\g__enumext__t__enumext_store_name_tl
__prop 71
\l__enumext__resume_name_tl 54, 55
__enumext_add_pre_parsep: . . 43, 945, 947, 947
__enumext_after_args_exec: . 40, 839, 851, 2719
__enumext_after_args_exec_v: 41, 855, 867, 2935
__enumext_after_args_exec_vii: . . . 871, 895
__enumext_after_args_exec_viii: 899
__enumext_after_env:nn . . 83, 96, 191, 191, 2911,
3252, 3544, 3653
__enumext_after_hyperref: . . . 30, 317, 319, 319
__enumext_after_list: 82, 93, 99, 2724, 2885, 2885
\l__enumext_after_list_args_v_tl 869
\l__enumext_after_list_args_vii_tl 897, 3500
\l__enumext_after_list_args_viii_tl 901, 3856
__enumext_after_list_v: . . 85, 2940, 3020, 3020
__enumext_after_list_vii: . . 3293, 3355, 3355
__enumext_after_list_viii: . . 3692, 3722, 3722
__enumext_after_star_env:nn 90
__enumext_after_stop_list: . . 40, 41, 839, 847,
2906
__enumext_after_stop_list_v: 41, 855, 863, 3035
\l__enumext_after_stop_list_v_tl 865
__enumext_after_stop_list_vii: 871, 887, 3358
\l__enumext_after_stop_list_vii_tl . . . 889
__enumext_after_stop_list_viii: . 891, 3725
\l__enumext_after_stop_list_viii_tl . . . 893
\l__enumext_align_label_vii_str . . 3492, 3496
\l__enumext_align_label_viii_str . 3848, 3852
\l__enumext_align_label_X_str 162
\c__enumext_all_envs_clist . . 182, 440, 694, 757,
823, 838, 919, 1355
\c__enumext_all_families_seq . . 105, 4012, 4021,
4043
__enumext_anskey_wrapper:n 1849, 2209
__enumext_at_begin_document:n . . 29, 187, 187,
306, 312
__enumext_before_args_exec: 40, 839, 839, 2823
__enumext_before_args_exec_v: 40, 41, 855, 855,
2969
__enumext_before_args_exec_vii: . . 871, 871,
3352
__enumext_before_args_exec_viii: 875, 3719


```

\__enumext_before_keys_exec: 40, 839, 843, 2717
\__enumext_before_keys_exec_v: .. 41, 855, 859,
    2933
\__enumext_before_keys_exec_vii ..... 871
\__enumext_before_keys_exec_vii: 41, 879, 3281
\__enumext_before_keys_exec_viii: .. 41, 883,
    3680
\__enumext_before_list: ... 81, 2711, 2820, 2820
\__enumext_before_list_v: . 84, 2928, 2966, 2966
\__enumext_before_list_vii: 92, 3276, 3348, 3348
\__enumext_before_list_viii: .. 99, 3676, 3716,
    3716
\l__enumext_before_no_starred_key_v_tl 861
\l__enumext_before_no_starred_key_vii_-
    tl ..... 881
\l__enumext_before_no_starred_key_viii_-
    tl ..... 885
\l__enumext_before_starred_key_v_tl ... 857
\l__enumext_before_starred_key_vii_tl . 873
\l__enumext_before_starred_key_viii_tl 877
\__enumext_calc_hspace:NNNNNN 76, 2600, 2600,
    2631, 2636, 2676
\l__enumext_check_ans_bool . 73, 134, 1756, 1760,
    1808, 2046, 2344, 2472, 2502, 2900, 3463
\__enumext_check_ans_exec: .. 60, 81, 1806, 1806,
    2824, 3351
\g__enumext_check_ans_item_tl .. 70, 134, 1839,
    1843, 2343
\__enumext_check_ans_set: . 60, 1770, 1770, 1810
\__enumext_check_ans_show: 60, 1813, 1813, 2917,
    3550
\g__enumext_check_ans_show_bool 82, 134, 2903,
    2915, 2920
\g__enumext_check_ans_show_h_bool 134, 3548,
    3553
\__enumext_check_first_level: . 28, 79, 227, 255,
    2733, 3304
\__enumext_check_starred_cmd:n .. 61, 70, 1826,
    1826, 2938, 3096, 3690
\g__enumext_check_starred_cmd_int .... 134
\l__enumext_check_starred_cmd_int 134, 1829,
    1834, 2542, 3066, 3798
\l__enumext_check_start_line_env_tl 139, 282,
    290, 298, 1831, 1836, 1838
\l__enumext_columns_sep_v_dim 2988, 2990, 2998
\l__enumext_columns_sep_vii_dim .. 3158, 3160,
    3169, 3214, 3340, 3528
\l__enumext_columns_sep_viii_dim . 3559, 3561,
    3570, 3615, 3891
\l__enumext_columns_v_int 1144, 2986, 2994, 3006,
    3011
\l__enumext_columns_vii_int .. 3163, 3166, 3170,
    3178, 3182, 3185, 3191, 3197, 3201, 3327, 3523, 3534
\l__enumext_columns_viii_int . 3564, 3567, 3571,
    3579, 3583, 3586, 3592, 3598, 3602, 3886, 3897
\g__enumext_count_item_anskey_int .. 70, 134,
    1816, 1824, 2048, 2346
\g__enumext_count_item_number_int 134, 1781,
    1786, 1789, 1792, 1800, 1816, 1823, 2474, 2504, 3465
\g__enumext_count_item_with_ans_int .... 65
\l__enumext_counter_i_tl ..... 32, 379
\l__enumext_counter_ii_tl ..... 32, 380
\l__enumext_counter_iii_tl ..... 32, 381
\l__enumext_counter_iv_tl ..... 32, 382
\c__enumext_counter_style_tl ..... 27, 37, 207
\g__enumext_counter_styles_tl . 23, 32, 55, 390,
    408
\l__enumext_counter_v_tl ..... 32, 383, 628
\l__enumext_counter_vi_tl ..... 32, 384
\l__enumext_counter_vii_tl ..... 32, 385, 558
\l__enumext_counter_viii_tl ..... 32, 386, 575
\__enumext_current_env_set_bool: . 28, 79, 227,
    227, 2728, 3297
\l__enumext_current_widest_dim 23, 55, 414, 491,
    538, 609, 613
\__enumext_default_item:n ... 2468, 2468, 2517
\__enumext_define_counters:Nn 23, 370, 370, 379,
    380, 381, 382, 383, 384, 385, 386
\__enumext_endminipage: 29, 312, 315, 1022, 3152,
    3508, 3871
\__enumext_fake_item: ..... 763, 763, 2663
\l__enumext_fake_item_indent_v_dim 782, 787
\l__enumext_fake_item_indent_v_tl 784, 2525,
    2529, 2537
\l__enumext_fake_item_indent_vii_dim 794, 799
\l__enumext_fake_item_indent_vii_tl 796, 3504
\l__enumext_fake_item_indent_viii_dim . 806,
    811, 3863
\l__enumext_fake_item_indent_viii_tl .. 808,
    3862, 3866
\l__enumext_fake_item_indent_X_tl ..... 85
\__enumext_fake_item_vii: .... 763, 791, 2693
\__enumext_fake_item_viii: .... 763, 803, 2699
\__enumext_filter_series:n 53, 1462, 1462, 1504,
    1516, 1521
\__enumext_filter_series_key:n 53, 1462, 1467,
    1471
\__enumext_filter_series_pair:nn .. 53, 1462,
    1468, 1480
\g__enumext_footnote_arg_seq . 159, 2441, 2454,
    2464
\g__enumext_footnote_int . 159, 2448, 2451, 2453,
    2455
\g__enumext_footnote_int_seq . 159, 2442, 2455,
    2460, 2463
\__enumext_footnotes_key_bool ..... 30
\l__enumext_footnotes_key_bool 26, 30, 95, 149,
    328, 333, 342, 3471, 3519, 3836, 3882
\__enumext_footnotetext:nn ... 2435, 2435, 2465
\__enumext_getkeyans:nn .. 104, 3921, 3925, 3925
\__enumext_getkeyans_aux:n 103, 3909, 3912, 3912
\l__enumext_hyperref_bool 26, 30, 149, 324, 345,
    362, 2112, 2331, 3458, 3827
\__enumext_hypertarget:nn 30, 319, 347, 351, 367
\__enumext_if_is_int:n ..... 199
\__enumext_if_is_int:nTF ..... 199, 647, 661
\l__enumext_item_column_pos_vii_int 93, 3185,
    3191, 3197, 3201, 3208, 3386, 3523, 3526
\l__enumext_item_column_pos_viii_int .. 100,
    3586, 3592, 3598, 3602, 3609, 3731, 3886, 3889
\l__enumext_item_column_pos_X_int ..... 162
\g__enumext_item_count_all_vii_int 93, 3209,
    3387, 3534, 3541
\g__enumext_item_count_all_viii_int 100, 3610,
    3732, 3897, 3904
\g__enumext_item_count_all_X_int ..... 162
\__enumext_item_peek_args_vii: 93, 3388, 3390,
    3390

```



```

\__enumext_item_peek_args_viii: .. 100, 3733,
    3735, 3735
\__enumext_item_starred: .. 75, 2560, 2560, 2578
\l__enumext_item_starred_vii_bool 3405, 3421,
    3475
\l__enumext_item_starred_viii_bool 3750, 3766,
    3840, 3860
\l__enumext_item_starred_X_bool ..... 162
\__enumext_item_std:w 29, 73, 74, 88, 306, 310, 2477,
    2483, 2507, 2525, 2529, 2537, 3133
\g__enumext_item_symbol_aux_vii_tl 3429, 3477,
    3480, 3484, 3486
\g__enumext_item_symbol_aux_X_tl ..... 162
\l__enumext_item_symbol_sep_vii_dim .. 3438,
    3446, 3483, 3485
\g__enumext_item_symbol_tl 23, 73, 48, 2492, 2566,
    2583
\l__enumext_item_symbol_vii_tl ..... 3480
\l__enumext_item_text_vii_box 3470, 3511, 3518
\l__enumext_item_text_viii_box 3835, 3874, 3881
\l__enumext_item_text_X_box ..... 162
\l__enumext_item_width_vii_dim ... 3167, 3212,
    3220, 3221
\l__enumext_item_width_viii_dim .. 3568, 3613,
    3621, 3622
\l__enumext_item_width_X_dim ..... 162
\l__enumext_itemindent_X_dim ..... 59
\l__enumext_itemsep_vii_skip ..... 3540
\l__enumext_itemsep_viii_skip ..... 3903
\l__enumext_joined_item_aux_vii_int .. 3206,
    3207, 3208, 3209, 3215
\l__enumext_joined_item_aux_viii_int . 3607,
    3608, 3609, 3610, 3616
\l__enumext_joined_item_aux_X_int .... 162
\__enumext_joined_item_vii:w 93, 94, 3393, 3394,
    3396, 3396
\l__enumext_joined_item_vii_int .. 3177, 3178,
    3181, 3183, 3189, 3194, 3199, 3204, 3206, 3212
\__enumext_joined_item_viii:w . 100, 3738, 3739,
    3741, 3741
\l__enumext_joined_item_viii_int . 3578, 3579,
    3582, 3584, 3590, 3595, 3600, 3605, 3607, 3613
\l__enumext_joined_item_X_int ..... 162
\l__enumext_joined_width_vii_dim . 3210, 3217,
    3220, 3501, 3513
\l__enumext_joined_width_viii_dim 3611, 3618,
    3621, 3857, 3876
\l__enumext_joined_width_X_dim ..... 162
\__enumext_keyans_addto_prop:n 68, 2233, 2233,
    2539, 3063
\__enumext_keyans_addto_seq:n . 70, 2307, 2307,
    2541, 3065
\__enumext_keyans_addto_seq_link: 2307, 2325,
    2327, 3797
\__enumext_keyans_anspic_code:nnn . 86, 3054,
    3057, 3057
\__enumext_keyans_default_item:n .. 74, 2520,
    2520, 2556
\l__enumext_keyans_env_bool 20, 2789, 2802, 2949,
    3034
\__enumext_keyans_fake_item: .. 763, 779, 2653
\l__enumext_keyans_item_opt_tl 97, 2353, 2366,
    2372, 3782
\l__enumext_keyans_level_h_int .. 20, 568, 595,
    2285, 3696, 3697
\l__enumext_keyans_level_int .. 20, 1297, 2035,
    2280, 2948, 2953, 3048
\__enumext_keyans_make_label: 32, 76, 2586, 2586,
    2651
\__enumext_keyans_mini_addvspace: 48, 84, 1201,
    1201, 2978
\__enumext_keyans_mini_right_cmd:n 50, 1299,
    1322, 1322
\__enumext_keyans_mini_set_vskip: . 47, 1139,
    1139, 1203
\__enumext_keyans_multi_addvspace: . 85, 988,
    999, 3003
\__enumext_keyans_multi_set_vskip: . 44, 988,
    988, 1001
\__enumext_keyans_multicols_start: 84, 2982,
    2984, 2984
\__enumext_keyans_multicols_stop: . 85, 1326,
    3009, 3009, 3033
\__enumext_keyans_parse_keys:n 2927, 2962, 2962
\l__enumext_keyans_pic_above_int . 129, 3143,
    3144, 3146
\l__enumext_keyans_pic_above_skip .. 88, 129,
    3087, 3127
\__enumext_keyans_pic_arg_two: 88, 3085, 3115,
    3115
\l__enumext_keyans_pic_below_int . 129, 3143,
    3144, 3147
\l__enumext_keyans_pic_body_seq .. 86–88, 129,
    3052, 3092, 3151
\__enumext_keyans_pic_do:n 88, 3092, 3094, 3135,
    3135, 3139
\l__enumext_keyans_pic_level_int .. 20, 1289,
    2039, 2236, 2275, 2310, 2385, 3103, 3104
\__enumext_keyans_pic_row:n 88, 3137, 3140, 3140
\__enumext_keyans_pic_safe_exec: .. 87, 3081,
    3101, 3101
\__enumext_keyans_pic_skip_abs:N .. 88, 3110,
    3110, 3126
\l__enumext_keyans_pic_width_dim . 129, 3142,
    3149
\__enumext_keyans_redefine_item: .. 75, 2544,
    2544, 2650
\__enumext_keyans_ref: ..... 36, 620, 638, 2652
\__enumext_keyans_ref:n ..... 36, 617, 620, 620
\__enumext_keyans_safe_exec: . 2926, 2942, 2942
\__enumext_keyans_save_start_line: . 29, 276,
    276, 2950, 3108, 3701
\__enumext_keyans_show_ans: .. 2349, 2357, 2376
\__enumext_keyans_show_item_opt: . 2349, 2364,
    2537, 3077, 3863
\__enumext_keyans_show_left:n . 74, 2349, 2349,
    2535, 3072
\__enumext_keyans_show_pos: .. 2349, 2361, 2383
\__enumext_keyans_starred_item:n .. 74, 2532,
    2532, 2552
\__enumext_keyans_store_ref: .. 69, 2254, 2254,
    2540, 3064, 3795
\__enumext_keyans_store_ref_aux_i: 69, 2254,
    2266, 2269
\__enumext_keyans_store_ref_aux_ii: 70, 2254,
    2295, 2297
\l__enumext_keyans_tmpa_dim ..... 97
\l__enumext_keyans_tmpa_tl 24, 101, 97, 2534, 2538
\l__enumext_keyans_tmpb_tl ..... 101, 97

```

`__enumext_keyans_wrapper_opt:n` . . 1852, 2372
`\l__enumext_label_copy_i_tl` . . 2166, 2273, 2278, 2283, 2288
`\l__enumext_label_copy_v_tl` 2283
`\l__enumext_label_copy_vi_tl` 2278
`\l__enumext_label_copy_vii_tl` 2141, 2152, 2183, 2273
`\l__enumext_label_copy_viii_tl` 2288
`\l__enumext_label_copy_X_tl` 151
`\l__enumext_label_fill_left_v_tl` 2590
`\l__enumext_label_fill_left_X_tl` 85
`\l__enumext_label_fill_right_v_tl` 2597
`\l__enumext_label_fill_right_X_tl` 85
`\l__enumext_label_font_style_v_tl` 2591, 3076
`\l__enumext_label_font_style_vii_tl` . . 3489
`\l__enumext_label_font_style_viii_tl` . . 3845
`\l__enumext_label_i_tl` 483
`\l__enumext_label_ii_tl` 483
`\l__enumext_label_iii_tl` 483
`\l__enumext_label_iv_tl` 483
`__enumext_label_style:Nnn` 23, 32, 403, 403, 418, 488, 535, 606, 610
`\l__enumext_label_v_tl` . . 68, 70, 603, 2241, 2315, 2378, 2412, 2534, 2538, 2930, 3071, 3073
`\l__enumext_label_vi_tl` . . 68, 70, 603, 2238, 2312, 3071, 3073, 3077
`\l__enumext_label_vii_tl` . . 530, 3416, 3441, 3448
`\l__enumext_label_viii_tl` 530, 3761, 3789, 3793
`\l__enumext_label_width_by_box` . . 55, 399, 400
`__enumext_label_width_by_box:Nn` 32, 397, 397, 402, 414, 671
`\l__enumext_labelsep_i_dim` . . 2380, 2416, 3801, 3816
`\l__enumext_labelsep_v_dim` 2993
`\l__enumext_labelsep_vii_dim` . 3162, 3171, 3213, 3439, 3499, 3515
`\l__enumext_labelsep_viii_dim` 3563, 3572, 3614, 3855, 3878
`\l__enumext_labelwidth_i_dim` . 2380, 2415, 3801, 3816
`\l__enumext_labelwidth_v_dim` 2993
`\l__enumext_labelwidth_vii_dim` . . 3162, 3170, 3213, 3492, 3496, 3514
`\l__enumext_labelwidth_viii_dim` . . 3563, 3571, 3614, 3848, 3852, 3877
`\l__enumext_leftmargin_tmp_v_bool` . . 88, 3117
`\l__enumext_leftmargin_tmp_X_bool` 59
`\l__enumext_leftmargin_tmp_X_dim` 59
`\l__enumext_leftmargin_X_dim` 59
`__enumext_level:` 195, 195, 512, 515, 516, 525, 527, 766, 770, 774, 841, 845, 849, 853, 937, 939, 941, 943, 976, 978, 980, 982, 986, 1026, 1029, 1048, 1057, 1063, 1068, 1072, 1083, 1087, 1088, 1093, 1129, 1133, 1306, 1312, 1359, 1361, 1363, 1366, 1373, 1375, 1377, 1380, 1943, 1951, 1955, 1959, 2204, 2207, 2208, 2476, 2477, 2481, 2482, 2483, 2490, 2492, 2496, 2497, 2500, 2506, 2507, 2562, 2565, 2567, 2574, 2575, 2576, 2579, 2582, 2714, 2716, 2757, 2762, 2763, 2764, 2766, 2770, 2775, 2776, 2777, 2779, 2795, 2808, 2815, 2826, 2828, 2831, 2832, 2834, 2839, 2846, 2849, 2851, 2853, 2854, 2855, 2856, 2859, 2865, 2870, 2876, 2879, 2881, 2887
`\l__enumext_level_h_int` . . 20, 235, 261, 269, 551, 588, 1778, 1795, 2160, 2177, 3298, 3299
`\l__enumext_level_int` . . 79, 20, 197, 246, 260, 270, 949, 1100, 1293, 1772, 2137, 2147, 2153, 2159, 2167, 2175, 2182, 2666, 2729, 2730, 2740, 2747, 2793, 2806, 2861, 2913, 2957, 3044, 3367, 3377, 3546, 3704
`__enumext_list_arg_two_i:` 2632
`__enumext_list_arg_two_ii:` 2632
`__enumext_list_arg_two_iii:` 2632
`__enumext_list_arg_two_iv:` 2632
`__enumext_list_arg_two_v:` . 75, 2632, 2932, 3118
`__enumext_list_arg_two_vii:` 2672, 3280
`__enumext_list_arg_two_viii:` 2672, 3679
`\l__enumext_listoffset_v_dim` 2995
`\l__enumext_listparindent_vii_dim` 3502
`\l__enumext_listparindent_viii_dim` . . . 3858
`__enumext_make_label:` 32, 73, 75, 2570, 2570, 2661
`\l__enumext_mark_answer_sym_tl` . 64, 124, 1858, 2013, 2219, 2387, 2400, 3805
`\l__enumext_mark_position_str` 124, 1862, 1863, 1886, 1887, 2011
`\l__enumext_mark_ref_sym_tl` . . 124, 1872, 2117, 2339
`__enumext_mini_addvspace:` . . 46, 81, 1122, 1122, 2836
`__enumext_mini_addvspace_vii:` 49, 1275, 1275, 3238
`__enumext_mini_addvspace_viii:` 49, 1275, 1281, 3639
`__enumext_mini_env*` 1016
`__enumext_mini_right_cmd:n` 50, 1301, 1303, 1303
`__enumext_mini_set_vskip:` . 45, 1023, 1023, 1124
`__enumext_mini_set_vskip_vii:` 48, 1218, 1218, 1277
`__enumext_mini_set_vskip_viii:` 48, 1218, 1240, 1283
`__enumext_minipage:w` . . 29, 312, 314, 1018, 3149, 3501, 3857
`\l__enumext_minipage_active_v_bool` . . 84, 85, 2976, 3001, 3014, 3022
`\g__enumext_minipage_active_vii_bool` . . . 90, 3249, 3254, 3266
`\l__enumext_minipage_active_vii_bool` . 3234, 3245
`\g__enumext_minipage_active_viii_bool` 3650, 3655, 3667
`\l__enumext_minipage_active_viii_bool` 3635, 3646
`\g__enumext_minipage_active_X_bool` . . . 162
`\l__enumext_minipage_active_X_bool` 73
`\g__enumext_minipage_after_skip` 73, 1222, 1234, 3264, 3665
`\l__enumext_minipage_after_skip` 45, 46, 82, 85, 73, 1039, 1054, 1074, 1090, 1105, 1111, 1117, 1131, 1141, 1150, 1153, 1165, 1183, 1194, 1210, 1242, 1255, 1269, 2896, 3031
`\g__enumext_minipage_center_vii_bool` . 3258, 3267
`\g__enumext_minipage_center_viii_bool` 3659, 3668
`\g__enumext_minipage_center_X_bool` . . . 162
`\l__enumext_minipage_hsep_v_dim` . . . 84, 2974
`\l__enumext_minipage_hsep_vii_dim` . . . 3232
`\l__enumext_minipage_hsep_viii_dim` . . . 3633
`\l__enumext_minipage_left_skip` 45, 84, 73, 1031, 1046, 1065, 1080, 1127, 1137, 1142, 1148, 1157, 1174, 1186, 1206, 1216, 1220, 1225, 1229, 1243, 1247, 1261,

1279, 1285
 \l__enumext_minipage_left_v_dim 84, 2972, 2980
 \l__enumext_minipage_left_vii_dim 3228, 3240
 \l__enumext_minipage_left_viii_dim 3629, 3641
 \l__enumext_minipage_left_X_dim 73
 \g__enumext_minipage_right_skip 73, 1221, 1226, 1230, 3257, 3658
 \l__enumext_minipage_right_skip . 45, 73, 1035, 1050, 1070, 1085, 1143, 1149, 1161, 1179, 1190, 1244, 1251, 1265, 1313, 1330
 \l__enumext_minipage_right_v_dim .. 84, 1324, 1329, 2970, 2974
 \g__enumext_minipage_right_vii_dim 90, 3236, 3256, 3269
 \l__enumext_minipage_right_vii_dim 90, 3226, 3231, 3237
 \g__enumext_minipage_right_viii_dim .. 3637, 3657, 3670
 \l__enumext_minipage_right_viii_dim .. 3627, 3632, 3638
 \g__enumext_minipage_right_X_dim 162
 \g__enumext_minipage_right_X_skip 162
 \g__enumext_minipage_stat_int . 81, 84, 73, 1318, 1335, 2835, 2889, 2894, 2977, 3024, 3029
 \g__enumext_miniright_code_vii_tl . 91, 3262, 3268
 \g__enumext_miniright_code_viii_tl 3663, 3669
 \g__enumext_miniright_code_X_tl 162
 __enumext_multi_addvspace: ... 43, 82, 971, 971, 2867
 __enumext_multi_set_vskip: .. 43, 935, 935, 973
 \l__enumext_multicols_above_ii_skip ... 954
 \l__enumext_multicols_above_iii_skip .. 960
 \l__enumext_multicols_above_iv_skip ... 966
 \l__enumext_multicols_above_v_skip 990, 1004, 1014
 \l__enumext_multicols_above_X_skip 67
 \l__enumext_multicols_below_v_skip 994, 1008, 3016
 \l__enumext_multicols_below_X_skip 67
 __enumext_multicols_start: 81, 2841, 2843, 2843
 __enumext_multicols_stop: 82, 1308, 2873, 2873, 2898
 __enumext_newlabel:nn 26, 30, 68, 355, 355, 2193, 2301
 \l__enumext_newlabel_arg_one_tl 26, 30, 67, 69, 151, 2116, 2186, 2194, 2290, 2302, 2337
 \l__enumext_newlabel_arg_two_tl 26, 30, 66, 151, 2140, 2150, 2164, 2180, 2195, 2277, 2282, 2287, 2303
 __enumext_parse_keys:n ... 54, 2710, 2735, 2735
 __enumext_parse_keys_vii:n 54, 3275, 3306, 3306
 __enumext_parse_keys_viii:n . 3675, 3709, 3709
 __enumext_parse_series:n .. 54, 79, 1492, 1492, 2743, 3312
 __enumext_parse_store_keys:n 79, 80, 2751, 2755, 2755
 __enumext_parse_store_keys_vii:n . 92, 3315, 3319, 3319
 \l__enumext_parsep_i_skip 952, 954, 1103, 1151
 \l__enumext_parsep_ii_skip 958, 960, 1109
 \l__enumext_parsep_iii_skip ... 964, 966, 1115
 \l__enumext_parsep_vii_skip 3503
 \l__enumext_parsep_viii_skip 3859
 \l__enumext_partopsep_v_skip . 1006, 1010, 1177, 1181, 1188, 1192, 1208, 1212
 \l__enumext_partopsep_viii_skip 1253
 __enumext_phantomsection: 30, 319, 348, 352, 368
 __enumext_print_footnote: ... 2435, 2458, 3521, 3884
 __enumext_print_keyans_box:NN 64, 2005, 2005, 2018, 2206, 2380, 2414, 3801, 3816
 \l__enumext_print_keyans_i_tl 3935, 3964
 \l__enumext_print_keyans_ii_tl ... 3940, 3965
 \l__enumext_print_keyans_iii_tl .. 3945, 3966
 \l__enumext_print_keyans_iv_tl ... 3950, 3967
 \l__enumext_print_keyans_vii_tl .. 3955, 3968
 \l__enumext_print_keyans_X_tl 113
 __enumext_printkeyans:nnn 105, 3969, 3972, 3972
 __enumext_redefine_item: . 74, 2509, 2509, 2660
 \l__enumext_ref_key_arg_tl 34, 37, 210, 505, 506, 519, 550, 553, 564, 570, 581, 622, 623, 634
 \l__enumext_ref_the_count_tl . 34, 37, 512, 515, 518, 558, 560, 563, 575, 577, 580, 628, 630, 633
 __enumext_regex_counter_style: .. 27, 34, 205, 205, 513, 559, 576, 629
 __enumext_register_counter_style:Nn .. 387, 387, 392, 393, 394, 395, 396
 __enumext_remove_extra_parsep_vii: .. 3290, 3530, 3530
 __enumext_remove_extra_parsep_viii: . 3689, 3893, 3893
 __enumext_renew_footnote: ... 2435, 2439, 3473, 3838
 \l__enumext_renew_the_count_v_tl 631, 640, 642
 \l__enumext_renew_the_count_vii_tl 561, 590, 592
 \l__enumext_renew_the_count_viii_tl 578, 597, 599
 \l__enumext_renew_the_count_X_tl 37
 \l__enumext_resume_active_bool 54, 56, 48, 1496, 1616
 \l__enumext_resume_bool 23
 __enumext_resume_counter: .. 55, 56, 1614, 1620, 1627
 __enumext_resume_counter:n . 54, 56, 1585, 1590, 1614, 1614, 1684, 1692
 __enumext_resume_counter_save_ans: .. 56, 57, 1614, 1625, 1657
 __enumext_resume_counter_series: 56, 57, 1614, 1623, 1640
 \g__enumext_resume_int . 23, 48, 1537, 1631, 1632
 __enumext_resume_last:n .. 54, 1492, 1498, 1511
 \l__enumext_resume_name_tl 48, 1533, 1541, 1544, 1560, 1568, 1571, 1617, 1618, 1646, 1653
 __enumext_resume_save_counter: 54, 1524, 1524, 2909, 3361
 __enumext_resume_series:n . 55, 1456, 1581, 1581
 __enumext_resume_starred: . 57, 1457, 1678, 1678
 \g__enumext_resume_vii_int .. 93, 48, 1564, 1636, 1637
 __enumext_safe_exec: .. 29, 79, 2709, 2726, 2726
 __enumext_safe_exec_vii: . 29, 3274, 3295, 3295
 __enumext_safe_exec_viii: ... 3674, 3694, 3694
 \l__enumext_series_name_tl 56
 \l__enumext_series_str . 54, 79, 1454, 1494, 1502, 1503, 1505, 1507, 1528, 1531, 1535, 1555, 1558, 1562, 2739, 3310
 __enumext_set_error:nn 4050, 4060, 4062

```

\__enumext_set_parse:n . . . . . 4033, 4050, 4050
\l__enumext_setkey_tmpa_int . . 108, 4026, 4030
\l__enumext_setkey_tmpa_seq . . 108, 4024, 4034,
    4040, 4042, 4044, 4057
\l__enumext_setkey_tmpa_tl . . . 108, 4032, 4036
\l__enumext_setkey_tmpb_seq . . 108, 4025, 4028,
    4032, 4033
\l__enumext_setkey_tmpb_tl 108, 4052, 4054, 4055
\l__enumext_show_answer_bool . 124, 1866, 1890,
    2213, 2355, 2369, 3068, 3799
\__enumext_show_length:nnn . . 39, 213, 213, 4102,
    4103, 4104, 4105, 4106, 4107, 4108, 4109, 4110, 4111,
    4117, 4118, 4119, 4120, 4121, 4122, 4123, 4124, 4125,
    4126
\l__enumext_show_position_bool 124, 1869, 1893,
    2217, 2359, 2370, 3069, 3803
\g__enumext_standar_bool 28, 29, 20, 234, 237, 259,
    1526, 1591, 1603, 1629, 1642, 1680, 1798, 2919
\l__enumext_standar_bool 83, 20, 2145, 2158, 2174,
    2732, 2908
\g__enumext_standar_keyans_pic_star_env-
    int . . . . . 148
\g__enumext_standar_keyans_star_env_int 147
\l__enumext_standar_level_one_bool 28, 79, 20,
    264, 1513, 1660, 1711, 1721
\__enumext_standar_ref: . . . . 34, 503, 523, 2662
\__enumext_standar_ref:n . . . . 34, 495, 503, 503
\g__enumext_standar_series_tl . 48, 1515, 1516,
    1682, 1685
\g__enumext_standar_star_env_int . . 144, 238
\g__enumext_standard_bool . . . . . 79
\l__enumext_standard_bool . . . . . 79
\__enumext_standard_item_vii:w 94, 3401, 3403,
    3403
\__enumext_standard_item_viii:w . . 100, 3746,
    3748, 3748
\g__enumext_starred_bool . 28, 29, 91, 93, 20, 245,
    248, 268, 1553, 1596, 1607, 1634, 1649, 1688, 1777,
    2136, 2146, 2176, 2271, 2790, 2803, 2901, 3270, 3552
\l__enumext_starred_bool . 91, 93, 20, 2069, 2077,
    2161, 2202, 3303, 3360
\__enumext_starred_columns_set_vii: . . 3156,
    3156, 3283
\__enumext_starred_columns_set_viii: . 3557,
    3557, 3682
\__enumext_starred_item:nn . . 2486, 2486, 2515
\__enumext_starred_item_exec: . 101, 3791, 3791,
    3842
\__enumext_starred_item_vii:w . 94, 3400, 3419,
    3419
\__enumext_starred_item_vii_aux_i:w . . 3419,
    3424, 3427
\__enumext_starred_item_vii_aux_ii:w . 3419,
    3425, 3430, 3432
\__enumext_starred_item_vii_aux_iii:w 3419,
    3435, 3444
\__enumext_starred_item_viii:w 100, 101, 3745,
    3764, 3764
\__enumext_starred_item_viii_aux_i:w . 3764,
    3769, 3772
\__enumext_starred_item_viii_aux_ii:w 3764,
    3770, 3784, 3786
\__enumext_starred_joined_item_vii:n . 89, 94,
    3175, 3175, 3398
\__enumext_starred_joined_item_viii:n . . 97,
    100, 3576, 3576, 3743
\g__enumext_starred_keyans_star_env_int 146
\l__enumext_starred_level_one_bool . 20, 273,
    1518, 1669, 1715, 1727
\__enumext_starred_ref: 35, 548, 586, 2690, 2694,
    2700
\__enumext_starred_ref:n . . . . 35, 542, 548, 548
\g__enumext_starred_series_tl . 48, 1520, 1521,
    1690, 1693
\g__enumext_starred_star_env_int . . 145, 249
\__enumext_start_from:NNn 36, 645, 645, 658, 680
\l__enumext_start_i_int . . . . 1632, 1644, 1663
\__enumext_start_item_tmp_vii: 91, 3286, 3383,
    3383
\__enumext_start_item_tmp_viii: 98, 3685, 3728,
    3728
\__enumext_start_item_vii:w . 94, 95, 3411, 3416,
    3441, 3448, 3450, 3450
\__enumext_start_item_viii:w . . 100, 3756, 3761,
    3789, 3819, 3819
\__enumext_start_list:nn 29, 76, 88, 306, 308, 2713,
    2929, 3082, 3278, 3677
\__enumext_start_mini_vii: . 92, 3224, 3224, 3353
\__enumext_start_mini_viii: 99, 3625, 3625, 3720
\__enumext_start_store_level: . 80, 2712, 2784,
    2784
\__enumext_start_store_level_vii: . 93, 3277,
    3363, 3363
\l__enumext_start_vii_int . . . 1637, 1651, 1672
\l__enumext_start_X_int . . . . . 85, 675
\__enumext_stop_item_tmp_vii: . 91, 93, 95, 3285,
    3289, 3385, 3452
\__enumext_stop_item_tmp_viii: . 98, 100, 3684,
    3688, 3730, 3821
\__enumext_stop_item_vii: 95, 96, 3452, 3506, 3506
\__enumext_stop_item_viii: 103, 3821, 3869, 3869
\__enumext_stop_list: . . 29, 306, 309, 2722, 2939,
    3095, 3291, 3691
\__enumext_stop_mini_vii: 90, 93, 3243, 3243, 3357
\__enumext_stop_mini_viii: . 99, 3625, 3644, 3724
\__enumext_stop_store_level: . . 80, 2723, 2784,
    2813
\__enumext_stop_store_level_vii: . . 93, 3292,
    3363, 3373
\l__enumext_store_active_bool 24, 58, 79, 92, 97,
    1661, 1670, 1737, 2031, 2749, 2788, 2801, 2944, 2952,
    3040, 3099, 3313, 3365, 3375, 3703
\__enumext_store_addto_prop:n 62, 68, 1924, 1925,
    1933, 2056, 2252, 3794
\__enumext_store_addto_seq:n 63, 70, 1934, 1934,
    1938, 1945, 1959, 1967, 1976, 1994, 2002, 2120, 2342
\l__enumext_store_ans_bool . 58, 134, 1738, 1759,
    1941, 1965, 1972, 2000, 2044
\l__enumext_store_anskey_arg_tl 24, 65, 66, 97,
    2062, 2071, 2073, 2079, 2087, 2090, 2100, 2105, 2108,
    2114, 2120
\__enumext_store_anskey_code:nnnn . 65, 2050,
    2054, 2054
\__enumext_store_anskey_show_left:n 68, 2061,
    2211, 2211
\__enumext_store_anskey_show_wrap:n 68, 2199,
    2199, 2215, 2230
\l__enumext_store_columns_break_bool . 2025,
    2068

```


`\l__enumext_store_columns_join_int` [97](#), [2076](#), [2081](#)
`\l__enumext_store_columns_sep_vii_bool` [3334](#)
`\l__enumext_store_columns_sep_vii_dim` [3339](#), [3343](#)
`\l__enumext_store_columns_sep_X_bool` [113](#)
`\l__enumext_store_columns_sep_X_dim` [113](#)
`\l__enumext_store_columns_vii_bool` [3321](#)
`\l__enumext_store_columns_vii_int` [3326](#), [3330](#)
`\l__enumext_store_columns_X_bool` [113](#)
`\l__enumext_store_columns_X_int` [113](#)
`__enumext_store_internal_ref:` [65](#), [66](#), [2059](#), [2122](#), [2122](#)
`\l__enumext_store_item_symbol_sep_dim` [2023](#), [2097](#), [2102](#)
`\l__enumext_store_item_symbol_tl` [2021](#), [2088](#), [2092](#)
`\l__enumext_store_keyans_item_opt_sep_tl` [1855](#), [2246](#), [2248](#), [2319](#), [2321](#), [3777](#), [3779](#)
`\l__enumext_store_keyans_item_opt_tl` [97](#)
`\l__enumext_store_keyans_label_tl` [24](#), [68](#), [70](#), [97](#), [2235](#), [2238](#), [2241](#), [2248](#), [2250](#), [2252](#), [2309](#), [2312](#), [2315](#), [2321](#), [2323](#), [2333](#), [2342](#), [2343](#), [3774](#), [3779](#), [3780](#), [3793](#), [3794](#), [3796](#)
`__enumext_store_level_close:` [63](#), [1939](#), [1963](#), [2817](#)
`__enumext_store_level_close_vii:` [1970](#), [1998](#), [3379](#)
`__enumext_store_level_open:` [62](#), [63](#), [80](#), [1939](#), [1939](#), [2796](#), [2809](#)
`__enumext_store_level_open_vii:` [92](#), [1970](#), [1970](#), [3369](#)
`\g__enumext_store_name_tl` [24](#), [82](#), [97](#), [1818](#), [1821](#), [2904](#), [2921](#), [3554](#)
`\l__enumext_store_name_tl` [24](#), [58](#), [97](#), [1547](#), [1550](#), [1574](#), [1577](#), [1665](#), [1674](#), [1708](#), [1709](#), [1724](#), [1730](#), [1739](#), [1741](#), [1743](#), [1745](#), [1747](#), [1749](#), [1927](#), [1929](#), [1936](#), [2188](#), [2189](#), [2225](#), [2292](#), [2293](#), [2393](#), [2406](#), [2904](#), [3811](#)
`\l__enumext_store_opt_vii_tl` [1974](#), [1984](#), [1990](#), [1994](#), [3328](#), [3341](#)
`\l__enumext_store_opt_X_tl` [113](#)
`\l__enumext_store_ref_key_bool` [65](#), [1875](#), [2057](#), [2111](#), [2256](#), [2330](#)
`\l__enumext_store_upper_level_X_bool` [113](#)
`\l__enumext_store_write_aux_file_tl` [26](#), [68](#), [70](#), [151](#), [2191](#), [2197](#), [2299](#), [2305](#)
`__enumext_storing_exec:` [58](#), [1706](#), [1725](#), [1731](#), [1735](#)
`__enumext_storing_set:n` [58](#), [1701](#), [1706](#), [1706](#)
`\l__enumext_the_counter_v_tl` [630](#)
`\l__enumext_the_counter_vii_tl` [560](#)
`\l__enumext_the_counter_viii_tl` [577](#)
`\l__enumext_the_counter_X_tl` [37](#)
`__enumext_tmp:n` [32](#), [36](#), [41](#), [47](#), [59](#), [66](#), [67](#), [72](#), [79](#), [84](#), [85](#), [96](#), [113](#), [123](#), [154](#), [158](#), [162](#), [181](#), [221](#), [225](#), [758](#), [762](#), [1450](#), [1461](#), [1697](#), [1705](#), [1752](#), [1769](#), [1845](#), [1880](#), [1881](#), [1898](#), [2124](#), [2131](#), [2132](#), [2153](#), [2167](#), [2170](#), [2182](#), [2258](#), [2265](#), [2632](#), [2671](#), [2672](#), [2706](#)
`__enumext_tmp:nn` [419](#), [440](#), [441](#), [469](#), [470](#), [482](#), [675](#), [694](#), [739](#), [757](#), [815](#), [823](#), [824](#), [838](#), [903](#), [919](#), [920](#), [934](#), [1339](#), [1355](#), [1899](#), [1923](#), [2419](#), [2434](#)
`__enumext_tmp:nnn` [483](#), [499](#), [500](#), [501](#), [502](#), [530](#), [546](#), [547](#)
`__enumext_tmp:nnnnnn` [695](#), [720](#), [723](#), [726](#), [728](#), [730](#), [733](#), [736](#)
`__enumext_tmp:w` [3918](#), [3921](#)
`\l__enumext_tmpa_vii_int` [3166](#), [3169](#)
`\l__enumext_tmpa_viii_int` [3567](#), [3570](#)
`\l__enumext_tmpa_X_int` [162](#)
`\l__enumext_topsep_v_skip` [992](#), [996](#), [1146](#), [1159](#), [1167](#), [1172](#), [1192](#), [1196](#), [3098](#), [3130](#)
`\l__enumext_topsep_vii_skip` [1223](#), [1232](#), [1236](#)
`\l__enumext_topsep_viii_skip` [1245](#), [1267](#), [1271](#)
`\l__enumext_vspace_a_star_v_bool` [1388](#)
`\l__enumext_vspace_a_star_vii_bool` [1410](#)
`\l__enumext_vspace_a_star_viii_bool` [1421](#)
`\l__enumext_vspace_a_star_X_bool` [85](#)
`__enumext_vspace_above:` [51](#), [1356](#), [1356](#), [2822](#)
`__enumext_vspace_above_v:` [52](#), [1384](#), [1384](#), [2968](#)
`\l__enumext_vspace_above_v_skip` [1386](#), [1390](#), [1392](#)
`__enumext_vspace_above_vii:` [52](#), [1406](#), [1406](#), [3350](#)
`\l__enumext_vspace_above_vii_skip` [1408](#), [1412](#), [1414](#)
`__enumext_vspace_above_viii:` [52](#), [1406](#), [1417](#), [3718](#)
`\l__enumext_vspace_above_viii_skip` [1419](#), [1423](#), [1425](#)
`\l__enumext_vspace_b_star_v_bool` [1399](#)
`\l__enumext_vspace_b_star_vii_bool` [1432](#)
`\l__enumext_vspace_b_star_viii_bool` [1443](#)
`\l__enumext_vspace_b_star_X_bool` [85](#)
`__enumext_vspace_below:` [51](#), [1370](#), [1370](#), [2907](#)
`__enumext_vspace_below_v:` [52](#), [1395](#), [1395](#), [3036](#)
`\l__enumext_vspace_below_v_skip` [1397](#), [1401](#), [1403](#)
`__enumext_vspace_below_vii:` [52](#), [1428](#), [1428](#), [3359](#)
`\l__enumext_vspace_below_vii_skip` [1430](#), [1434](#), [1436](#)
`__enumext_vspace_below_viii:` [52](#), [1428](#), [1439](#), [3726](#)
`\l__enumext_vspace_below_viii_skip` [1441](#), [1445](#), [1447](#)
`__enumext_widest_from:nnnn` [37](#), [659](#), [659](#), [674](#), [686](#)
`\g__enumext_widest_label_tl` [23](#), [32](#), [55](#), [407](#), [411](#), [415](#)
`\l__enumext_wrap_label_opt_v_bool` [2528](#)
`\l__enumext_wrap_label_opt_vii_bool` [94](#), [3410](#)
`\l__enumext_wrap_label_opt_viii_bool` [100](#), [3755](#)
`\l__enumext_wrap_label_opt_X_bool` [85](#)
`\l__enumext_wrap_label_v_bool` [2524](#), [2528](#), [2536](#), [2592](#)
`\l__enumext_wrap_label_vii_bool` [94](#), [3409](#), [3414](#), [3422](#), [3490](#)
`\l__enumext_wrap_label_viii_bool` [100](#), [3754](#), [3759](#), [3767](#), [3846](#)
`\l__enumext_wrap_label_X_bool` [85](#)
`__enumext_wrapper_label_v:n` [2594](#), [3077](#)
`__enumext_wrapper_label_vii:n` [3493](#)
`__enumext_wrapper_label_viii:n` [3849](#)
`__enumext_zero_count_level:` [219](#), [219](#)
`__enumext_zero_parsep:` [46](#), [1043](#), [1098](#), [1098](#)
`enumext*` [5](#), [3272](#)
`enumXi` [379](#)

enumXii 379

enumXiii 379

enumXiv 379

enumXv 379

enumXvi 379

enumXvii 379

enumXviii 379

Environments provide by **enumext**:

enumext* 22, 23, 25-29, 31, 34, 35, 38, 39, 41, 42, 48, 49, 52-55, 57-59, 61-67, 69, 72, 78-80, 91-93, 95, 96, 98, 100, 102, 104, 107, 109

enumext 22, 23, 25, 27-29, 31-40, 42-51, 53-55, 57-65, 67, 69, 72-77, 79, 80, 83, 87, 89, 90, 93, 104, 107, 108

keyans* 22-24, 26, 27, 31, 34-36, 38, 39, 41, 42, 48, 49, 52, 58, 59, 61, 62, 69, 72, 78, 99, 107, 109

keyanspic .. 22-25, 31, 32, 35, 49, 58, 59, 61-63, 68-71, 86-88, 108

keyans .. 22-25, 27, 31, 32, 35, 37-42, 44, 47-52, 58, 59, 61-63, 68-71, 75-77, 83, 84, 86, 88, 90, 100, 107, 108

Environments:

list 27, 29, 76, 79

lrbox 89, 95, 96, 102, 103

minipage 27, 29, 42, 44, 86-89, 95, 96, 103

multicols 42-45, 50, 81, 82, 84, 85

exp commands:

\exp_after:wN 3921

\exp_args:Ne 2746, 3909

\exp_not:N . 45, 410, 518, 563, 580, 633, 772, 786, 787, 798, 799, 810, 811, 2116, 2222, 2223, 2335, 2390, 2391, 2403, 2404, 3808, 3809, 3918

\exp_not:n 284, 292, 300, 518, 519, 563, 564, 580, 581, 633, 634, 773, 1478, 1490, 1907, 1914, 2081, 2092, 2102, 2116, 2117, 2194, 2302, 2337, 2339, 2766, 2779, 3330, 3343

F

\fbox 1850

file commands:

 \file_input_stop: 4236

first 824

font 419

\footnote 72

\footnote 72, 2443

\footnotemark 2453

\footnotesize 2223, 2391, 2404, 3809

\footnotetext 2437

G

\getkeyans 14, 103, 3907

group commands:

 \group_begin: .. 2043, 2221, 2389, 2402, 3469, 3488, 3807, 3834, 3844, 3929, 3963

 \group_end: 2052, 2228, 2396, 2409, 3498, 3510, 3814, 3854, 3873, 3931, 3970

H

\hbadness 3517, 3880

hbox commands:

 \hbox_set:Nn 399

\hfill 449, 453, 458, 459, 1310, 1328, 2116, 2335, 3248, 3649

hook commands:

 \hook_gput_code:nnn 9, 189, 193, 317

 \hook_gset_rule:nnnn 318

\hspace 3528, 3891

\hyperlink 66, 70

\hyperlink 2116, 2335

\hypertarget 30

\hypertarget 347

I

\IfHyperBoolean 325

\IfPackageLoadedTF 11, 321, 335

\ignorespaces 775

\inputlineno 238, 249, 284, 292, 300

int commands:

 \int_add:Nn 3208, 3609

 \int_case:nn 949, 1100, 1772, 1795

 \int_compare:nNnTF . 551, 568, 588, 595, 1025, 1144, 1289, 1293, 1297, 1815, 1828, 1833, 2035, 2039, 2236, 2275, 2280, 2285, 2310, 2385, 2730, 2740, 2793, 2806, 2845, 2861, 2875, 2889, 2913, 2953, 2957, 2986, 3011, 3024, 3044, 3048, 3104, 3178, 3188, 3204, 3299, 3367, 3377, 3523, 3532, 3546, 3579, 3589, 3605, 3697, 3704, 3886, 3895, 4030

 \int_compare_p:nNn ... 235, 246, 260, 261, 269, 270, 1778, 2137, 2147, 2159, 2160, 2175, 2177

 \int_decr:N 3207, 3608

 \int_eval:n 1929, 2189, 2223, 2293, 2391, 2404, 2647, 2689, 3196, 3597, 3809

 \int_from_alph:n 653, 667

 \int_from_roman:n 655, 669

 \int_gadd:Nn 3209, 3610

 \int_gdecr:N 1781, 1786, 1789, 1792, 1800

 \int_gincr:N 1631, 1636, 2048, 2346, 2474, 2504, 2835, 2977, 3387, 3465, 3732

 \int_gset:Nn 238, 249, 2451

 \int_gset_eq:NN 1530, 1537, 1543, 1549, 1557, 1564, 1570, 1576, 2448

 \int_gzero:N 223, 1318, 1335, 1823, 1824, 2894, 3029, 3541, 3904

 \int_if_exist:NNTF 1505, 1541, 1547, 1568, 1574, 1747

 \int_incr:N 2542, 2729, 2948, 3066, 3103, 3298, 3386, 3696, 3731, 3798

 \int_mod:nn 3534, 3897

 \int_new:N . 20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 110, 118, 131, 132, 140, 141, 142, 143, 144, 145, 146, 147, 148, 159, 165, 166, 167, 168, 169, 1507, 1749

 \int_set:Nn 649, 653, 655, 1644, 1651, 1663, 1672, 1904, 2076, 3143, 3144, 3166, 3177, 3183, 3199, 3517, 3567, 3578, 3584, 3600, 3880, 4026

 \int_set_eq:NN . 1632, 1637, 2761, 3206, 3325, 3607

 \int_step_function:nnN 2153, 2167, 2182

 \int_step_inline:nnn 3145, 4053

 \int_to_roman:n 197, 2133, 2171

 \int_use:N . 1026, 1646, 1653, 1665, 1674, 2647, 2666, 2689, 2747, 2846, 2855, 2870, 2876, 3181, 3182, 3194, 3582, 3583, 3595

 \int_zero:N 3526, 3889

 \c_one_int . 3166, 3185, 3191, 3197, 3201, 3204, 3567, 3586, 3592, 3598, 3602, 3605

 \c_zero_int .. 235, 246, 2137, 2147, 2159, 2160, 2175, 2177, 3367, 3377, 3537, 3900

\item 29, 40, 41, 63, 73, 86, 88, 89, 91, 98

\item 73, 74, 93, 95, 100, 102, 310, 1947, 1953, 1978, 1986, 2073, 2312, 2315, 2511, 2546, 3284, 3286, 3683, 3685, 3796

\item* 6, 12, 61, 2544

item-pos* 2419

item-sym* 2419

\itemindent 23, 77

\itemindent 76

itemindent	739
\itemsep	87, 88
\itemsep	3119, 3125
\itemwidth	3173, 3217, 3221, 3574, 3618, 3622
K	
keyans	12, 2924
keyans*	12, 3672
keyanspic	13, 3079
Keys for environments provide by enumext:	
above*	24, 51, 52
above	24, 51, 52, 81, 84, 92, 99
after	40, 41, 83, 86, 93, 99
align	24, 33, 75, 95
before*	40, 41, 81, 92, 99
before	40, 41, 84
below*	24, 51, 52
below	24, 51, 52, 83, 86, 93, 99
check-ans	24, 25, 27, 28, 58, 59, 61, 65, 70, 73, 74, 81-83, 96, 107
columns-sep*	25, 62, 80, 92
columns-sep	42, 63, 80, 81, 85, 92
columns*	25, 62, 80, 92
columns	24, 42, 45, 51, 63, 80, 81, 84, 92
first	40, 41, 95
font	32, 75, 95
item-pos*	64, 66, 72
item-sym*	23, 64, 66, 72, 73
item*-sep	73
itemindent	24, 38, 39, 75, 95
itemsep	37, 78
labelsep	32, 73, 77, 95
labelwidth	31-35, 37, 77
label	23, 31-33, 36, 37, 89
lisparindent	78
list-indent	23, 38, 88
list-offset	38
listparindent	38, 95
mark-ans	25, 61, 68
mark-pos	61, 62
mark-ref	25, 61, 66
mini-env	24, 42, 44, 50, 51, 72, 81, 84, 90, 92, 98, 99
mini-sep	24, 42, 81, 84
miniright*	24, 42
miniright	24, 42, 49, 91
minirigth*	27
minirigth	27
no-store	25, 58-60
noitemsep	37, 46
nosep	37, 46
parindent	78
parsep	37, 78, 95
partopsep	37
ref	23, 27, 33-35, 107
resume*	53, 54, 57, 58, 83
resume	23, 53-58, 83, 93
rightmargin	38
save-ans	24, 53-58, 62, 63, 65, 68, 70, 74, 79, 83, 86, 93, 100, 101, 103, 104, 107
save-key	25, 53, 79
save-ref	26, 30, 61, 65, 66, 69, 70, 75, 101
save-sep	61
series	53-57, 79, 83
show-ans	25, 61, 62, 64, 65, 68, 74, 101
show-length	27, 39, 107

show-pos	25, 61, 62, 64, 65, 68, 71, 74, 101
start	24, 27, 36, 37, 53
store-brk	64, 65
topsep	37
widest	23, 27, 37
wrap-ans	61, 64, 68
wrap-label*	32, 73, 75, 94, 95, 100
wrap-label	32, 75, 94, 95, 100
wrap-opt	61
keys commands:	
\keys_define:nn	421, 443, 472, 485, 532, 603, 677, 697, 741, 760, 817, 826, 905, 922, 1341, 1452, 1699, 1754, 1847, 1883, 1901, 2019, 2421, 3933, 3996
\l_keys_key_str	4087
\keys_set:nn	435, 929, 1346, 1351, 1593, 1598, 1685, 1693, 2065, 2742, 2746, 2964, 3311, 3713, 3998, 3999, 4000, 4001, 4002, 4003, 4004, 4005, 4006, 4007, 4008, 4009, 4047
keyval commands:	
\keyval_parse:NNn	1466
L	
label	483, 530, 603
Labels provide by enumext:	
\Alph*	31, 32
\Roman*	31, 32
\alph*	31, 32
\arabic*	27, 31, 32
\roman*	31, 32
\labelsep	88
\labelsep	3120, 3123
labelsep	419
\labelwidth	32, 88
\labelwidth	3120, 3121
labelwidth	419
\leftmargin	23, 77
\leftmargin	76, 3120
legacy commands:	
\legacy_if:nTF	3453, 3456, 3822, 3825
\legacy_if_gset_false:n	1019
\legacy_if_set_false:n	3455, 3824
\legacy_if_set_true:n	3415, 3440, 3447, 3460, 3760, 3788, 3829
\linewidth	81, 84
\linewidth	2830, 2974, 3142, 3169, 3230, 3570, 3631
\list	29
\list	308
list-indent	739
list-offset	739
\listparindent	3122
listparindent	739
\lrbox	3470, 3835
M	
\makebox	89
\makebox	2009, 2011, 2566, 3484, 3492, 3496, 3848, 3852
\makelabel	73, 75, 76, 89
\makelabel	75, 76, 2572, 2588
\makesavenoteenv	341
mark-ans	1845
mark-pos	1845, 1881
mark-ref	1845
mini-env	903
mini-sep	903
\minipage	29

\minipage 314

\miniright 10, 49, 1287, 2892, 3027

\miniright* 10

mode commands:

 \mode_if_vertical:TF 974, 1002, 1125, 1204

 \mode_leave_vertical: .. 772, 786, 798, 810, 1978, 1986, 2007, 2564, 3482

msg commands:

 \msg_error:nn .. 2955, 2959, 3046, 3106, 3301, 3699, 3706, 4010

 \msg_error:nnn 508, 555, 572, 625, 1291, 1295, 1320, 1337, 1605, 1609, 1713, 1717, 3923, 3928, 3993, 4063

 \msg_error:nnnn 2033, 2037, 2041, 2946, 3042, 3050

 \msg_fatal:nn 2731

 \msg_fatal:nnn 373

 \msg_info:nnn 13, 16, 323, 337

 \msg_line_context: .. 4091, 4096, 4101, 4116, 4131, 4135, 4140, 4152, 4164, 4169, 4174, 4179, 4184, 4188, 4193, 4198, 4202, 4207, 4211, 4216, 4221, 4226, 4230, 4234

 \msg_new:nnn 4064, 4068, 4072, 4076, 4081, 4085, 4089, 4094, 4099, 4114, 4129, 4133, 4138, 4142, 4149, 4154, 4158, 4162, 4167, 4172, 4177, 4182, 4186, 4191, 4196, 4200, 4205, 4209, 4214, 4219, 4224, 4228, 4232

 \msg_note:nnnn 1723, 1729

 \msg_term:nnn 1818

 \msg_term:nnnn 2656, 2666, 2696, 2702

 \msg_warning:nn 2891, 3026

 \msg_warning:nnn 1821

 \msg_warning:nnnn 1831, 1836, 1841, 2604, 2609, 3180, 3193, 3581, 3594

\multicolsep 81, 85

\multicolsep 2860, 2999

N

\NeedsTeXFormat 3

\newcounter 376

\NewDocumentCommand 1287, 2029, 3038, 3907, 3961, 4017

\NewDocumentEnvironment . 2707, 2924, 3079, 3272, 3672

\newlabel 31

\newlabel 359

no-store 1752

\noindent 91, 98

\noindent . 2837, 2979, 3239, 3285, 3525, 3640, 3684, 3888

\nointerlineskip 2837, 2979, 3239, 3640

noitemsep 695

\nopagebreak 985, 1013, 1136, 1215, 1278, 1284

\normalfont 2222, 2390, 2403, 3808

nosep 695

P

Packages:

 enumext 22, 33, 58, 76, 77, 86, 106

 enumitem 31

 expl3 89

 footnotehyper 30

 hyperref 26, 27, 30, 31, 66, 70, 95, 106

 lua-visual-debug 44

 multicol 22, 106

 shortlst 89

\par 985, 1013, 1136, 1215, 1278, 1284, 1313, 1330, 2201, 2881, 2896, 3016, 3031, 3154, 3257, 3264, 3525, 3539, 3658, 3665, 3888, 3902

\parindent 3502, 3858

\parsep 43, 46, 87, 88

\parsep 1979, 1987, 2686, 3119, 3126, 3131

parsep 695

\parskip 3503, 3859

\partopsep 88

\partopsep 2687, 3124

partopsep 695

peek commands:

 \peek_meaning:NTF 3392, 3406, 3423, 3434, 3737, 3751, 3768

 \peek_meaning_remove:NTF 3399, 3744

 \peek_remove_spaces:n 2550

\phantomsection 30

\phantomsection 348

prg commands:

 \prg_do_nothing: 352

 \prg_new_protected_conditional:Npnn ... 199

 \prg_replicate:nn 216, 4147

 \prg_return_false: 203

 \prg_return_true: 202

\printkeyans 14, 104, 3961

prop commands:

 \prop_count:N 1929, 2189, 2225, 2293, 2393, 2406, 3811

 \prop_gput_if_not_in:Nnn 1924, 1927

 \prop_if_exist:NTF 1739, 3927

 \prop_item:Nn 3930

 \prop_new:N 1741

\ProvidesExplPackage 4

R

\raggedcolumns 2869, 3005

\ref 66, 69

ref 483, 530, 603

\refstepcounter 3462, 3831

regex commands:

 \regex_match:nnTF 201, 652, 654, 666, 668, 2759, 2772, 3323, 3336

 \regex_replace_once:nnN 209

\renewcommand 518, 563, 580, 633

\RenewDocumentCommand ... 2443, 2511, 2546, 2572, 2588

\RequirePackage 17

resume 1450

resume* 1450

rightmargin 739

\Roman 32, 36, 37

\Roman 395

\roman 32, 36, 37

\roman 396, 501, 3949

S

save-ans 1697

save-ref 1845

save-sep 1845

scan commands:

 \scan_stop: 88, 3133, 3284, 3683, 3918, 3921

seq commands:

 \seq_clear:N 4024

 \seq_const_from_clist:Nn 4012

 \seq_count:N 3092, 4028

 \seq_gclear:N 2441, 2442

 \seq_gput_right:Nn 1936, 2454, 2455

 \seq_if_empty:NTF 2460, 3976, 4042

 \seq_if_exist:NTF 1743, 3974

 \seq_item:Nn 3151

 \seq_map_function:NN 4033

 \seq_map_inline:Nn .. 3982, 3987, 4021, 4043, 4044

 \seq_map_pairwise_function:NNN 2462

\seq_new:N	111, 112, 129, 160, 161, 1745
\seq_pop_left:NN	4032
\seq_put_right:Nn	3052, 4040, 4057
\seq_set_from_clist:Nn	4025
\seq_set_map_e:NNn	4034
\seq_show:N	3978
series	1450
\setcounter	663, 667, 669, 2647, 2689, 3097
\setenumext	6–9, 105, 3937, 3942, 3947, 3952, 3957, 4017
\setlength	1980, 1988
show-ans	1845, 1881
show-length	815
show-pos	1881
skip commands:	
\skip_add:Nn	954, 960, 966, 976, 980, 1004, 1008, 1105, 1111, 1117, 1127, 1131, 1153, 1206, 1210, 3119
\skip_eval:n	1979, 1987
\skip_gset:Nn	1226, 1230, 1234
\skip_gzero_new:N	1221, 1222
\skip_horizontal:N	787, 799, 811, 3485, 3499, 3855
\skip_horizontal:n	773, 2008, 2016, 2565, 2567, 3483, 3863
\skip_if_eq:nnTF	952, 958, 964, 1028, 1062, 1103, 1109, 1115, 1146, 1151, 1172, 1223, 1245, 1358, 1372, 1386, 1397, 1408, 1419, 1430, 1441
\skip_new:N	69, 70, 74, 75, 76, 77, 78, 133, 179
\skip_set:Nn	937, 941, 990, 994, 1031, 1035, 1039, 1046, 1050, 1054, 1065, 1070, 1074, 1080, 1085, 1090, 1148, 1149, 1150, 1157, 1161, 1165, 1174, 1179, 1183, 1186, 1190, 1194, 1225, 1229, 1247, 1251, 1255, 1261, 1265, 1269, 3113, 3127
\skip_set_eq:NN	2645, 2685, 2686, 3502, 3503, 3858, 3859
\skip_use:N	939, 943, 978, 982, 986, 1006, 1010, 1029, 1048, 1057, 1063, 1068, 1072, 1083, 1087, 1088, 1093, 1129, 1133, 1159, 1359, 1363, 1366, 1373, 1377, 1380, 2881
\skip_zero:N	2687, 2860, 2999, 3124, 3125
\skip_zero_new:N	1141, 1142, 1143, 1220, 1242, 1243, 1244
\c_zero_skip	952, 958, 964, 1029, 1063, 1103, 1109, 1115, 1146, 1151, 1172, 1223, 1245, 1359, 1373, 1386, 1397, 1408, 1419, 1430, 1441
\small	3939, 3944, 3949, 3954, 3959
\star	2425
start	675
\stepcounter	2447, 3059
str commands:	
\c_backslash_str	4156, 4160, 4169, 4170, 4174, 4175, 4179, 4180, 4211, 4212, 4216, 4221, 4222
\c_colon_str	2188, 2292, 3918
\str_case:nn	229, 278
\str_case:nnTF	1473, 1482
\str_clear:N	2739, 3310
\str_count:n	216, 4147
\str_if_empty:NTF	1494, 1535, 1562
\str_if_eq:nnTF	2648, 2691
\str_if_in:nnTF	3914
\str_new:N	128, 174
\str_set:Nn	475, 476, 477, 1862, 1863, 1886, 1887
\string	341
\strutbox	1033, 1037, 1041, 1052, 1056, 1067, 1076, 1082, 1092, 1105, 1111, 1117, 1148, 1149, 1150, 1153, 1163, 1167, 1176, 1183, 1188, 1196, 1225, 1226, 1229, 1236, 1249, 1257, 1263, 1271, 3129

T

TeX and L ^A T _E X 2 _ε commands:	
\@auxout	357
\@currentenv	229, 278
\protected@write	357
text commands:	
\text_expand:n	3910
\textasteriskcentered	1859, 1873
\thepage	363
tl commands:	
\c_space_tl	2372, 4101, 4116
\tl_clear:N	448, 454, 1838, 2062, 2235, 2309, 3774
\tl_clear_new:N	405
\tl_const:Nn	37, 389
\tl_gclear:N	1515, 1520, 1843, 2583, 2921, 3268, 3486, 3554, 3669
\tl_gclear_new:N	1502
\tl_gput_right:Nn	390
\tl_greplace_all:Nnn	411
\tl_gset:Nn	1503, 1516, 1521, 2343, 2904, 3429
\tl_gset_eq:NN	407, 2492, 3479
\tl_if_blank:nTF	3477
\tl_if_empty:NTF	506, 525, 553, 570, 590, 597, 623, 640, 1528, 1533, 1555, 1560, 1618, 1682, 1690, 1709, 1839, 1943, 1974, 2088, 2246, 2319, 2366, 2562, 3777, 4055
\tl_if_empty:nTF	1583
\tl_if_exist:NTF	1588
\tl_if_novalue:nTF	2063, 2074, 2243, 2317, 2351, 2445, 2470, 2488, 2493, 2522, 2737, 3090, 3308, 3711, 3775, 4019
\tl_map_inline:Nn	207, 408
\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, 139, 151, 152, 153, 156, 173, 176
\tl_put_left:Nn	1951, 1984, 2071, 2378, 2412, 3793, 3796
\tl_put_right:Nn	406, 516, 561, 578, 631, 1905, 1912, 1955, 1990, 2073, 2079, 2087, 2090, 2100, 2105, 2108, 2114, 2140, 2150, 2164, 2180, 2186, 2191, 2238, 2241, 2248, 2250, 2277, 2282, 2287, 2290, 2299, 2312, 2315, 2321, 2323, 2333, 2764, 2777, 3328, 3341, 3779, 3780, 3935, 3940, 3945, 3950, 3955
\tl_remove_all:Nn	4054
\tl_remove_once:Nn	2128, 2262
\tl_replace_all:Nnn	410
\tl_reverse:N	2127, 2129, 2261, 2263
\tl_set:Nn	45, 282, 290, 298, 375, 449, 453, 458, 459, 505, 550, 622, 770, 784, 796, 808, 1617, 1708, 2219, 2353, 2387, 2400, 2490, 3782, 3805, 4052
\tl_set_eq:NN	416, 511, 514, 558, 560, 575, 577, 628, 630, 2126, 2260, 2273, 2534, 2538, 3071, 3073
\tl_to_str:n	1588, 1594, 1599, 3910
\tl_trim_spaces:n	406, 4040, 4052, 4058
\tl_use:N	412, 415, 527, 592, 599, 642, 841, 845, 849, 853, 857, 861, 865, 869, 873, 877, 881, 885, 889, 893, 897, 901, 2013, 2133, 2141, 2152, 2166, 2171, 2183, 2477, 2483, 2507, 2525, 2529, 2537, 2574, 2575, 2582, 2590, 2591, 2597, 2714, 2930, 3076, 3262, 3489, 3500, 3504, 3663, 3845, 3856, 3862, 3866, 3964, 3965, 3966, 3967, 3968, 4036
token commands:	
\token_to_str:N	359
\topsep	1980, 1988
topsep	695

<code>\typeout</code>	. 239, 250, 286, 294, 302, 327, 330, 340, 341, 1782, 1801	<code>\vspace</code>	1020, 1363, 1366, 1377, 1380, 1390, 1392, 1401, 1403, 1412, 1414, 1423, 1425, 1434, 1436, 1445, 1447, 1979, 1987, 3087, 3098, 3540, 3903
U			
<code>\u</code> 210		
use commands:			
<code>\use:N</code> 217, 2579, 2716	W	
<code>\use:n</code> 1464, 3916	<code>widest</code> <u>675</u>
<code>\use_none:nn</code> 351	<code>wrap-ans</code> <u>1845</u>
<code>\usecounter</code> 2646, 2688	<code>wrap-label</code> <u>419</u>
V			
<code>\value</code> 1531, 1537, 1544, 1550, 1558, 1564, 1571, 1577	<code>wrap-label*</code> <u>419</u>
		<code>wrap-opt</code> <u>1845</u>