

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-15<sup>\*</sup>

©2024 by Pablo González<sup>†</sup>

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

<b>1</b>	<b>Introduction</b>	<b>3</b>	<b>4</b>	<b>The storage system</b>	<b>10</b>
1.1	Description and usage	4	4.1	Keys for storage	10
1.2	The concept of left margin	4	4.2	Keys for internal label and ref	11
1.3	User interface	4	4.3	Keys for debugging and checking	11
1.3.1	Internal counters	4	4.4	The command <code>\anskey</code>	11
1.3.2	Support for multicol	5	4.5	The environment <code>keyans</code>	11
1.3.3	Support for minipage	5	4.5.1	The <code>\item*</code> in <code>keyans</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	5	4.6	The environment <code>keyanspic</code>	12
1.3.5	Support for <code>\footnote</code>	5	4.6.1	The command <code>\anspic</code>	13
<b>2</b>	<b>The environment <code>enumext</code></b>	<b>5</b>	4.7	Printing stored content	13
2.1	The <code>\item*</code> in <code>enumext</code>	6	4.7.1	The command <code>\getkeyans</code>	13
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	6	4.7.2	The command <code>\printkeyans</code>	13
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>6</b>	<b>5</b>	<b>Full examples</b>	<b>14</b>
3.1	Keys for label and ref	7	<b>6</b>	<b>The way of non-enumerated lists</b>	<b>17</b>
3.2	Keys for spaces	7	<b>7</b>	<b>References</b>	<b>19</b>
3.2.1	Vertical spaces	8	<b>8</b>	<b>Change history</b>	<b>19</b>
3.2.2	Horizontal spaces	8	<b>9</b>	<b>Index of Documentation</b>	<b>20</b>
3.3	Keys for add code	9	<b>10</b>	<b>Implementation</b>	<b>22</b>
3.4	Keys for start and resume	9	<b>11</b>	<b>Index of Implementation</b>	<b>108</b>
3.5	Keys for multicols	9			
3.6	Keys for minipage	9			
3.6.1	The command <code>\miniright</code>	10			
3.6.2	The key <code>miniright</code>	10			

## 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  $\text{\LaTeX}$  team for their great work and to the different members of the `TeX-SX` community who have provided great answers and ideas. Here a note of the main ones:

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

<sup>\*</sup>This file describes a documentation for v1.0, last revised 2024-05-15.

<sup>†</sup>E-mail: [pablgonz@educarchile.cl](mailto: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 (lpp), version 1.3 or later (<https://www.latex-project.org/lppl.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[3] package, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by L<sup>A</sup>T<sub>E</sub>X: `book`, `report`, `article` and `letter` on 10pt, 11pt and 12pt.

1 Introduction

In the  $\text{\LaTeX}$  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)  $\text{\LaTeX}$ 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)  $\text{\LaTeX}$ 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

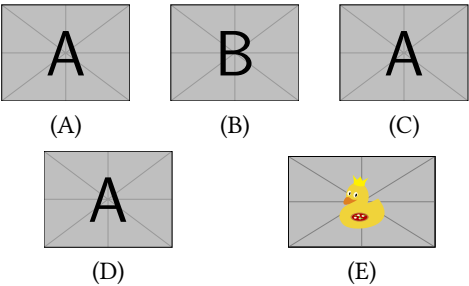
(B) value

(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  $\text{\LaTeX}$ , 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  $\text{\TeX}$  Live and  $\text{\MiKTeX}$ , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktxlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

### 1.3.2 Support for multicols

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

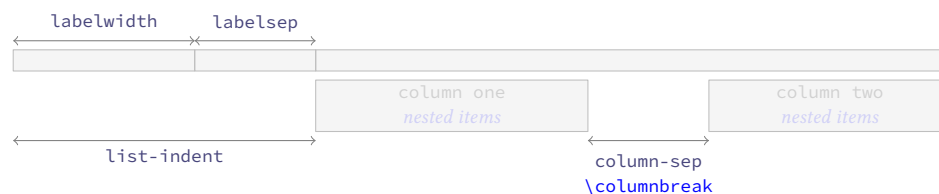


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

### 1.3.3 Support for minipage

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



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

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

### 1.3.4 The \label and \ref system

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

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

### 1.3.5 Support for \footnote

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

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

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

## 2 The environment enumext

```
enumext \begin{enumext} [⟨keyval list⟩]
enumext* \item ⟨item content⟩
          \item [⟨custom⟩] ⟨item content⟩
          \item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩
          \end{enumext}
```

```
\begin{enumext*} [⟨keyval list⟩]
\item ⟨item content⟩
\item [⟨custom⟩] ⟨item content⟩
\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩
\end{enumext*}
```

The `enumext` is an “*enumerated list*” environment that works in the same way as the standard `enumerate` environment provided by L<sup>A</sup>T<sub>E</sub>X, `\item` and `\item[⟨custom⟩]` commands work in the usual way.

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

### Example

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

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

## 2.1 The `\item*` in `enumext`

---

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

---

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

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

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

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

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

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

## 3 The command `\setenumext`

---

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

---

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

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

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

### 3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3.2 Keys for spaces

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

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



### 3.2.1 Vertical spaces

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

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

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

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

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

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

- The value of this parameter also affects the *inner levels* and the `keyans` environment. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T<sub>E</sub>X 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<sub>E</sub>X does *not discard* this space at page break.

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

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

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

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

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

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

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

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

### 3.2.2 Horizontal spaces

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

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

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

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

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

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

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



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

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

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

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

### 3.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

### 3.4 Keys for start and resume

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

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

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

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

### 3.5 Keys for multicol

`columns = {⟨integer⟩}` default: `1`

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

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

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

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

### 3.6 Keys for minipage

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

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

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

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

### 3.6.1 The command `\miniright`

`\miniright` The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”. The *starred version* ‘`*`’ inhibits the use of `\centering` command i.e. the usual L<sup>A</sup>T<sub>E</sub>X justification is maintained in the `minipage` on the “right side”.

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

### 3.6.2 The key `miniright`

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

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

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

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

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

## 4 The storage system

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

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

### 4.1 Keys for storage

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

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

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

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

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

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

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

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

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

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

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

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

## 4.2 Keys for internal label and ref

`save-ref = {⟨true | false⟩}`

default: *false*

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

`mark-ref = {⟨symbol⟩}`

default: *\textasteriskcentered*

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

## 4.3 Keys for debugging and checking

`show-ans = {⟨true | false⟩}`

default: *false*

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

`show-pos = {⟨true | false⟩}`

default: *false*

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

`check-ans = {⟨true | false⟩}`

default: *false*

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

`no-store ⟨value forbidden⟩`

default: *not used*

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

## 4.4 The command \anskey

`\anskey {⟨content⟩}`

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

### Example

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

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

## 4.5 The environment keyans

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

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

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

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

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

4.5.1 The `\item*` in `keyans`

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

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

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

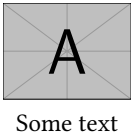
Example

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

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

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

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



Some text

4.6 The environment `keyanspic`

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

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

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

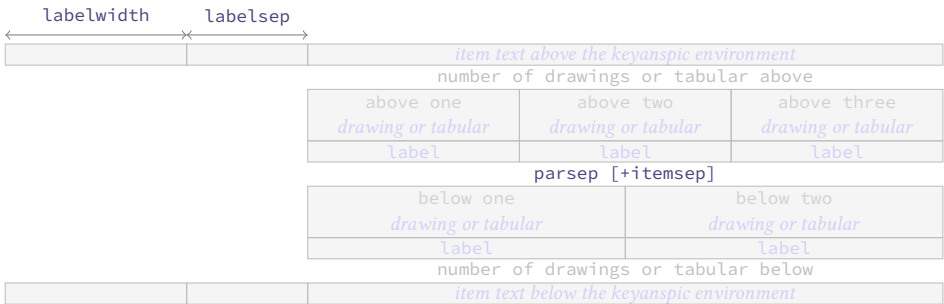


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

4.6.1 The command `\anspic`

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

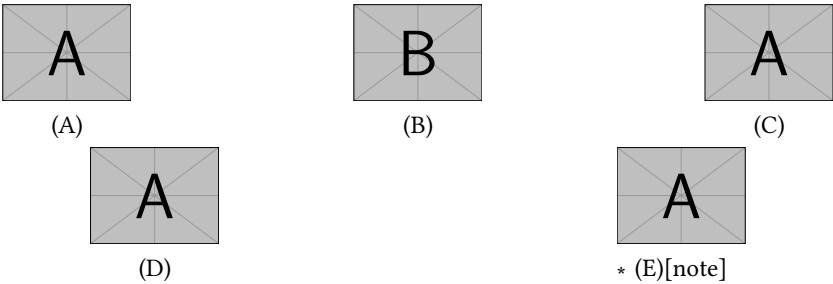
The `\anspic` command take three arguments, the *starred version* ‘`*`’ store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command `\getkeyans`

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

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

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

4.7.2 The command `\printkeyans`

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1] 

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

(a) 

TeXe 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`<sup>1</sup> tool to compile them.

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

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



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

B

C

D

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


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

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

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

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

Example 2

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

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

☒ B

C

D
- $36 \text{ km/h.}$

$360 \text{ km/h.}$

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

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

B

C

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

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

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

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

☒ B

C

D
- $36 \text{ km/h.}$

$360 \text{ km/h.}$

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

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

B

C

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

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

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

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

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

\*

\*

\*

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A value

B correct

C value

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

II.  $\alpha = \delta$

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

A I only

B II only

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

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

A value

B value

C value
4. Question with image and label below:



A



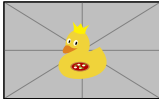
D



B



C



E

5. Question with image on left side:
- A value

B value

C value

D correct

E value



Test keys

1. B,  $x = 5$
2. D
3. C, some note

- \* 4. E, A duck
- \* 5. D, other note
- \*

\*

\*

\*

Example 4

A “simple worksheet” using ducks :) 📄.



Factor  $x^2 - 2x + 1$



Factor  $3x + 3y + 3z$

The following questions need to be cuaqtified :)



True False

- (a)  $\alpha > \delta$
- (b) ~~ETX~~ze is cool?



Related to Linux

- (a) You use linux?
- (b) Usually uses the package manager?
- (c) Rate the following package and class
- i. xsim-exam

ii. xsim

iii. exsheets

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

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

- \* (b) Yes, dnf
- \* (c) i. doesn't exist for now :(
- \* ii. very good
- \* iii. obsolete
- \*

\*

\*

\*

\*

Example 5

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

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<sup>A</sup>T<sub>E</sub>X 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  $\text{\LaTeX}$ , 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  $\text{\LaTeX}$  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. “ $\text{\LaTeX}$  2<sub>ε</sub>: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [7] The  $\text{\LaTeX}$  Project. “Extensive support for hypertext in  $\text{\LaTeX}$ ”. 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  $\text{\LaTeX}$  Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [10] The  $\text{\LaTeX}$  Project. “The  $\text{\LaTeX}$ 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The  $\text{\LaTeX}$  Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.
- [12] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [13] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [14] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

## 8 Change history

**v1.0** 2024-05-15 – First public release.

9 Index of Documentation

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

C

Document class:

article . . . . . 2

book . . . . . 2

exam . . . . . 3

letter . . . . . 2

report . . . . . 2

\columnbreak . . . . . 5

\columnsep . . . . . 9

Commands provide by enumext:

\anskey . . . . . 4, 10, 11

\anspic\* . . . . . 4, 10–13

\anspic . . . . . 10, 12, 13

\getkeyans . . . . . 4, 11, 13

\item\* . . . . . 4–7, 10–12

\item . . . . . 6, 7, 9–11

\miniright . . . . . 4, 5, 10

\printkeyans . . . . . 4, 6, 11, 13

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

Counters defined by enumext:

enumXiii . . . . . 4

enumXii . . . . . 4

enumXiv . . . . . 4

enumXi . . . . . 4

enumXviii . . . . . 4

enumXvii . . . . . 4

enumXvi . . . . . 4

enumXv . . . . . 4

E

Environments provide by enumext:

enumext\* . . . . . 4, 5, 10

enumext . . . . . 4–6, 9–14, 17

keyans\* . . . . . 4, 5, 10

keyanspic . . . . . 4, 7, 10–13, 17

keyans . . . . . 4–12, 17

Environments:

enumerate . . . . . 1, 3, 4, 6, 19

list . . . . . 4, 9, 19

minipage . . . . . 3–5, 9, 10, 19

multicols . . . . . 3, 5, 9

I

\item . . . . . 4, 5

\itemsep . . . . . 8

K

Keys for environments provide by enumext:

above\* . . . . . 8

above . . . . . 8

after . . . . . 9, 10

align . . . . . 7, 18

before\* . . . . . 9

before . . . . . 9

below\* . . . . . 8

below . . . . . 8

check-ans . . . . . 11

columns-sep . . . . . 5, 9

columns . . . . . 5, 8, 9

first . . . . . 9

font . . . . . 7

item-pos\* . . . . . 6

item-sym\* . . . . . 6

itemindent . . . . . 8

itemsep . . . . . 8, 12

labelsep . . . . . 4, 6–10, 18

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

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

list-indent . . . . . 4, 8, 9

list-offset . . . . . 4, 8, 18

listparindent . . . . . 8

mark-ans . . . . . 10

mark-pos . . . . . 10

mark-ref . . . . . 11

mini-env . . . . . 5, 8, 9

mini-sep . . . . . 5, 9

miniright\* . . . . . 10

miniright . . . . . 10

no-store . . . . . 11

noitemsep . . . . . 8

nosep . . . . . 8, 17

parsep . . . . . 8, 12

partopsep . . . . . 8

ref . . . . . 5, 7

resume . . . . . 9

rightmargin . . . . . 8

save-ans . . . . . 5, 9–13

save-ref . . . . . 5, 7, 11, 13

save-sep . . . . . 10

show-ans . . . . . 10, 11

show-length . . . . . 7

show-pos . . . . . 10, 11, 13

start . . . . . 9

topsep . . . . . 8

widest . . . . . 7

wrap-ans . . . . . 10

wrap-label\* . . . . . 7, 18

wrap-label . . . . . 7

wrap-opt . . . . . 10

L

\label . . . . . 5

Labels provide by enumext:

\Alph\* . . . . . 7, 12

\Roman\* . . . . . 7

\alph\* . . . . . 7

\arabic\* . . . . . 7

\roman\* . . . . . 7

\labelsep . . . . . 4, 7

\labelwidth . . . . . 4, 7

\linewidth . . . . . 9

\listparindent . . . . . 8

P

Packages:

enumerate . . . . . 18

enumext . . . . . 1–4, 13, 18, 19

enumitem . . . . . 4, 5, 9, 18, 19

footnotehyper . . . . . 5

hyperref . . . . . 5, 11, 19

l3prop . . . . . 1, 19

l3seq . . . . . 1, 19

©2024 by Pablo González L

20 / 120



multicol .....	1, 2, 5, 19	\ref .....	5
xsim .....	3	\rightmargin .....	8
\parsep .....	8		
\partopsep .....	8		
<b>R</b>		<b>T</b>	
\raggedcolumns .....	5	\topsep .....	8

## 10 Implementation

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

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a T<sub>E</sub>Xpert 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<sup>A</sup>T<sub>E</sub>X3 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<sup>A</sup>T<sub>E</sub>X to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-15}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

### 10.4 Definition of variables

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

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

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

```

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

```

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

```

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

```

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

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

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

```

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

```

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

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\g__enumext_item_symbol_tl
\g__enumext_series_standar_default_tl
\g__enumext_series_starred_default_tl

```

The boolean variable `\l__enumext_resume_bool` is used by `resume` key, the value from which the environment’s will start is stored in the integer variable `\g__enumext_resume_int` (§10.21). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.26).

```

37 \int_new:N \g__enumext_resume_int
38 \int_new:N \g__enumext_resume_vii_int
39 \tl_new:N \g__enumext_item_symbol_tl
40 \tl_new:N \g__enumext_series_standar_default_tl
41 \tl_new:N \g__enumext_series_starred_default_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

42 \dim_new:N \l__enumext_current_widest_dim
43 \tl_new:N \g__enumext_counter_styles_tl
44 \tl_new:N \g__enumext_widest_label_tl
45 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

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

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `\__enumext_calc_hspace:NNNNNNNNNN` (§10.30) which determines the internal values for `\leftmargin` and `\itemindent`.

```

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

```

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

```

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

```

72 \cs_set_protected:Npn \__enumext_tmp:n #1
73 {
74   \bool_new:c { \l__enumext_wrap_label_#1_bool }
75   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
76   \int_new:c { \l__enumext_start_#1_int }
77   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
78   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
79   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
80   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
81   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
82 }
83 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

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

```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§10.21) activates all the mechanism related to `\anskey`, `keyans`, `keyans*` and `keyanspic`.

The variable `\l__enumext_store_name_tl` sets the name for the storage in `⟨sequence⟩` and `⟨prop list⟩`, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the `check-ans` key (§10.21).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.28.2) 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.

```

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

```

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

```

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

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

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

```
100 \cs_set_protected:Npn \l__enumext_tmp:n #1
101 {
102   \tl_new:c { \l__enumext_store_opt_#1_tl }
103   \tl_new:c { \l__enumext_print_keyans_#1_tl }
104   \bool_new:c { \l__enumext_store_columns_#1_bool }
105   \int_new:c { \l__enumext_store_columns_#1_int }
106   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
107   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
108   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
109 }
110 \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` (§10.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.23).

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

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

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

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

Internal variables used by “check answer” mechanism (§10.22) controlled by the `check-ans` and `no-store` keys.

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

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

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

```

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

```

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

`\g__enumext_footnote_int`

Internal variables used for redefinition of `\footnote`.

`\g__enumext_footnote_arg_seq`  
`\g__enumext_footnote_int_seq`

```

143 \int_new:N \g__enumext_footnote_int
144 \seq_new:N \g__enumext_footnote_arg_seq
145 \seq_new:N \g__enumext_footnote_int_seq

```

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

`\c__enumext_counter_style_tl`  
`\l__enumext_ref_key_arg_tl`  
`\l__enumext_ref_aux_tl`  
`\l__enumext_the_counter_X_tl`  
`\l__enumext_counter_style_for_ref_X_tl`

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

```

146 \tl_const:Nn \c__enumext_counter_style_tl
147 { { arabic } { roman } { Roman } { alph } { Alph } }
148 \tl_new:N \l__enumext_ref_key_arg_tl
149 \tl_new:N \l__enumext_ref_aux_tl
150 \cs_set_protected:Npn \__enumext_tmp:n #1
151 {
152   \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
153   \tl_new:c { l__enumext_the_counter_#1_tl }
154   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
155 }
156 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

`\l__enumext_item_starred_X_bool`  
`\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.

```

157 \cs_set_protected:Npn \__enumext_tmp:n #1
158 {
159   \bool_new:c { l__enumext_item_starred_#1_bool }
160   \int_new:c { l__enumext_item_column_pos_#1_int }
161   \int_new:c { g__enumext_item_count_all_#1_int }
162   \int_new:c { l__enumext_joined_item_#1_int }
163   \int_new:c { l__enumext_joined_item_aux_#1_int }
164   \int_new:c { l__enumext_tmpa_#1_int }
165   \box_new:c { l__enumext_item_text_#1_box }
166   \dim_new:c { l__enumext_joined_width_#1_dim }
167   \dim_new:c { l__enumext_item_width_#1_dim }
168   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
169   \str_new:c { l__enumext_align_label_#1_str }
170   \bool_new:c { g__enumext_minipage_active_#1_bool }
171   \tl_new:c { g__enumext_miniright_code_#1_tl }
172   \bool_new:c { g__enumext_minipage_center_#1_bool }
173   \dim_new:c { g__enumext_minipage_right_#1_dim }
174   \skip_new:c { g__enumext_minipage_right_#1_skip }
175 }
176 \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`.

```

177 \clist_const:Nn \c__enumext_all_envs_clist
178 {
179   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
180   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
181 }

```

(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.

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

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

`\__enumext_after_env:nn`

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

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

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
190 \cs_new:Nn \__enumext_level:
191 {
192   \int_to_roman:n { \l__enumext_level_int }
193 }
```

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

`\__enumext_if_is_int:nT`

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

`\__enumext_if_is_int:nF`  
`\__enumext_if_is_int:nTF`

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

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

`\__enumext_show_length:nnn`

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

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

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

`\__enumext_zero_count_level:`

Internal function used by `check-ans` key.

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

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

`\__enumext_current_env:`

The function `\__enumext_current_env:` 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.

```
214 \cs_new_protected:Nn \__enumext_current_env:
215 {
216   \str_case:en { \@currenenv }
217   {
218     {enumext}
```

```

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

```

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

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

```

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

```

(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.

```

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

```

`\__enumext_after_hyperref:` The function `\__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hy  
\__enumext_hypertarget:nn 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  
\__enumext_phantomsection: of the boolean variable \__enumext_footnotes_key_bool to “true”.`

```

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

```

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`.

```

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

```

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

```

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

```

(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.

```

291 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
292 {
293   \protected@write \@auxout { }
294   {
295     \token_to_str:N \newlabel {#1}
296     {
297       {#2}
298       \bool_if:NT \l__enumext_hyref_bool
299       { { \thepage } {#2} {#1} }
300       { }
301     }

```

```

302     }
303     \__enumext_hypertarget:nn {#1} { }
304     \__enumext_phantomsection:
305 }

```

(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 `\__enumext_counter_X_tl` for “store” the counter’s name.  
 #2: The counter’s name.

```

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

```

(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      315 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     316 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    317 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     318 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXvii    319 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXviii   320 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
            321 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
            322 \__enumext_define_counters:Nn \__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.

```

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

```

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

```

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

```

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

```

377 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
378 {
379   \keys_define:nn { enumext / #1 }
380   {
381     align .choice:,
382     align / left .code:n =
383     {
384       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
385       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
386     },
387     align / right .code:n =
388     {
389       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
390       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
391     }
392   }
393 }

```

```

391         },
392         align / center .code:n =
393         {
394             \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
395             \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
396         },
397         align .initial:n = left,
398         align .value_required:n = true,
399     }
400 }
401 \clist_map_inline:nn
402 {
403     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
404 }
405 { \__enumext_tmp:nn #1 }

```

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

```

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

```

(End of definition for `align`.)

## 10.11 Setting label and ref keys

`\__enumext_regex_label_ref_key:`

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

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

```

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

```

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

`\__enumext_set_label_ref:n`

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

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

```

427 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
428 {
429     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
430     \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
431     \__enumext_regex_label_ref_key:
432     \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
433     \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
434     {
435         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
436         { \exp_not:V \l__enumext_ref_key_arg_tl }
437     }
438 }

```

(End of definition for `\__enumext_set_label_ref:n`.)



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

```

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

```

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

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

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

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

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

```

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

```

(End of definition for `\__enumext_set_label_ref_h:n`.)

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

```

471 \cs_new_protected:Nn \__enumext_use_key_ref_h:
472 {
473   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
474   {
475     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
476     {
477       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
478     }
479   }
480   {
481     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
482     {
483       \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
484     }
485   }
486 }

```

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

### 10.11.1 Define and set label key for enumext environment

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

```

\l__enumext_label_i_tl 487 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 488 {
\l__enumext_label_iii_tl 489   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 490   {
491     label .code:n = {
492       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
493       { l__enumext_counter_#2_tl } {##1}
494       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
495       \l__enumext_current_widest_dim
496     },
497     label .initial:n = #3,
498     label .value_required:n = true,
499     ref .code:n = \__enumext_set_label_ref:n {##1},
500     ref .value_required:n = true,
501   }
502 }
503 \__enumext_tmp:nnn { level-1 } { i } { \arabic*.}
504 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
505 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
506 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

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

```

\l__enumext_label_vii_tl 507 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 508 {
509   \keys_define:nn { enumext / #1 }
510   {
511     label .code:n = {
512       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
513       { l__enumext_counter_#2_tl } {##1}
514       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
515       \l__enumext_current_widest_dim
516     },
517     label .initial:n = #3,
518     label .value_required:n = true,
519     ref .code:n = \__enumext_set_label_ref_h:n {##1},
520     ref .value_required:n = true,
521   }
522 }
523 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
524 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

### 10.11.3 Define and set label key for keyans and keyanspic environment

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

\l\_\_enumext\_label\_v\_tl Define and set `label` key for `keyans` environment.

```

\l__enumext_label_vi_tl 525 \keys_define:nn { enumext / keyans }
526 {
527   label .code:n = {
528     \__enumext_label_style:cvn { l__enumext_label_v_tl }
529     { l__enumext_counter_v_tl } {#1}
530     \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
531     \l__enumext_current_widest_dim
532     \__enumext_label_style:cvn { l__enumext_label_vi_tl }
533     { l__enumext_counter_vi_tl } {#1}
534     \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
535     \l__enumext_current_widest_dim
536   },
537   label .initial:n = (\Alph*),
538   label .value_required:n = true,
539 }

```

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

## 10.12 Setting start and widest keys

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

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

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

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

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

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

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

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

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

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

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

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

```
start
widest
\l__enumext_start_X_int
```

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

```
570 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
571 {
572   \keys_define:nn { enumext / #1 }
573   {
574     start .code:n = {
575       \__enumext_start_from:ccn
576       { \l__enumext_label_#2_tl }
577       { \l__enumext_start_#2_int } {##1}
578     },
579     start .initial:n = 1,
580     widest .code:n = {
581       \__enumext_widest_from:nccn {#2}
582       { \l__enumext_label_#2_tl }
583       { \l__enumext_labelwidth_#2_dim } {##1}
584     },
585   }
```

```

585         widest .value_required:n = true,
586         start .value_required:n = true,
587     }
588 }
589 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for start, widest, and \\_\_enumext\_start\_X\_int.)

### 10.13 Setting keys for vertical spaces

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

```

590 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
591 {
592     \keys_define:nn { enumext / #1 }
593     {
594         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
595         topsep .initial:n = {#3},
596         topsep .value_required:n = true,
597         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
598         partopsep .initial:n = {#4},
599         partopsep .value_required:n = true,
600         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
601         parsep .initial:n = {#5},
602         parsep .value_required:n = true,
603         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
604         itemsep .initial:n = {#6},
605         itemsep .value_required:n = true,
606         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
607         noitemsep .value_forbidden:n = true,
608         nosep .meta:n = {
609             itemsep = 0pt, parsep = 0pt,
610             topsep = 0pt, partopsep = 0pt,
611         },
612         nosep .value_forbidden:n = true,
613     }
614 }

```

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

```

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

```

(End of definition for topsep and others.)

### 10.14 Setting keys for horizontal spaces

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

```

634 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
635 {
636     \keys_define:nn { enumext / #1 }
637     {
638         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
639         itemindent .value_required:n = true,
640         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },

```

```

641     rightmargin .value_required:n = true,
642     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
643     listparindent .value_required:n = true,
644     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
645     list-offset .value_required:n = true,
646     list-indent .code:n =
647         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
648         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
649     list-indent .value_required:n = true,
650 }
651 }
652 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

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

```

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

```

#### 10.14.1 Functions for setting the fake *itemindent*

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

```

658 \cs_set_protected:Nn \__enumext_fake_item:
659 {
660     \dim_compare:nNnT
661     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
662     >
663     { \c_zero_dim }
664     {
665         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
666         {
667             \exp_not:N \mode_leave_vertical:
668             \exp_not:n { \skip_horizontal:n }
669             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
670             \ignorespaces
671         }
672     }
673 }
674 \cs_set_protected:Nn \__enumext_keyans_fake_item:
675 {
676     \dim_compare:nNnT
677     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
678     {
679         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
680         {
681             \exp_not:N \mode_leave_vertical:
682             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
683         }
684     }
685 }
686 \cs_set_protected:Nn \__enumext_fake_item_vii:
687 {
688     \dim_compare:nNnT
689     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
690     {
691         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
692         {
693             \exp_not:N \mode_leave_vertical:
694             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
695         }
696     }
697 }
698 \cs_set_protected:Nn \__enumext_fake_item_viii:
699 {
700     \dim_compare:nNnT

```

```

701     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
702     {
703         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
704         {
705             \exp_not:N \mode_leave_vertical:
706             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
707         }
708     }
709 }

```

(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.

```

710 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
711 {
712     \keys_define:nn { enumext / #1 }
713     {
714         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
715         show-length .initial:n = false,
716     }
717 }
718 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 10.16 Setting before, after and first keys

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

```

before*
after
first
719 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
720 {
721     \keys_define:nn { enumext / #1 }
722     {
723         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
724         before .value_required:n = true,
725         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
726         before* .value_required:n = true,
727         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
728         after .value_required:n = true,
729         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
730         first .value_required:n = true,
731     }
732 }
733 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

#### 10.16.1 Functions for before, after and first keys in enumext

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

```

734 \cs_new_protected:Nn \__enumext_before_args_exec:
735 {
736     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
737 }

```

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

```

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

```

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

```

742 \cs_new_protected:Nn \__enumext_after_stop_list:
743 {

```

```

744     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
745   }

```

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

```

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

```

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

### 10.16.2 Functions for before, after and first keys in keyans

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

```

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

```

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⟩`.

```

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

```

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

```

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

```

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`.

```

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

```

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

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

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

```

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

```

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⟩`.

```

774 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
775 {
776     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
777 }
778 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
779 {
780     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
781 }

```



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

```

782 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
783 {
784   \tl_use:N \l__enumext_after_stop_list_vii_tl
785 }
786 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
787 {
788   \tl_use:N \l__enumext_after_stop_list_viii_tl
789 }

```

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`.

```

790 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
791 {
792   \tl_use:N \l__enumext_after_list_args_vii_tl
793 }
794 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
795 {
796   \tl_use:N \l__enumext_after_list_args_viii_tl
797 }

```

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

### 10.17 Setting keys for multicols and minipage

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.

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

```

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

```

815 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
816 {
817   \keys_define:nn { enumext / #1 }
818   {
819     columns .initial:n = 2,
820     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
821     miniright .value_required:n = true,
822     miniright* .code:n = {
823       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
824       \keys_set:nn { enumext / #1 } { miniright = {##1} }
825     },
826     miniright* .value_required:n = true,
827   }
828 }
829 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

## 10.18 Adjustment of vertical spaces for multicol

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



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

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

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

### 10.18.1 Adjustment of vertical spaces for multicol in enumext

`\__enumext_multi_set_vskip:`

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

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

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

```

830 \cs_new_protected:Nn \__enumext_multi_set_vskip:
831 {
832   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip }
833   {
834     \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip }
835   }
836   \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip }
837   {
838     \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip }
839   }
840   \__enumext_add_pre_parsep:
841 }
```

(End of definition for `\__enumext_multi_set_vskip:`)

`\__enumext_add_pre_parsep:`

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

```

842 \cs_new_protected:Nn \__enumext_add_pre_parsep:
843 {
844   \int_case:nn { \__enumext_level_int }
845   {
846     { 2 }{
847       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
848       {
849         \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
850       }
851     }
852     { 3 }{
853       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip }
854       {
855         \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
856       }
857     }
858     { 4 }{
859       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip }
860       {
861         \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
862       }
863     }
864   }
```

```

864     }
865 }

```

(End of definition for `\__enumext_add_pre_parse:`)

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

```

866 \cs_new_protected:Nn \__enumext_multi_addvspace:
867 {
868   \__enumext_multi_set_vskip:
869   \mode_if_vertical:T
870   {
871     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
872     {
873       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
874     }
875     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
876     {
877       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
878     }
879   }
880   \par\nopagebreak
881   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
882 }

```

(End of definition for `\__enumext_multi_addvspace:`)

### 10.18.2 Adjustment of vertical spaces for multicols in keyans

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

`\__enumext_keyans_multi_addvspace:`

```

883 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
884 {
885   \skip_set:Nn \__enumext_multicols_above_v_skip
886   {
887     \__enumext_topsep_v_skip
888   }
889   \skip_set:Nn \__enumext_multicols_below_v_skip
890   {
891     \__enumext_topsep_v_skip
892   }
893 }
894 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
895 {
896   \__enumext_keyans_multi_set_vskip:
897   \mode_if_vertical:T
898   {
899     \skip_add:Nn \__enumext_multicols_above_v_skip
900     {
901       \skip_use:N \__enumext_partopsep_v_skip
902     }
903     \skip_add:Nn \__enumext_multicols_below_v_skip
904     {
905       \skip_use:N \__enumext_partopsep_v_skip
906     }
907   }
908   \par\nopagebreak
909   \addvspace{ \__enumext_multicols_above_v_skip }
910 }

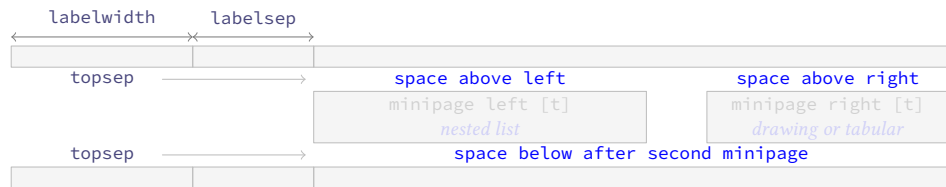
```

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

### 10.19 Adjustment of vertical spaces for minipage

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

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

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

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

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

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

```

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

(End of definition for `__enumext_mini_env*`.)

#### 10.19.1 Adjustment of vertical spaces for minipage in enumext

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

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

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

```

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

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

```

923     \skip_if_eq:nnTF
924     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
925     {
926         \skip_set:Nn \l__enumext_minipage_left_skip
927         {
928             -0.150\box_dp:N \strutbox
929         }
930         \skip_set:Nn \l__enumext_minipage_right_skip
931         {
932             0.695\box_dp:N \strutbox
933         }
934         \skip_set:Nn \l__enumext_minipage_after_skip
935         {
936             \box_dp:N \strutbox
937         }
938         \__enumext_zero_parsep:
939     }
```

```

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

```

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.

```

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

```

(End of definition for `\__enumext_mini_set_vskip:`)

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

```

993 \cs_new_protected:Nn \__enumext_zero_parsep:
994 {
995   \int_case:nn { \l__enumext_level_int }
996   {
997     { 2 }{
998       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
999       {
1000         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }

```

```

1001         }
1002     }
1003     { 3 }{
1004         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1005         {
1006             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1007         }
1008     }
1009     { 4 }{
1010         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1011         {
1012             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1013         }
1014     }
1015 }
1016 }

```

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

`\__enumext_mini_addvspace:` The function `\__enumext_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\__enumext_mini_env*` environment in `enumext`, taking into account whether  $\text{\TeX}$  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*.

```

1017 \cs_new_protected:Nn \__enumext_mini_addvspace:
1018 {
1019     \__enumext_mini_set_vskip:
1020     \mode_if_vertical:T
1021     {
1022         \skip_add:Nn \l__enumext_minipage_left_skip
1023         {
1024             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1025         }
1026         \skip_add:Nn \l__enumext_minipage_after_skip
1027         {
1028             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1029         }
1030     }
1031     \par\nopagebreak
1032     \addvspace { \l__enumext_minipage_left_skip }
1033 }

```

(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`.

```

1034 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1035 {
1036     \skip_zero_new:N \l__enumext_minipage_after_skip
1037     \skip_zero_new:N \l__enumext_minipage_left_skip
1038     \skip_zero_new:N \l__enumext_minipage_right_skip
1039     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1040     {
1041         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1042         {
1043             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1044             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1045             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1046             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1047             {
1048                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1049             }
1050         }
1051     }
1052     {
1053         \skip_set:Nn \l__enumext_minipage_left_skip
1054         {
1055             \skip_use:N \l__enumext_topsep_v_skip
1056         }
1057         \skip_set:Nn \l__enumext_minipage_right_skip

```

```

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

```

(End of definition for `\l__enumext_keyans_mini_set_vskip:`)

`\l__enumext_keyans_mini_addvspace:`

The function `\l__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\l__enumext_mini_env*` environment in `keyans`, taking into account whether  $\text{\TeX}$  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`.

```

1096 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1097 {
1098     \l__enumext_keyans_mini_set_vskip:
1099     \mode_if_vertical:T
1100     {
1101         \skip_add:Nn \l__enumext_minipage_left_skip
1102         {
1103             \l__enumext_partopsep_v_skip
1104         }
1105         \skip_add:Nn \l__enumext_minipage_after_skip
1106         {
1107             \l__enumext_partopsep_v_skip
1108         }
1109     }
1110     \par\nopagebreak
1111     \addvspace { \l__enumext_minipage_left_skip }
1112 }

```

(End of definition for `\l__enumext_keyans_mini_addvspace:`)

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

`\l__enumext_mini_set_vskip_vii:`

`\l__enumext_mini_set_vskip_viii:`

The functions `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `\l__enumext_mini_env*` environment in `enumext*` and `keyans*`.



```

1113 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1114 {
1115   \skip_zero_new:N \l__enumext_minipage_left_skip
1116   \skip_gzero_new:N \g__enumext_minipage_right_skip
1117   \skip_gzero_new:N \g__enumext_minipage_after_skip
1118   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1119   {
1120     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1121     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1122   }
1123   {
1124     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1125     \skip_gset:Nn \g__enumext_minipage_right_skip
1126     {
1127       \l__enumext_topsep_vii_skip
1128     }
1129     \skip_gset:Nn \g__enumext_minipage_after_skip
1130     {
1131       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1132     }
1133   }
1134 }
1135 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1136 {
1137   \skip_zero_new:N \l__enumext_minipage_after_skip
1138   \skip_zero_new:N \l__enumext_minipage_left_skip
1139   \skip_zero_new:N \l__enumext_minipage_right_skip
1140   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1141   {
1142     \skip_set:Nn \l__enumext_minipage_left_skip
1143     {
1144       0.5\box_dp:N \strutbox
1145     }
1146     \skip_set:Nn \l__enumext_minipage_right_skip
1147     {
1148       \l__enumext_partopsep_viii_skip
1149     }
1150     \skip_set:Nn \l__enumext_minipage_after_skip
1151     {
1152       1.6\box_dp:N \strutbox
1153     }
1154   }
1155   {
1156     \skip_set:Nn \l__enumext_minipage_left_skip
1157     {
1158       0.5875\box_dp:N \strutbox
1159     }
1160     \skip_set:Nn \l__enumext_minipage_right_skip
1161     {
1162       \l__enumext_topsep_viii_skip
1163     }
1164     \skip_set:Nn \l__enumext_minipage_after_skip
1165     {
1166       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1167     }
1168   }
1169 }

```

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

`\__enumext_mini_addvspace_vii:`  
`\__enumext_mini_addvspace_viii:`

The functions `\__enumext_mini_addvspace_vii:` and `\__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `\__enumext_mini_env*` environment on the *left side* when the `miniright` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether T<sub>E</sub>X is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to 0pt in both environments.

```

1170 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1171 {
1172   \__enumext_mini_set_vskip_vii:
1173   \par\nopagebreak
1174   \addvspace { \l__enumext_minipage_left_skip }
1175 }

```

```

1176 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1177 {
1178   \__enumext_mini_set_vskip_viii:
1179   \par\nopagebreak
1180   \addvspace { \l__enumext_minipage_left_skip }
1181 }

```

(End of definition for \\_\_enumext\_mini\_addvspace\_vii: and \\_\_enumext\_mini\_addvspace\_viii:.)

#### 10.19.4 The command \miniright

The command `\miniright` will close the `__enumext_mini_env*` environment on the “left side”, open the `__enumext_mini_env*` environment on the “right side” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “right side” environment. The *starred version* ‘\*’ inhibits the use of `\centering` command i.e. the usual L<sup>A</sup>T<sub>E</sub>X 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.

```

1182 \NewDocumentCommand \miniright { s }
1183 {
1184   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1185   {
1186     \msg_error:nnn { enumext } { wrong-miniright-place }
1187   }
1188   \int_compare:nNt { \l__enumext_level_int } = { 0 }
1189   {
1190     \msg_error:nnn { enumext } { wrong-miniright-place }
1191   }
1192   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1193   {
1194     \__enumext_keyans_mini_right_cmd:n {#1}
1195   }
1196   { \__enumext_mini_right_cmd:n {#1} }
1197 }

```

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

`\__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 `\l__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.

```

1198 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1199 {
1200   \dim_compare:nNtF
1201   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1202   {
1203     \__enumext_multicols_stop:
1204     \end{__enumext_mini_env*}
1205     \hfill
1206     \begin{__enumext_mini_env*}
1207     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1208     \par\addvspace { \l__enumext_minipage_right_skip }
1209     \bool_if:nF {#1}
1210     {
1211       \centering
1212     }
1213     \int_gzero:N \g__enumext_minipage_stat_int
1214   }
1215   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1216 }

```

(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.

```

1217 \\cs_new_protected:Npn \\_enumext_keyans_mini_right_cmd:n #1
1218 {
1219   \\dim_compare:nNnTF { \\_enumext_minipage_right_v_dim } > { \\c_zero_dim }
1220   {
1221     \\_enumext_keyans_multicols_stop:
1222     \\end{\\_enumext_mini_env*}
1223     \\hfill
1224     \\begin{\\_enumext_mini_env*}{ \\_enumext_minipage_right_v_dim }
1225     \\par\\addvspace { \\_enumext_minipage_right_skip }
1226     \\bool_if:nF {#1}
1227     {
1228       \\centering
1229     }
1230     \\int_gzero:N \\g__enumext_minipage_stat_int
1231   }
1232   { \\msg_error:nnn { enumext } { wrong-miniright-use } }
1233 }

```

(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.

```

1234 \\cs_set_protected:Npn \\_enumext_tmp:nn #1 #2
1235 {
1236   \\keys_define:nn { enumext / #1 }
1237   {
1238     above .skip_set:c = { \\_enumext_vspace_above_#2_skip },
1239     above .value_required:n = true,
1240     above* .code:n = \\bool_set_true:c { \\_enumext_vspace_a_star_#2_bool }
1241               \\keys_set:nn { enumext / #1 } { above = {##1} },
1242     above* .value_required:n = true,
1243     below .skip_set:c = { \\_enumext_vspace_below_#2_skip },
1244     below .value_required:n = true,
1245     below* .code:n = \\bool_set_true:c { \\_enumext_vspace_b_star_#2_bool }
1246               \\keys_set:nn { enumext / #1 } { below = {##1} },
1247     below* .value_required:n = true,
1248   }
1249 }
1250 \\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.

```

1251 \\cs_new_protected:Nn \\_enumext_vspace_above:
1252 {
1253   \\skip_if_eq:nnF
1254   { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } } { \\c_zero_skip }
1255   {
1256     \\bool_if:cTF { \\_enumext_vspace_a_star_ \\_enumext_level: _bool }
1257     {
1258       \\vspace*{ \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1259     }
1260     {
1261       \\vspace { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1262     }
1263   }
1264 }

```

(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.

```

1265 \cs_new_protected:Nn \__enumext_vspace_below:
1266 {
1267   \skip_if_eq:nnF
1268   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1269   {
1270     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1271     {
1272       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1273     }
1274     {
1275       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1276     }
1277   }
1278 }

```

(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.

```

1279 \cs_new_protected:Nn \__enumext_vspace_above_v:
1280 {
1281   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1282   {
1283     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1284     {
1285       \vspace*{ \l__enumext_vspace_above_v_skip }
1286     }
1287     { \vspace { \l__enumext_vspace_above_v_skip } }
1288   }
1289 }

```

(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.

```

1290 \cs_new_protected:Nn \__enumext_vspace_below_v:
1291 {
1292   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1293   {
1294     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1295     {
1296       \vspace*{ \l__enumext_vspace_below_v_skip }
1297     }
1298     { \vspace { \l__enumext_vspace_below_v_skip } }
1299   }
1300 }

```

(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`:

```

1301 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1302 {
1303   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1304   {
1305     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1306     {
1307       \vspace*{ \l__enumext_vspace_above_vii_skip }
1308     }
1309     { \vspace { \l__enumext_vspace_above_vii_skip } }
1310   }
1311 }
1312 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1313 {
1314   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1315   {

```

```

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

```

(End of definition for \\_\_enumext\_vspace\_above\_vii: and \\_\_enumext\_vspace\_above\_viii:.)

The functions \\_\_enumext\_vspace\_below\_vii: and \\_\_enumext\_vspace\_below\_viii: apply the *vertical space below* the `enumext*` and `keyans*` environments set by the `below*` and `below` keys.

```

1323 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1324 {
1325     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1326     {
1327         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1328         {
1329             \vspace*{ \l__enumext_vspace_below_vii_skip }
1330         }
1331         { \vspace { \l__enumext_vspace_below_vii_skip } }
1332     }
1333 }
1334 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1335 {
1336     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1337     {
1338         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1339         {
1340             \vspace*{ \l__enumext_vspace_below_viii_skip }
1341         }
1342         { \vspace { \l__enumext_vspace_below_viii_skip } }
1343     }
1344 }

```

(End of definition for \\_\_enumext\_vspace\_below\_vii: and \\_\_enumext\_vspace\_below\_viii:.)

## 10.21 Setting save-ans and resume keys

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

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

```

1345 \keys_define:nn { enumext / level-1 }
1346 {
1347     save-ans .code:n = \__enumext_storing_set:n {#1},
1348     save-ans .value_required:n = true,
1349     series .str_set:N = \l__enumext_series_str,
1350     series .value_required:n = true,
1351     resume .code:n = \__enumext_resume_counter_series:n {#1},
1352     resume* .code:n = \__enumext_resume_starred:,
1353     resume* .value_forbidden:n = true,
1354 }
1355 \keys_define:nn { enumext / enumext* }
1356 {
1357     save-ans .code:n = \__enumext_storing_set_vii:n {#1},
1358     save-ans .value_required:n = true,
1359     series .str_set:N = \l__enumext_series_str,
1360     series .value_required:n = true,
1361     resume .code:n = \__enumext_resume_counter_series_vii:n {#1},
1362     resume* .code:n = \__enumext_resume_starred_vii:,
1363     resume* .value_forbidden:n = true,
1364 }

```

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

### 10.21.1 Internal function for save-ans key

\\_\_enumext\_storing\_set:n  
 \\_\_enumext\_storing\_exec:

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

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

```

1365 \cs_new_protected:Npn \__enumext_storing_set:n #1
1366 {
1367   \tl_set:Nx \__enumext_store_name_tl {#1}
1368   \tl_if_empty:NTF \__enumext_store_name_tl
1369   {
1370     \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1371   }
1372   {
1373     \__enumext_storing_standar:
1374   }
1375 }
1376 \cs_new_protected:Npn \__enumext_storing_set_vii:n #1
1377 {
1378   \tl_set:Nx \__enumext_store_name_tl {#1}
1379   \tl_if_empty:NTF \__enumext_store_name_tl
1380   {
1381     \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1382   }
1383   {
1384     \__enumext_storing_starred:
1385   }
1386 }
1387 \cs_new_protected:Nn \__enumext_storing_standar:
1388 {
1389   \bool_if:NTF \__enumext_standar_level_one_bool
1390   {
1391     \__enumext_storing_exec:
1392   }
1393   {
1394     \msg_warning:nnn { enumext } { save-ans-nested } { enumext }
1395   }
1396 }
1397 \cs_new_protected:Nn \__enumext_storing_starred:
1398 {
1399   \bool_if:NTF \__enumext_starred_level_one_bool
1400   {
1401     \__enumext_storing_exec:
1402   }
1403   {
1404     \msg_warning:nnn { enumext } { save-ans-nested } { enumext* }
1405   }
1406 }
1407 \cs_new_protected:Nn \__enumext_storing_exec:
1408 {
1409   \prop_if_exist:cF { g__enumext_ \__enumext_store_name_tl _prop }
1410   {
1411     \prop_new:c { g__enumext_ \__enumext_store_name_tl _prop }
1412   }
1413   \seq_if_exist:cF { g__enumext_ \__enumext_store_name_tl _seq }
1414   {
1415     \seq_new:c { g__enumext_ \__enumext_store_name_tl _seq }
1416   }
1417   \bool_set_true:N \__enumext_store_active_bool
1418   \bool_set_true:N \__enumext_store_ans_bool
1419   \int_if_exist:cF { g__enumext_resume_ \__enumext_store_name_tl _int }
1420   {
1421     \int_new:c { g__enumext_resume_ \__enumext_store_name_tl _int }
1422   }
1423 }

```

(End of definition for \\_\_enumext\_storing\_set:n and \\_\_enumext\_storing\_exec:.)

### 10.21.2 Internal function for series key

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, but, discarding some specific `<keys>`.

```

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

```

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

```

1424 \cs_new:Npn \__enumext_filter_series:n #1
1425 {
1426   \use:e
1427   {
1428     \keyval_parse:NNn
1429     \__enumext_filter_series_key:n
1430     \__enumext_filter_series_pair:nn {#1}
1431   }
1432 }

```

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.

```

1433 \cs_new:Npn \__enumext_filter_series_key:n #1
1434 {
1435   \str_case:nnF {#1}
1436   {
1437     { resume } {}
1438     { resume* } {}
1439   }
1440   { , { \exp_not:n {#1} } }
1441 }

```

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

```

1442 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1443 {
1444   \str_case:nnF {#1}
1445   {
1446     { series } {}
1447     { resume } {}
1448     { columns* } {}
1449     { save-key } {}
1450     { save-ans } {}
1451     { columns-sep* } {}
1452   }
1453   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1454 }

```

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

```

\__enumext_parse_series_resume:n
\__enumext_resume_series_default:n

```

The function `\__enumext_parse_series_resume:n` will be in charge of saving the filtered `<keys>` in a global variable `\g__enumext_series_<series name>_tl` created globally when using the key `series`, otherwise it will call the function `\__enumext_resume_series_default:n`. This function is passed to the function `\__enumext_parse_keys_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).

```

1455 \cs_new_protected:Npn \__enumext_parse_series_resume:n #1
1456 {
1457   \bool_set_false:N \l__enumext_resume_name_bool
1458   \str_if_empty:NTF \l__enumext_series_str
1459   {
1460     \__enumext_resume_series_default:n {#1}
1461   }
1462   {
1463     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1464     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1465     { \__enumext_filter_series:n {#1} }
1466     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1467     {
1468       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1469     }
1470   }
1471 }

```



The function `\__enumext_resume_series_default: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_series_standar_default_tl` for the `enumext` environment and in the variable `\g__enumext_series_starred_default_tl` for the `enumext*` environment. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the `series` key is not being used.

```

1472 \cs_new_protected:Npn \__enumext_resume_series_default:n #1
1473 {
1474   \bool_lazy_all:nT
1475   {
1476     { \bool_if_p:N \g__enumext_standar_bool }
1477     { \int_compare_p:nNn { \__enumext_level_int } = { 1 } }
1478     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
1479   }
1480   {
1481     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT]]}
1482     \tl_gclear:N \g__enumext_series_standar_default_tl
1483     \tl_gset:Ne \g__enumext_series_standar_default_tl { \__enumext_filter_series:n {#1} }
1484   }
1485   \bool_lazy_all:nT
1486   {
1487     { \bool_if_p:N \g__enumext_starred_bool }
1488     { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1489     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
1490   }
1491   {
1492     %%\typeout{[[ON-LEVEL-ONE-ENUMEXT*]]}
1493     \tl_gclear:N \g__enumext_series_starred_default_tl
1494     \tl_gset:Ne \g__enumext_series_starred_default_tl { \__enumext_filter_series:n {#1} }
1495   }
1496 }

```

(End of definition for `\__enumext_parse_series_resume:n` and `\__enumext_resume_series_default:n`)

### 10.21.3 Internal function for resume and resume\* keys

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

```

\__enumext_resume_counter_series:n
\__enumext_resume_counter:
\__enumext_resume_starred:

```

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

```

1497 \cs_new_protected:Npn \__enumext_resume_counter_series:n #1
1498 {
1499   \tl_if_empty:nTF {#1}
1500   {
1501     %%\__enumext_resume_counter:
1502     \__enumext_resume_counter_new:n {#1}
1503   }
1504   {
1505     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1506     {
1507       %\__enumext_resume_counter:
1508       \__enumext_resume_counter_new:n {#1}
1509       \keys_set:nv { enumext / level-1 }
1510       { g__enumext_series_ \tl_to_str:n {#1} _tl }
1511       %\__enumext_resume_counter_new:n {#1}
1512     }
1513     { \msg_error:nnn { enumext } { unknown-series } {#1} }
1514   }
1515 }
1516 \cs_new_protected:Npn \__enumext_resume_counter:
1517 {
1518   \bool_lazy_and:nnT

```

```

1519     { \bool_if_p:N \l__enumext_standar_level_one_bool }
1520     { \bool_if_p:N \l__enumext_store_active_bool }
1521     {
1522         \int_gset:Nn \g__enumext_resume_int
1523         {
1524             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1525         }
1526     }
1527     \int_gincr:N \g__enumext_resume_int
1528     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1529 }
1530 \bool_new:N \l__enumext_resume_name_bool
1531 \tl_new:N \l__enumext_resume_name_tl
1532 \cs_new_protected:Npn \__enumext_resume_counter_new:n #1
1533 {
1534     % Primero incremento
1535     % \int_gincr:N \g__enumext_resume_int
1536     % Ahora chequeo
1537     \tl_if_empty:nTF {#1}
1538     {
1539         \int_gincr:N \g__enumext_resume_int
1540         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1541     }
1542     {
1543         \tl_set:Nn \l__enumext_resume_name_tl {#1}
1544         %\bool_set_true:N \l__enumext_resume_name_bool
1545         %\int_if_exist:cF { g__enumext_series_ \tl_to_str:n {#1} _int }
1546         % {
1547         % }
1548         \int_set:Nn \l_tmpa_int
1549         {
1550             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1551         }
1552         \int_set_eq:NN \l__enumext_start_i_int \l_tmpa_int
1553     }
1554     % Si está la llave save-ans
1555     \bool_lazy_and:nnT
1556     { \bool_if_p:N \l__enumext_standar_level_one_bool }
1557     { \bool_if_p:N \l__enumext_store_active_bool }
1558     {
1559         %\int_gset:Nn \g__enumext_resume_int
1560         \int_set:Nn \l__enumext_start_i_int
1561         {
1562             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1563         }
1564     }
1565     %\int_gincr:N \g__enumext_resume_int
1566     %\int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1567 }
1568 \cs_new_protected:Nn \__enumext_resume_starred:
1569 {
1570     \tl_if_empty:NF \g__enumext_series_standar_default_tl
1571     {
1572         \__enumext_resume_counter:
1573         \keys_set:nV { enumext / level-1 } \g__enumext_series_standar_default_tl
1574     }
1575 }

```

(End of definition for \\_\_enumext\_resume\_counter\_series:n, \\_\_enumext\_resume\_counter:, and \\_\_enumext\_resume\_starred:.)

\\_\_enumext\_resume\_counter\_series\_vii:n

\\_\_enumext\_resume\_counter\_vii:

\\_\_enumext\_resume\_starred\_vii:

```

1576 \cs_new_protected:Npn \__enumext_resume_counter_series_vii:n #1
1577 {
1578     \tl_if_empty:nTF {#1}
1579     {
1580         \__enumext_resume_counter_vii:
1581     }
1582     {
1583         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1584         {

```

```

1585         \__enumext_resume_counter_vii:
1586         \keys_set:nv { enumext / enumext* }
1587         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1588     }
1589     { \msg_error:nnn { enumext } { unknown-series } {#1} }
1590 }
1591 }
1592 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1593 {
1594     \bool_lazy_and:nnT
1595     { \bool_if_p:N \__enumext_starred_level_one_bool }
1596     { \bool_if_p:N \__enumext_store_active_bool }
1597     {
1598         \int_gset:Nn \g__enumext_resume_vii_int
1599         {
1600             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1601         }
1602     }
1603     \int_gincr:N \g__enumext_resume_vii_int
1604     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1605 }
1606 \cs_new_protected:Nn \__enumext_resume_starred_vii:
1607 {
1608     \tl_if_empty:NF \g__enumext_series_starred_default_tl
1609     {
1610         \__enumext_resume_counter_vii:
1611         \keys_set:nv { enumext / enumext* } \g__enumext_series_starred_default_tl
1612     }
1613 }

```

(End of definition for `\__enumext_resume_counter_series_vii:n`, `\__enumext_resume_counter_vii:`, and `\__enumext_resume_starred_vii:`.)

## 10.22 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.

### 10.22.1 Setting check-ans key

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

```

check-ans 1614 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store  1615 {
          1616     \keys_define:nn { enumext / #1 }

```

```

1617     {
1618         check-ans .bool_set:N = \l__enumext_check_ans_bool,
1619         check-ans .initial:n = false,
1620         no-store .code:n = {
1621             \bool_set_false:N \l__enumext_store_ans_bool
1622             \bool_set_false:N \l__enumext_check_ans_bool
1623         },
1624         no-store .value_forbidden:n = true,
1625     }
1626 }
1627 \clist_map_inline:nn
1628 {
1629     level-1, level-2, level-3, level-4, enumext*
1630 }
1631 { \l__enumext_tmp:n {#1} }

```

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

### 10.22.2 Set-up check answer mechanism

`\l__enumext_check_ans_set:` The function `\l__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.

```

1632 \cs_new_protected:Nn \l__enumext_check_ans_set:
1633 {
1634     \int_case:nn { \l__enumext_level_int }
1635     {
1636         { 1 }{
1637             \bool_lazy_all:nT
1638             {
1639                 { \bool_if_p:N \g__enumext_starred_bool }
1640                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1641             }
1642             {
1643                 \int_gdecr:N \g__enumext_count_item_number_int
1644                 \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1645             }
1646         }
1647         { 2 }{
1648             \int_gdecr:N \g__enumext_count_item_number_int
1649         }
1650         { 3 }{
1651             \int_gdecr:N \g__enumext_count_item_number_int
1652         }
1653         { 4 }{
1654             \int_gdecr:N \g__enumext_count_item_number_int
1655         }
1656     }
1657     \int_case:nn { \l__enumext_level_h_int }
1658     {
1659         { 1 }{
1660             \bool_if:NT \g__enumext_standar_bool
1661             {
1662                 \int_gdecr:N \g__enumext_count_item_number_int
1663                 \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1664             }
1665         }
1666     }
1667 }

```

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

`\l__enumext_check_ans_exec:` The function `\l__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.

```

1668 \cs_new_protected:Nn \l__enumext_check_ans_exec:
1669 {
1670     \bool_if:NT \l__enumext_check_ans_bool
1671     {
1672         \l__enumext_check_ans_set:
1673     }

```

```
1674 }
```

(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.

```
1675 \cs_new_protected:Nn \__enumext_check_ans_show:
1676 {
1677   \int_compare:nNnTF
1678     { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1679     {
1680       \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1681     }
1682     {
1683       \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1684     }
1685     \int_gzero:N \g__enumext_count_item_number_int
1686     \int_gzero:N \g__enumext_count_item_anskey_int
1687 }
```

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

## 10.23 Keys and functions associated with storage

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “storage system” and internal mechanism of “*label and ref*” only at the first level of `enumext` and `enumext*`.

```
1688 \cs_set_protected:Npn \__enumext_tmp:n #1
1689 {
1690   \keys_define:nn { enumext / #1 }
1691   {
1692     wrap-ans      .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1693     wrap-ans      .initial:n = \fbox{##1},
1694     wrap-ans      .value_required:n = true,
1695     wrap-opt      .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1696     wrap-opt      .initial:n = [{##1}],
1697     wrap-opt      .value_required:n = true,
1698     save-sep      .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1699     save-sep      .initial:n = {, ~ },
1700     save-sep      .value_required:n = true,
1701     mark-ans      .tl_set:N = \l__enumext_mark_answer_sym_tl,
1702     mark-ans      .initial:n = \textasteriskcentered,
1703     mark-ans      .value_required:n = true,
1704     mark-pos      .choice:,
1705     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1706     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1707     mark-pos      .initial:n = right,
1708     mark-pos      .value_required:n = true,
1709     show-ans      .bool_set:N = \l__enumext_show_answer_bool,
1710     show-ans      .initial:n = false,
1711     show-ans      .value_required:n = true,
1712     show-pos      .bool_set:N = \l__enumext_show_position_bool,
1713     show-pos      .initial:n = false,
1714     show-pos      .value_required:n = true,
1715     mark-ref      .tl_set:N = \l__enumext_mark_ref_sym_tl,
1716     mark-ref      .initial:n = \textasteriskcentered,
1717     mark-ref      .value_required:n = true,
1718     save-ref      .bool_set:N = \l__enumext_store_ref_key_bool,
1719     save-ref      .initial:n = false,
1720     save-ref      .value_required:n = true,
1721   }
1722 }
1723 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

```
mark-pos
show-ans
```

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

```
1724 \cs_set_protected:Npn \__enumext_tmp:n #1
1725 {
1726   \keys_define:nn { enumext / #1 }
```

```

1727     {
1728       mark-pos .choice:,
1729       mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1730       mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1731       mark-pos .initial:n = right,
1732       mark-pos .value_required:n = true,
1733       show-ans .bool_set:N = \__enumext_show_answer_bool,
1734       show-ans .initial:n = false,
1735       show-ans .value_required:n = true,
1736       show-pos .bool_set:N = \__enumext_show_position_bool,
1737       show-pos .initial:n = false,
1738       show-pos .value_required:n = true,
1739     }
1740   }
1741   \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1742   \cs_set_protected:Npn \__enumext_tmp:n #1 #2
1743   {
1744     \keys_define:nn { enumext / #1 }
1745     {
1746       columns* .code:n = \bool_set_true:c { \__enumext_store_columns_#2_bool }
1747       \int_set:cn { \__enumext_store_columns_#2_int } {##1}
1748       \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1749       {
1750         columns = \exp_not:v { \__enumext_store_columns_#2_int },
1751       },
1752       columns* .value_required:n = true,
1753       columns-sep* .code:n = \bool_set_true:c { \__enumext_store_columns_sep_#2_bool }
1754       \dim_set:cn { \__enumext_store_columns_sep_#2_dim } {##1}
1755       \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1756       {
1757         columns-sep = \exp_not:v { \__enumext_store_columns_sep_#2_dim },
1758       },
1759       columns-sep* .value_required:n = true,
1760     }
1761   }
1762   \clist_map_inline:nn
1763   {
1764     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1765   }
1766   { \__enumext_tmp:n #1 }

```

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

### 10.23.1 Function for storing content in prop list

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

`\__enumext_store_addto_prop:V`

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

```

1767   \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1768   \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1769   {
1770     \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
1771     {
1772       \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
1773     }
1774     { #1 }
1775   }
1776   \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

### 10.23.2 Function for storing content in sequence

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

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

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

```

1777 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1778 {
1779   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
1780 }
1781 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

### 10.23.3 Functions for storing the list structure in the sequence

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

```

1782 \cs_new_protected:Nn \__enumext_store_level_open:
1783 {
1784   \bool_if:NT \l__enumext_store_ans_bool
1785   {
1786     \tl_if_empty:CTF { l__enumext_store_opt_ \__enumext_level: _tl }
1787     {
1788       \__enumext_store_addto_seq:n
1789       {
1790         \item \begin{enumext}
1791       }
1792     }
1793     {
1794       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1795       {
1796         \item \begin{enumext} [
1797       }
1798       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1799       {
1800         ]
1801       }
1802       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1803     }
1804   }
1805 }
1806 \cs_new_protected:Nn \__enumext_store_level_close:
1807 {
1808   \bool_if:NT \l__enumext_store_ans_bool
1809   {
1810     \__enumext_store_addto_seq:n { \end{enumext} }
1811   }
1812 }

```

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

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

```

1813 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1814 {
1815   \bool_if:NT \l__enumext_store_ans_bool
1816   {
1817     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1818     {
1819       \__enumext_store_addto_seq:n
1820       {

```



```

1821         \item \mode_leave_vertical:
1822         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1823         \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1824     }
1825 }
1826 {
1827     \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1828     {
1829         \item \mode_leave_vertical:
1830         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1831         \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1832     }
1833     \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1834     {
1835     }
1836 }
1837 \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1838 }
1839 }
1840 }
1841 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1842 {
1843     \bool_if:NT \l__enumext_store_ans_bool
1844     {
1845         \__enumext_store_addto_seq:n { \end{enumext*} }
1846     }
1847 }

```

(End of definition for \\_\_enumext\_store\_level\_open\_vii: and \\_\_enumext\_store\_level\_close\_vii:.)

#### 10.23.4 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`

```

1848 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1849 {
1850     \mode_leave_vertical:
1851     \skip_horizontal:n { -\dim_use:N #2 }
1852     \makebox[0pt][ r ]
1853     {
1854         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1855         {
1856             \tl_use:N \l__enumext_mark_answer_sym_tl
1857         }
1858     }
1859     \skip_horizontal:n { \dim_use:N #2 }
1860 }
1861 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for \\_\_enumext\_print\_keyans\_box:NN.)

### 10.24 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`.

```

1862 \keys_define:nn { enumext / anskey }
1863 {
1864     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1865     item-sym* .value_required:n = true,
1866     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1867     item-pos* .value_required:n = true,
1868     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1869     store-brk .default:n = true,
1870     store-brk .value_forbidden:n = true,
1871 }

```

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>`.

```

1872 \NewDocumentCommand \anskey { d() s o +m }
1873 {
1874   \bool_if:NF \l__enumext_store_active_bool
1875   {
1876     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1877   }
1878   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1879   {
1880     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1881   }
1882   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1883   {
1884     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1885   }
1886   \group_begin:
1887     \bool_if:NT \l__enumext_store_ans_bool
1888     {
1889       \bool_if:NT \l__enumext_check_ans_bool
1890       {
1891         \int_gincr:N \g__enumext_count_item_anskey_int
1892       }
1893       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1894     }
1895   \group_end:
1896 }

```

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

`\__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.

```

1897 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1898 {
1899   \__enumext_store_addto_prop:n {#4}
1900   \bool_if:NT \l__enumext_store_ref_key_bool
1901   {
1902     \__enumext_store_internal_ref:
1903   }
1904   \__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`.

```

1905   \tl_clear:N \l__enumext_store_anskey_arg_tl
1906   \tl_if_novalue:nF {#3}
1907   {
1908     \keys_set:nn { enumext / anskey } {#3}
1909   }
1910   \bool_lazy_and:nnT
1911   { \bool_if_p:N \l__enumext_store_columns_break_bool }
1912   { \bool_not_p:n { \l__enumext_starred_bool } }
1913   {
1914     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1915   }
1916   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

1917   \tl_if_novalue:nF {#1}
1918   {
1919     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1920     \bool_if:NT \l__enumext_starred_bool

```

```

1921     {
1922         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1923         {
1924             ( \exp_not:V \l__enumext_store_columns_join_int )
1925         }
1926     }
1927 }

```

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`.

```

1928 \bool_if:nTF {#2}
1929 {
1930     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1931     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1932     {
1933         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1934         {
1935             [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1936         }
1937     }
1938     \dim_compare:nT
1939     {
1940         \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1941     }
1942     {
1943         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1944         {
1945             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1946         }
1947     }
1948     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1949 }
1950 {
1951     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1952 }

```

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

```

1953 \bool_lazy_and:nnT
1954 { \bool_if_p:N \l__enumext_store_ref_key_bool }
1955 { \bool_if_p:N \l__enumext_hyperref_bool }
1956 {
1957     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1958     {
1959         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1960         { \exp_not:V \l__enumext_mark_ref_sym_tl }
1961     }
1962 }
1963 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1964 }

```

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

`\__enumext_store_internal_ref:`

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

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

```

1965 \cs_new_protected:Nn \__enumext_store_internal_ref:
1966 {
1967     \cs_set_protected:Npn \__enumext_tmp:n ##1
1968     {
1969         \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
1970         \tl_reverse:c { l__enumext_label_copy_##1_tl }
1971         \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
1972         \tl_reverse:c { l__enumext_label_copy_##1_tl }
1973     }
1974     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1975     \cs_set:Npn \__enumext_tmp:n ##1
1976     { . \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.

```

1977 \bool_lazy_all:nT
1978 {
1979   { \bool_if_p:N \g__enumext_starred_bool }
1980   { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1981 }
1982 {
1983   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1984   { \tl_use:N \l__enumext_label_copy_vii_tl }
1985 }
1986 \bool_lazy_all:nT
1987 {
1988   { \bool_if_p:N \l__enumext_standar_bool }
1989   { \bool_if_p:N \g__enumext_starred_bool }
1990   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1991 }
1992 {
1993   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1994   {
1995     \tl_use:N \l__enumext_label_copy_vii_tl
1996     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1997   }
1998 }

```

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.

```

1999 \bool_lazy_all:nT
2000 {
2001   { \bool_if_p:N \l__enumext_standar_bool }
2002   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2003   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2004   { \bool_not_p:n { \l__enumext_starred_bool } }
2005 }
2006 {
2007   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2008   {
2009     \tl_use:N \l__enumext_label_copy_i_tl
2010     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2011   }
2012 }
2013 \cs_set:Npn \__enumext_tmp:n ##1
2014 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2015 \bool_lazy_all:nT
2016 {
2017   { \bool_if_p:N \l__enumext_standar_bool }
2018   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2019   { \bool_not_p:n { \g__enumext_starred_bool } }
2020   { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2021 }
2022 {
2023   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2024   {
2025     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2026     . \tl_use:N \l__enumext_label_copy_vii_tl
2027   }
2028 }

```

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

```

2029 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2030 {
2031   \l__enumext_store_name_tl \c_colon_str
2032   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2033 }

```

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

```

2034 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2035 {
2036   \__enumext_newlabel:nn
2037   { \exp_not:V \l__enumext_newlabel_arg_one_tl }

```

```

2038         { \l__enumext_newlabel_arg_two_tl }
2039     }
2040     \l__enumext_store_write_aux_file_tl
2041 }

```

(End of definition for \l\_\_enumext\_store\_internal\_ref:.)

\l\_\_enumext\_store\_anskey\_show\_wrap:n

The function \l\_\_enumext\_store\_anskey\_show\_wrap:n “wraps” the *⟨argument⟩* passed to \anskey when using the wrap-ans key.

```

2042 \cs_new_protected:Npn \l__enumext_store_anskey_show_wrap:n #1
2043 {
2044     \par
2045     \bool_if:NT \l__enumext_starred_bool
2046     {
2047         \cs_set:Nn \l__enumext_level: { vii }
2048     }
2049     \__enumext_print_keyans_box:cc
2050     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2051     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2052     \__enumext_anskey_wrapper:n { #1 }
2053 }

```

(End of definition for \l\_\_enumext\_store\_anskey\_show\_wrap:n.)

\l\_\_enumext\_store\_anskey\_show\_left:n

The function \l\_\_enumext\_store\_anskey\_show\_left:n will show the “mark” defined by the mark-ans key or the “position” of the content stored in the *⟨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.

```

2054 \cs_new_protected:Npn \l__enumext_store_anskey_show_left:n #1
2055 {
2056     \bool_if:NT \l__enumext_show_answer_bool
2057     {
2058         \__enumext_store_anskey_show_wrap:n { #1 }
2059     }
2060     \bool_if:NT \l__enumext_show_position_bool
2061     {
2062         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2063         {
2064             \group_begin:
2065             \exp_not:N \normalfont
2066             \exp_not:N \footnotesize [ \int_eval:n
2067             {
2068                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2069             }
2070             ]
2071             \group_end:
2072         }
2073         \__enumext_store_anskey_show_wrap:n { #1 }
2074     }
2075 }

```

(End of definition for \l\_\_enumext\_store\_anskey\_show\_left:n.)

## 10.25 Common functions for keyans, keyans\* and keyanspic

### 10.25.1 Storing content in prop list

\l\_\_enumext\_keyans\_addto\_prop:n

The function \l\_\_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 \l\_\_enumext\_store\_addto\_prop:V.

```

2076 \cs_new_protected:Npn \l__enumext_keyans_addto_prop:n #1
2077 {
2078     \tl_clear:N \l__enumext_store_keyans_label_tl
2079     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2080     {
2081         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2082     }
2083 }

```

```

2084     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2085   }
2086   \tl_if_novalue:nF { #1 }
2087   {
2088     % Set save-sep
2089     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2090     {
2091       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2092     }
2093     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2094   }
2095   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2096 }

```

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

### 10.25.2 The save-ref key for keyans, keyans\* and keyanspic

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

```

\__enumext_keyans_store_ref:
  \__enumext_keyans_store_ref_aux_i:
  \__enumext_keyans_store_ref_aux_ii:

```

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

```

2097 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2098 {
2099   \bool_if:NT \l__enumext_store_ref_key_bool
2100   {
2101     \cs_set_protected:Npn \__enumext_tmp:n #1
2102     {
2103       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2104       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2105       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2106       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2107     }
2108     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2109     \__enumext_keyans_store_ref_aux_i:
2110   }
2111 }

```

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

```

2112 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2113 {
2114   \bool_if:NT \g__enumext_starred_bool
2115   {
2116     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2117   }
2118   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2119   {
2120     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2121     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2122   }
2123   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2124   {
2125     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2126     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2127   }
2128   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2129   {
2130     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2131     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2132   }
2133   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2134   {
2135     \l__enumext_store_name_tl \c_colon_str
2136     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2137   }
2138   \__enumext_keyans_store_ref_aux_ii:

```

```
2139 }
```

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.

```
2140 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2141 {
2142   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2143   {
2144     \__enumext_newlabel:nn
2145     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2146     { \l__enumext_newlabel_arg_two_tl }
2147   }
2148   \l__enumext_store_write_aux_file_tl
2149 }
```

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

### 10.25.3 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.

```
2150 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2151 {
2152   \tl_clear:N \l__enumext_store_keyans_label_tl
2153   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2154   {
2155     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2156   }
2157   {
2158     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2159   }
2160   \tl_if_novalue:nF { #1 }
2161   {
2162     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2163     {
2164       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2165     }
2166     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2167   }
2168   \__enumext_keyans_addto_seq_link:
2169 }
```

Checks if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store using the `\__enumext_store_addto_seq:V` function. Finally, copy the contents of the variable `\l__enumext_store_keyans_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `\__enumext_keyans_check_ans:nn` and increment the value of the integer variable `\g__enumext_count_item_anskey_int` handled by the `check-ans` key.

```
2170 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2171 {
2172   \bool_lazy_and:nnT
2173   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2174   { \bool_if_p:N \l__enumext_hyperref_bool }
2175   {
2176     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2177     {
2178       \hfill \exp_not:N \hyperlink
2179       {
2180         \exp_not:V \l__enumext_newlabel_arg_one_tl
2181       }
2182       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2183     }
2184   }
2185   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2186   \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2187   \bool_if:NT \l__enumext_check_ans_bool
2188   {
2189     \int_gincr:N \g__enumext_count_item_anskey_int
```



```

2190     }
2191   }

```

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

#### 10.25.4 Check for starred commands

```
\__enumext_keyans_check_ans:nn
```

The function `\__enumext_keyans_check_ans:nn` performs an extra check for the `keyans` and `keyanspic` environments. Unlike the check executed by `check-ans` key this one is not controlled by any key, it is intended to prevent the forgetting of `\item*` or `\anspic*` in these environments.

```

2192 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2193 {
2194   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2195   {
2196     \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2197   }
2198   { \tl_gclear:N \g__enumext_check_ans_item_tl }
2199 }

```

(End of definition for `\__enumext_keyans_check_ans:nn.`)

#### 10.25.5 The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off `⟨label⟩` are incorrect.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:

```

Common function to show *starred commands* `\item*` and `⟨position⟩` of stored content in `⟨prop list⟩` for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_` `\__enumext_store_name_tl` `_prop` for `show-pos` key.

```

2200 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2201 {
2202   \tl_if_novalue:nF { #1 }
2203   {
2204     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2205   }
2206   \bool_if:NT \l__enumext_show_answer_bool
2207   {
2208     \__enumext_keyans_show_ans:
2209   }
2210   \bool_if:NT \l__enumext_show_position_bool
2211   {
2212     \__enumext_keyans_show_pos:
2213   }
2214 }
2215 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2216 {
2217   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2218   {
2219     \bool_lazy_or:nnT
2220     { \bool_if_p:N \l__enumext_show_answer_bool }
2221     { \bool_if_p:N \l__enumext_show_position_bool }
2222     {
2223       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2224     }
2225   }
2226 }
2227 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2228 {
2229   \tl_put_left:Nn \l__enumext_label_v_tl
2230   {
2231     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2232   }
2233 }
2234 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2235 {
2236   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2237   {
2238     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2239     {
2240       \group_begin:
2241       \exp_not:N \normalfont
2242       \exp_not:N \footnotesize [ \int_eval:n
2243       {

```

```

2244         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2245     }
2246 ]
2247 \group_end:
2248 }
2249 }
2250 {
2251     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2252     {
2253         \group_begin:
2254         \exp_not:N \normalfont
2255         \exp_not:N \footnotesize [ \int_eval:n
2256             {
2257                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2258             }
2259         ]
2260         \group_end:
2261     }
2262 }
2263 \tl_put_left:Nn \l__enumext_label_v_tl
2264 {
2265     \__enumext_print_keyans_box:NN
2266     \l__enumext_labelwidth_i_dim
2267     \l__enumext_labelsep_i_dim
2268 }
2269 }

```

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

## 10.26 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*
2270 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2271 {
2272     \keys_define:nn { enumext / #1 }
2273     {
2274         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2275         item-sym* .value_required:n = true,
2276         item-sym* .initial:n = { $\star$ },
2277         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2278         item-pos* .value_required:n = true,
2279     }
2280 }
2281 \clist_map_inline:nn
2282 {
2283     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2284 }
2285 { \__enumext_tmp:nn #1 }

```

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

## 10.27 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*.

```

2286 \cs_new_protected:Nn \__enumext_footnotetext:nn
2287 {
2288     \footnotetext[#1]{#2}
2289 }
2290 \cs_new_protected:Nn \__enumext_renew_footnote:
2291 {
2292     \seq_gclear:N \g__enumext_footnote_arg_seq
2293     \seq_gclear:N \g__enumext_footnote_int_seq
2294     \RenewDocumentCommand \footnote { o +m }
2295     {
2296         \tl_if_novalue:nTF {##1}
2297         {
2298             \stepcounter{footnote}

```

```

2299         \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2300     }
2301     {
2302         \int_gset:Nn \g__enumext_footnote_int { ##1 }
2303     }
2304     \footnotemark [ \g__enumext_footnote_int ]
2305     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2306     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2307 }
2308 }
2309 \cs_new_protected:Nn \__enumext_print_footnote:
2310 {
2311     \seq_if_empty:NF \g__enumext_footnote_int_seq
2312     {
2313         \seq_map_pairwise_function:NNN
2314         \g__enumext_footnote_int_seq
2315         \g__enumext_footnote_arg_seq
2316         \__enumext_footnotetext:nn
2317     }
2318 }

```

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

## 10.28 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.

### 10.28.1 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:` (§10.29).

```

2319 \cs_new_protected:Npn \__enumext_default_item:n #1
2320 {
2321     \tl_if_novalue:nTF {#1}
2322     {
2323         \bool_if:NT \l__enumext_check_ans_bool
2324         {
2325             \int_gincr:N \g__enumext_count_item_number_int
2326         }
2327         \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2328         \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2329     }
2330     {
2331         \bool_set_eq:cc
2332         { l__enumext_wrap_label_ \__enumext_level: _bool }
2333         { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2334         \__enumext_item_std:w [ #1 ] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2335     }
2336 }

```

(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 `\__enumext_check_ans_bool` set by the key `check-ans`, set the boolean variable `\__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 `\__enumext_wrap_label_X_bool` and the vars `\__enumext_item_symbol_sep_X_dim`, `\__enumext_item_symbol_tl` are used by the function `\__enumext_make_label:` (§10.29).

```

2337 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2338 {
2339   \tl_if_novalue:nF {#1}
2340   {
2341     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2342   }
2343   \tl_gset_eq:Nc \__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2344   \tl_if_novalue:nTF {#2}
2345   {
2346     \dim_set_eq:cc
2347     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2348     { \__enumext_labelsep_ \__enumext_level: _dim }
2349   }
2350   {
2351     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2352   }
2353   \bool_if:NT \__enumext_check_ans_bool
2354   {
2355     \int_gincr:N \__enumext_count_item_number_int
2356   }
2357   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2358   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2359 }

```

(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).

```

2360 \cs_new_protected:Nn \__enumext_redefine_item:
2361 {
2362   \RenewDocumentCommand \item { s o o }
2363   {
2364     \bool_if:nTF {##1}
2365     {
2366       \__enumext_starred_item:nn {##2} {##3}
2367     }
2368     { \__enumext_default_item:n {##2} }
2369   }
2370 }

```

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

### 10.28.2 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`.

```

2371 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2372 {
2373   \tl_if_novalue:nTF { #1 }
2374   {
2375     \bool_set_true:N \__enumext_wrap_label_v_bool
2376     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2377   }
2378   {
2379     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2380     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2381   }
2382 }

```

(End of definition for \\_\_enumext\_keyans\_default\_item:n.)

\\_\_enumext\_keyans\_starred\_item:n

The function \\_\_enumext\_keyans\_starred\_item:n which will make a temporary copy of the current  $\langle label \rangle$ , execute the `show-ans` or `show-pos` keys using the function \\_\_enumext\_keyans\_show\_left:n and will display the contents of that item using the internal copy \\_\_enumext\_item\_std:w, this is necessary to prevent incrementing the current “counter” of the original  $\langle label \rangle$ .

```
2383 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2384 {
2385   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2386   \__enumext_keyans_show_left:n { #1 }
2387   \bool_set_true:N \__enumext_wrap_label_v_bool
2388   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item
```

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

```
2389   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2390   \__enumext_keyans_addto_prop:n { #1 }
2391   \__enumext_keyans_store_ref:
2392   \__enumext_keyans_addto_seq:n { #1 }
2393 }
```

(End of definition for \\_\_enumext\_keyans\_starred\_item:n.)

\item\*  
\\_\_enumext\_keyans\_redefine\_item:

The function \\_\_enumext\_keyans\_redefine\_item: is responsible for adding the *starred* and *optional* argument by the \\_\_enumext\_list\_arg\_two\_v: function in the definition of the `keyans` environment. Here we need to use \peek\_remove\_spaces:n to prevent an unwanted space when using \item\* in conjunction with the `itemindent` key.

This function is passed to \\_\_enumext\_list\_arg\_two\_v: which is used in the definition of the `keyans` environment (§10.31).

```
2394 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2395 {
2396   \RenewDocumentCommand \item { s o }
2397   {
2398     \bool_if:nTF {##1}
2399     {
2400       \peek_remove_spaces:n
2401       {
2402         \__enumext_keyans_starred_item:n {##2}
2403       }
2404     }
2405     {
2406       \__enumext_keyans_default_item:n {##2}
2407     }
2408   }
2409 }
```

(End of definition for \item\* and \\_\_enumext\_keyans\_redefine\_item:. This function is documented on page 12.)

## 10.29 Redefining \makelabel command

Redefine \makelabel for the keys `align`, `font`, `wrap-label`, `wrap-label*` and \item\* for `enumext` and `keyans` environments.

### 10.29.1 Redefining \makelabel for enumext

\\_\_enumext\_item\_starred:

The function \\_\_enumext\_item\_starred: will be responsible for executing \item\* for the `enumext` environment.

```
2410 \cs_new_protected:Nn \__enumext_item_starred:
2411 {
2412   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2413   {
2414     \mode_leave_vertical:
2415     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2416     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2417     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2418   }
2419 }
```

(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).

```

2420 \cs_new_protected:Nn \__enumext_make_label:
2421 {
2422   \RenewDocumentCommand \makelabel { m }
2423   {
2424     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2425     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2426     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2427     {
2428       \__enumext_item_starred:
2429       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2430     }
2431     { ##1 }
2432     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2433     \tl_gclear:N \g__enumext_item_symbol_tl
2434   }
2435 }

```

(End of definition for `\__enumext_make_label:`)

### 10.29.2 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).

```

2436 \cs_new_protected:Nn \__enumext_keyans_make_label:
2437 {
2438   \RenewDocumentCommand \makelabel { m }
2439   {
2440     \tl_use:N \l__enumext_label_fill_left_v_tl
2441     \tl_use:N \l__enumext_label_font_style_v_tl
2442     \bool_if:NTF \l__enumext_wrap_label_v_bool
2443     {
2444       \__enumext_wrapper_label_v:n { ##1 }
2445     }
2446     { ##1 }
2447     \tl_use:N \l__enumext_label_fill_right_v_tl
2448   }
2449 }

```

(End of definition for `\__enumext_keyans_make_label:`)

## 10.30 Calculation of `\leftmargin` and `\itemindent`

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

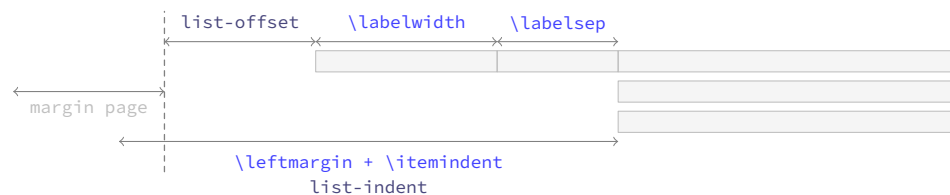


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

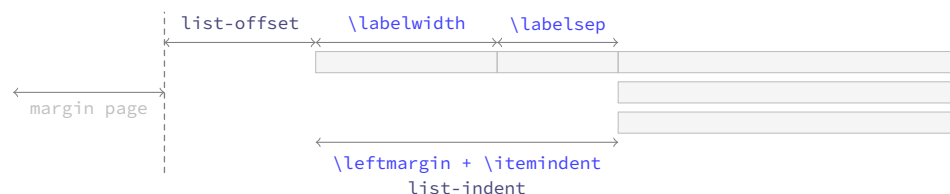


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

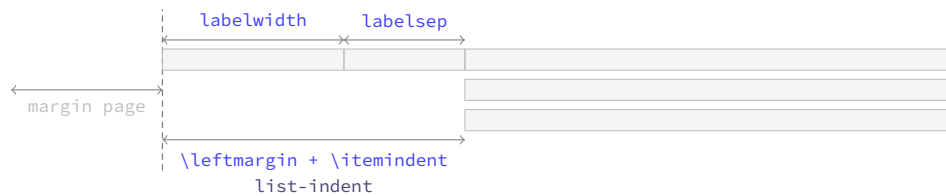
Where the default values will look like in the figure 11.

```

\__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).

```

2450 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2451 {
2452   \dim_compare:nNt { #1 } < { \c_zero_dim }
2453   {
2454     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2455     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2456   }
2457   \dim_compare:nNt { #2 } < { \c_zero_dim }
2458   {
2459     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2460     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2461   }

```

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

```

2462   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2463   \dim_compare:nNtF { #4 } < { \c_zero_dim }
2464   {
2465     \dim_set:Nn #6 { #1 + #2 - #4 }
2466     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2467   }
2468   {
2469     \dim_compare:nNt { #4 } = { #1 + #2 }
2470     { \dim_set:Nn #6 { \c_zero_dim } }
2471     \dim_compare:nNt { #4 } < { #1 + #2 }
2472     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2473     \dim_compare:nNt { #4 } > { #1 + #2 }
2474     {
2475       \dim_set:Nn #6 { -#1 - #2 + #4 }
2476       \dim_set:Nn #6 { #6*-1 }
2477     }
2478     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2479   }
2480 }
2481 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

### 10.31 Setting second argument of the lists

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

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments. We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

2482 \cs_set_protected:Npn \__enumext_tmp:n #1
2483 {

```



```

2484 \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2485 {
2486   \__enumext_calc_hspace:cccccc
2487   { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2488   { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2489   { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2490   { \__enumext_leftmargin_tmp_#1_bool }
2491 \clist_map_inline:nn
2492   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2493   { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2494 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2495   { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2496 \usecounter { enumX#1 }
2497 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2498 \str_if_eq:nnTF {#1} { v }
2499 {
2500   \__enumext_keyans_redefine_item:
2501   \__enumext_keyans_make_label:
2502   \__enumext_keyans_fake_item:
2503   \bool_if:cT { \__enumext_show_length_#1_bool }
2504   {
2505     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2506   }
2507 }
2508 {
2509   \__enumext_redefine_item:
2510   \__enumext_make_label:
2511   \__enumext_use_key_ref:
2512   \__enumext_fake_item:
2513   \bool_if:cT { \__enumext_show_length_#1_bool }
2514   {
2515     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_#1_int }
2516   }
2517 }
2518 }
2519 }
2520 \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 `\parskip` locally.

```

2521 \cs_set_protected:Npn \__enumext_tmp:n #1
2522 {
2523   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2524   {
2525     \__enumext_calc_hspace:cccccc
2526     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2527     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2528     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2529     { \__enumext_leftmargin_tmp_#1_bool }
2530 \clist_map_inline:nn
2531   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2532   { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2533 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2534   { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2535 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2536 \skip_zero:N \partopsep
2537 \usecounter { enumX#1 }
2538 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2539 \__enumext_use_key_ref_h:
2540 \str_if_eq:nnTF {#1} { vii }
2541 {
2542   \__enumext_fake_item_vii:
2543   \bool_if:cT { \__enumext_show_length_vii_bool }
2544   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2545 }
2546 {
2547   \__enumext_fake_item_viii:

```

```

2548         \bool_if:cT { \__enumext_show_length_#1_bool }
2549         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2550     }
2551 }
2552 }
2553 \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.

```

2554 \NewDocumentEnvironment{enumext}{0}{ }
2555 {
2556     \__enumext_current_env:
2557     \__enumext_safe_exec:
2558     \__enumext_parse_keys:n {#1}
2559     \__enumext_before_list:
2560     \__enumext_start_store_level:
2561     \__enumext_start_list:nn
2562     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2563     {
2564         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2565         \__enumext_before_keys_exec:
2566     }
2567     \__enumext_after_args_exec:
2568 }
2569 {
2570     \__enumext_stop_list:
2571     \__enumext_stop_store_level:
2572     \__enumext_after_list:
2573 }

```

(End of definition for enumext. This function is documented on page 5.)

**\\_\_enumext\_safe\_exec:** First check the maximum nesting level for the **enumext** environment and set the state of the booleans vars **\l\_\_enumext\_standar\_bool** and **\l\_\_enumext\_standar\_first\_bool** to “true”, the latter only if the environment is NOT nested in the **enumext\*** environment.

```

2574 \cs_new_protected:Nn \__enumext_safe_exec:
2575 {
2576     \int_incr:N \l__enumext_level_int
2577     \int_compare:nNt { \l__enumext_level_int } > { 4 }
2578     { \msg_fatal:nn { enumext } { list-too-deep } }
2579     \bool_set_true:N \l__enumext_standar_bool
2580     \bool_lazy_all:nT
2581     {
2582         { \bool_if_p:N \g__enumext_standar_bool }
2583         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2584         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2585     }
2586     {
2587         \typeout{[[ON-FIRST-LEVEL-ENUMEXT-NOT-NESTED]]}
2588         \bool_set_true:N \l__enumext_standar_level_one_bool
2589     }
2590 }

```

(End of definition for \\_\_enumext\_safe\_exec:.)

**\\_\_enumext\_parse\_keys:n** Parse [**⟨key = val⟩**] by levels in **enumext**. If the variable **\l\_\_enumext\_store\_active\_bool** is true it will call the function **\\_\_enumext\_parse\_store\_keys:n** and reprocess the **⟨keys⟩** to pass them to the storage sequence.

```

2591 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2592 {
2593     \tl_if_novalue:nF {#1}
2594     {
2595         \str_clear:N \l__enumext_series_str
2596         \int_compare:nNtF { \l__enumext_level_int } = { 1 }
2597         {
2598             \keys_set:nn { enumext / level-1 } {#1}
2599             \__enumext_parse_series_resume:n {#1}
2600         }

```

```

2601     {
2602         \exp_args:Ne \keys_set:nn
2603         { enumext / level-\int_use:N \__enumext_level_int } {#1}
2604     }
2605     \bool_if:NT \__enumext_store_active_bool
2606     {
2607         \__enumext_parse_store_keys:n {#1}
2608     }
2609 }
2610 }

```

(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 \\_\_enumext\_store\_opt\_X\_tl which is used by the function \\_\_enumext\_store\_level\_open:.

```

2611 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2612 {
2613     \bool_if:cF { \__enumext_store_columns_ \__enumext_level: _bool }
2614     {
2615         \regex_match:nnT { \b columns\b } {#1}
2616         {
2617             \int_set_eq:cc
2618             { \__enumext_store_columns_ \__enumext_level: _int }
2619             { \__enumext_columns_ \__enumext_level: _int }
2620             \tl_put_right:ce { \__enumext_store_opt_ \__enumext_level: _tl }
2621             {
2622                 columns = \exp_not:v { \__enumext_store_columns_ \__enumext_level: _int },
2623             }
2624         }
2625     }
2626     \bool_if:cF { \__enumext_store_columns_sep_ \__enumext_level: _bool }
2627     {
2628         \regex_match:nnT { \b columns-sep\b } {#1}
2629         {
2630             \dim_set_eq:cc
2631             { \__enumext_store_columns_sep_ \__enumext_level: _dim }
2632             { \__enumext_columns_sep_ \__enumext_level: _dim }
2633             \tl_put_right:ce { \__enumext_store_opt_ \__enumext_level: _tl }
2634             {
2635                 columns-sep = \exp_not:v { \__enumext_store_columns_sep_ \__enumext_level: _dim }
2636             }
2637         }
2638     }
2639 }

```

(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.

```

2640 \cs_new_protected:Npn \__enumext_start_store_level:
2641 {
2642     \bool_lazy_all:nT
2643     {
2644         { \bool_if_p:N \__enumext_store_active_bool }
2645         { \bool_not_p:n { \__enumext_keyans_env_bool } }
2646         { \bool_not_p:n { \g__enumext_starred_bool } }
2647     }
2648     {
2649         \int_compare:nNnT { \__enumext_level_int } > { 1 }
2650         {
2651             \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
2652             \__enumext_store_level_open:
2653         }
2654     }
2655     \bool_lazy_all:nT
2656     {
2657         { \bool_if_p:N \__enumext_store_active_bool }

```

```

2658     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2659     { \bool_if_p:N \g__enumext_starred_bool }
2660   }
2661   {
2662     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2663     {
2664       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2665       \__enumext_store_level_open:
2666     }
2667   }
2668 }
2669 \cs_new_protected:Nn \__enumext_stop_store_level:
2670 {
2671   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2672   {
2673     \__enumext_store_level_close:
2674   }
2675 }

```

(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.

```

2676 \cs_new_protected:Nn \__enumext_before_list:
2677 {
2678   \__enumext_vspace_above:
2679   \__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.

```

2680   \__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.

```

2681   \dim_compare:nNnT
2682   { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2683   {
2684     \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2685     {
2686       \linewidth
2687       - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2688       - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2689     }

```

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

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

```

2690     \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2691     \int_gincr:N \g__enumext_minipage_stat_int
2692     \__enumext_mini_addvspace:
2693     \nointerlineskip\noindent
2694     \begin{\__enumext_mini_env*}
2695     { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2696   }
2697   \__enumext_multicols_start:
2698 }

```

(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.

```

2699 \cs_new_protected:Nn \__enumext_multicols_start:
2700 {
2701   \int_compare:nNt
2702     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2703   {
2704     \dim_compare:nNt
2705       { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2706     {
2707       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
2708       {
2709         ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
2710           + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
2711         ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
2712         - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
2713       }
2714     }
2715     \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
2716     \skip_zero:N \multicolsep
2717     \int_compare:nNt { \__enumext_level_int } > { 1 }
2718     {
2719       \dim_zero:N \columnseprule
2720     }
2721   }

```

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.

```

2721   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2722   {
2723     \__enumext_multi_addvspace:
2724   }
2725   \raggedcolumns
2726   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2727 }
2728 }

```

(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.

```

2729 \cs_new_protected:Nn \__enumext_multicols_stop:
2730 {
2731   \int_compare:nNt
2732     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2733   {
2734     \end{multicols}
2735     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2736     {
2737       \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2738     }
2739   }

```

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

```

2740   \bool_lazy_and:nnT
2741     { \bool_if_p:N \__enumext_check_ans_bool }
2742     { \bool_not_p:n { \g__enumext_starred_bool } }
2743   {
2744     \bool_gset_true:N \g__enumext_check_ans_show_bool
2745     \tl_gset:NV \g__enumext_store_name_tl \__enumext_store_name_tl
2746   }
2747 }

```

(End of definition for `\__enumext_multicols_stop:`)

`\__enumext_after_list:`

The function `\__enumext_after_list:` will check the state of the boolean variable `\__enumext_minipage_active_X_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `\__enumext_mini_env*` environment has not been closed), then close `\__enumext_mini_env*` and add the *adjusted vertical space* `\__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

2748 \cs_new_protected:Nn \__enumext_after_list:
2749 {
2750   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2751   {
2752     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2753     {
2754       \msg_warning:nn { enumext } { missing-miniright }
2755       \miniright
2756     }
2757     \int_gzero:N \g__enumext_minipage_stat_int
2758     \end{__enumext_mini_env*}
2759     \par\addvspace { \l__enumext_minipage_after_skip }
2760   }
2761   { \__enumext_multicols_stop: }

```

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

```

2762   \__enumext_after_stop_list:
2763   \__enumext_vspace_below:

```

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

```

2764   \bool_set_false:N \l__enumext_standar_bool
2765   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2766   % Quizas aquí pueda incrementar directo y luego pasar todo con \g__enumext_resume_int
2767   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2768   {
2769     \int_gset_eq:cN { g__enumext_resume_ \l__enumext_store_name_tl _int } \g__enumext_resume_
2770   }
2771   % Si se ejecuta la llave series, establecemos el valor solo cuando se ejecuta
2772   \tl_if_empty:NF \l__enumext_series_str
2773   {
2774     \int_gset_eq:cN { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
2775   }
2776   % Si ejecutamos resume=name (\l__enumext_resume_name_tl), la variable entera ya está y la gua
2777   \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
2778   {
2779     \int_gset_eq:cN { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
2780   }
2781 }

```

(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`.

```

2782 \__enumext_after_env:nn {enumext}
2783 {
2784   \int_compare:nNt { \l__enumext_level_int } = { 0 }
2785   {
2786     \bool_if:NT \g__enumext_check_ans_show_bool
2787     {
2788       \__enumext_check_ans_show:
2789     }
2790     \bool_gset_false:N \g__enumext_standar_bool
2791     \bool_gset_false:N \g__enumext_check_ans_show_bool
2792     \tl_gclear:N \g__enumext_store_name_tl
2793   }
2794 }

```

### 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.

```

2795 \NewDocumentEnvironment{keyans}{ 0{ } }
2796 {
2797   \__enumext_keyans_safe_exec:
2798   \__enumext_keyans_parse_keys:n {#1}
2799   \__enumext_before_list_v:
2800   \__enumext_start_list:nn

```

```

2801     { \tl_use:N \l__enumext_label_v_tl }
2802     {
2803       \__enumext_list_arg_two_v:
2804       \__enumext_before_keys_exec_v:
2805     }
2806     \__enumext_after_args_exec_v:
2807   }
2808   {
2809     \__enumext_keyans_check_ans:nn { item } { keyans }
2810     \__enumext_stop_list:
2811     \__enumext_after_list_v:
2812   }

```

(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.

```

2813 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2814 {
2815   \bool_if:NF \l__enumext_store_active_bool
2816   {
2817     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
2818   }
2819   \int_incr:N \l__enumext_keyans_level_int
2820   \bool_set_true:N \l__enumext_keyans_env_bool
2821   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2822   \bool_set_false:N \l__enumext_store_active_bool
2823   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2824   {
2825     \msg_error:nn { enumext } { keyans-nested }
2826   }
2827   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2828   {
2829     \msg_error:nn { enumext } { keyans-wrong-level }
2830   }
2831 }

```

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

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

```

2832 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2833 {
2834   \keys_set:nn { enumext / keyans } { #1 }
2835 }

```

(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.

```

2836 \cs_new_protected:Nn \__enumext_before_list_v:
2837 {
2838   \__enumext_vspace_above_v:
2839   \__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.

```

2840   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2841   {
2842     \dim_set:Nn \l__enumext_minipage_left_v_dim
2843     {
2844       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2845     }

```



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

```

2846         \bool_set_true:N \l__enumext_minipage_active_v_bool
2847         \int_gincr:N \g__enumext_minipage_stat_int
2848         \__enumext_keyans_mini_addvspace:
2849         \nointerlineskip\noindent
2850         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2851     }

```

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

```

2852     \__enumext_keyans_multicols_start:
2853 }

```

(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.

```

2854 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2855 {
2856     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2857     {

```

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

```

2858         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2859         {
2860             \dim_set:Nn \l__enumext_columns_sep_v_dim
2861             {
2862                 (
2863                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2864                 ) / \l__enumext_columns_v_int
2865                 - \l__enumext_listoffset_v_dim
2866             }
2867         }
2868         \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).

```

2869         \skip_zero:N \multicolsep
2870         \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.

```

2871         \bool_if:NF \l__enumext_minipage_active_v_bool
2872         {
2873             \__enumext_keyans_multi_addvspace:
2874         }
2875         \raggedcolumns
2876         \begin{multicols}{ \l__enumext_columns_v_int }
2877     }
2878 }

```

(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.

```

2879 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2880 {
2881     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2882     {
2883         \end{multicols}
2884         \bool_if:NF \l__enumext_minipage_active_v_bool
2885         {
2886             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2887         }
2888     }
2889 }

```

(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.

```

2890 \cs_new_protected:Nn \__enumext_after_list_v:
2891 {
2892   \bool_if:NTF \l__enumext_minipage_active_v_bool
2893   {
2894     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2895     {
2896       \msg_warning:nn { enumext } { missing-miniright }
2897       \miniright
2898     }
2899     \int_gzero:N \g__enumext_minipage_stat_int
2900     \end{__enumext_mini_env*}
2901     \par\addvspace{ \l__enumext_minipage_after_skip }
2902   }
2903   { \__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.

```

2904   \bool_set_false:N \l__enumext_keyans_env_bool
2905   \__enumext_after_stop_list_v:
2906   \__enumext_vspace_below_v:
2907 }

```

(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.

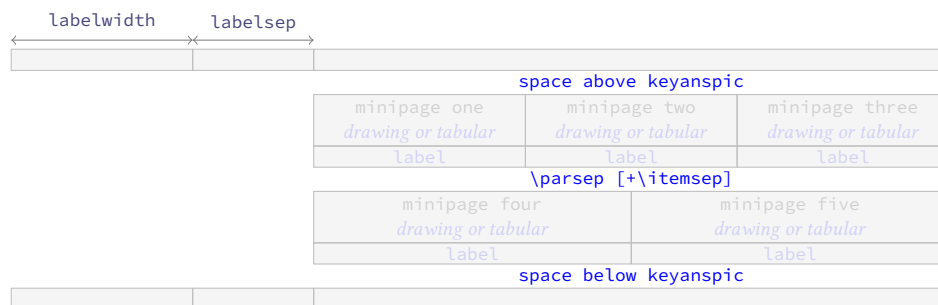


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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.

```

2908 \NewDocumentCommand \anspic { s o +m }
2909 {

```

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

```

2910   \bool_if:NF \l__enumext_store_active_bool
2911   {
2912     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2913   }
2914   \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2915   {

```

```

2916         \msg_error:nn { enumext } { keyanspic-wrong-level }
2917     }
2918     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2919     {
2920         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2921     }

```

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.

```

2922     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2923     {
2924         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2925     }
2926 }

```

(End of definition for `\anspic`. This function is documented on page 13.)

`\__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.

```

2927 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2928 {
2929     \stepcounter { enumXvi }
2930     #3 \
2931     \bool_if:nT { #1 }
2932     {
2933         \__enumext_keyans_addto_prop:n { #2 }
2934         \__enumext_keyans_store_ref:
2935         \__enumext_keyans_addto_seq:n { #2 }
2936         \bool_lazy_or:nnT
2937         { \bool_if_p:N \l__enumext_show_answer_bool }
2938         { \bool_if_p:N \l__enumext_show_position_bool }
2939         {
2940             \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2941             \__enumext_keyans_show_left:n { #2 }
2942             \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2943         }
2944     }
2945     \tl_use:N \l__enumext_label_font_style_v_tl
2946     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2947 }

```

(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.

```

2948 \NewDocumentEnvironment{keyanspic}{o}
2949 {
2950     \__enumext_keyans_pic_safe_exec:
2951     \__enumext_start_list:
2952     { }
2953     {
2954         \__enumext_keyans_pic_arg_two:
2955     }

```

We apply the “adjusted” vertical spacing above the environment

```

2956     \vspace { \l__enumext_keyans_pic_above_skip }
2957 }

```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_keyans_pic_body_seq` and placed in `minipage` environments on a single line. Finally we check if `\anspic*` has been used, set the counter to zero and apply our “adjusted” vertical space below the environment.

```

2958 {
2959     \tl_if_novalue:nTF { #1 }
2960     {
2961         \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2962     }
2963     { \__enumext_keyans_pic_do:n { #1 } }
2964     \__enumext_stop_list:
2965     \__enumext_keyans_check_ans:nn { anspic } { keyanspic }

```

```

2966     \setcounter { enumXvi } { 0 }
2967     \vspace { \l__enumext_topsep_v_skip }
2968     %\bool_set_false:N \l__enumext_store_active_bool
2969 }

```

(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.

```

2970 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2971 {
2972     \int_incr:N \l__enumext_keyans_pic_level_int
2973     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2974     {
2975         \msg_error:nn { enumext } { keyanspic-nested }
2976     }
2977 }

```

(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`.

```

2978 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2979 {
2980     \dim_compare:nNnT { #1 } < { 0pt }
2981     { \skip_set:Nn #1 { -#1 } }
2982 }

```

(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.

```

2983 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2984 {

```

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.

```

2985     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2986     \__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.

```

2987     \skip_add:Nn \parsep { \itemsep }
2988     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2989     \dim_zero:N \labelwidth
2990     \dim_zero:N \listparindent
2991     \dim_zero:N \labelsep
2992     \skip_zero:N \partopsep
2993     \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.

```

2994     \__enumext_keyans_pic_skip_abs:N \parsep
2995     \skip_set:Nn \l__enumext_keyans_pic_above_skip
2996     {
2997         \box_dp:N \strutbox
2998         + \l__enumext_topsep_v_skip
2999         - \parsep
3000     }
3001     \__enumext_item_std:w \scan_stop:
3002 }

```

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

`\__enumext_keyans_pic_do:n`  
`\__enumext_keyans_pic_do:e`

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

```

3003 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3004 {
3005   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3006 }
3007 \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.

```

3008 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3009 {
3010   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3011   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3012   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3013   \int_step_inline:nnn
3014     { \l__enumext_keyans_pic_above_int + 1 }
3015     { \l__enumext_keyans_pic_below_int }
3016     {
3017       \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3018       \centering
3019       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3020       \__enumext_endminipage:
3021     }
3022   \par
3023 }

```

(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 (*number*) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an optional first argument (*number*).

#### 10.35.1 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.

```

3024 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3025 {
3026   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3027   {
3028     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3029     {
3030       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3031       / \l__enumext_columns_vii_int
3032     }
3033   }
3034   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3035   \dim_set:Nn \l__enumext_item_width_vii_dim
3036   {
3037     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3038     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3039     - \l__enumext_labelsep_vii_dim
3040   }
3041   \dim_zero_new:N \itemwidth
3042 }

```

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

\\_\_enumext\_starred\_joined\_item\_vii:n

The function \\_\_enumext\_starred\_joined\_item\_vii:n will set the *width* of the box in which the content passed to \item(<number>) will be stored together with the value of \itemwidth.

```

3043 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3044 {
3045   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3046   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3047   {
3048     \msg_warning:nnee { enumext } { item-joined }
3049     { \int_use:N \l__enumext_joined_item_vii_int }
3050     { \int_use:N \l__enumext_columns_vii_int }
3051     \int_set:Nn \l__enumext_joined_item_vii_int
3052     {
3053       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3054     }
3055   }
3056   \int_compare:nNnT
3057   { \l__enumext_joined_item_vii_int }
3058   >
3059   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3060   {
3061     \msg_warning:nnee { enumext } { item-joined-columns }
3062     { \int_use:N \l__enumext_joined_item_vii_int }
3063     {
3064       \int_eval:n
3065       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3066     }
3067     \int_set:Nn \l__enumext_joined_item_vii_int
3068     {
3069       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3070     }
3071   }

```

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

```

3072   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3073   {
3074     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3075     \int_decr:N \l__enumext_joined_item_aux_vii_int
3076     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3077     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3078     \dim_set:Nn \l__enumext_joined_width_vii_dim
3079     {
3080       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3081       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3082         + \l__enumext_columns_sep_vii_dim
3083         ) * \l__enumext_joined_item_aux_vii_int
3084     }
3085     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3086   }
3087   {
3088     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3089     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3090   }
3091 }

```

(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.

```

3092 \cs_new_protected:Nn \__enumext_start_mini_vii:
3093 {
3094   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3095   {
3096     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3097     {
3098       \linewidth
3099       - \l__enumext_minipage_right_vii_dim
3100       - \l__enumext_minipage_hsep_vii_dim
3101     }
3102     \bool_set_true:N \l__enumext_minipage_active_vii_bool

```

```

3103     \dim_gset_eq:NN
3104     \g__enumext_minipage_right_vii_dim
3105     \l__enumext_minipage_right_vii_dim
3106     \__enumext_mini_addvspace_vii:
3107     \nointerlineskip\noindent
3108     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3109   }
3110 }

```

(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”.

```

3111 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3112 {
3113   \bool_if:NT \l__enumext_minipage_active_vii_bool
3114   {
3115     \end{__enumext_mini_env*}
3116     \hfill
3117     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3118   }
3119 }

```

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”.

```

3120 \__enumext_after_env:nn {enumext*}
3121 {
3122   \bool_if:NT \g__enumext_minipage_active_vii_bool
3123   {
3124     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3125     \par\addvspace { \g__enumext_minipage_right_skip }
3126     \bool_if:NF \g__enumext_minipage_center_vii_bool
3127     {
3128       \centering
3129     }
3130     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3131     \end{__enumext_mini_env*}
3132     \par\addvspace{ \g__enumext_minipage_after_skip }
3133   }
3134   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3135   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3136   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3137   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3138 }

```

(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.

```

3139 \NewDocumentEnvironment{enumext*}{ o }
3140 {
3141   \__enumext_current_env:
3142   \__enumext_safe_exec_vii:
3143   \__enumext_parse_keys_vii:n {#1}
3144   \__enumext_before_list_vii:
3145   \__enumext_start_store_level_vii:
3146   \__enumext_start_list:nn { }
3147   {
3148     \__enumext_list_arg_two_vii:
3149     \__enumext_before_keys_exec_vii:
3150   }
3151   \__enumext_starred_columns_set_vii:
3152   \item[] \scan_stop:
3153   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3154   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3155 }
3156 {
3157   \__enumext_stop_item_tmp_vii:

```



```

3158     \__enumext_remove_extra_parsep_vii:
3159     \__enumext_stop_list:
3160     \__enumext_stop_store_level_vii:
3161     \__enumext_after_list_vii:
3162 }

```

(End of definition for enumext\*. This function is documented on page 5.)

\\_\_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.

```

3163 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3164 {
3165     \int_incr:N \l__enumext_level_h_int
3166     \int_compare:nNt { \l__enumext_level_h_int } > { 1 }
3167     {
3168         \msg_error:nn { enumext } { nested }
3169     }
3170     \bool_set_true:N \l__enumext_starred_bool
3171     \bool_lazy_all:nT
3172     {
3173         { \bool_if_p:N \g__enumext_starred_bool }
3174         { \int_compare_p:nNt { \l__enumext_level_h_int } = { 1 } }
3175         { \int_compare_p:nNt { \l__enumext_level_int } = { 0 } }
3176     }
3177     {
3178         \typeout{[[ON-FIRST-LEVEL-ENUMEXT*-NOT-NESTED]]}
3179         \bool_set_true:N \l__enumext_starred_level_one_bool
3180     }
3181 }

```

(End of definition for \\_\_enumext\_safe\_exec\_vii:.)

\\_\_enumext\_parse\_keys\_vii:n Parse [*key = val*] for enumext\*. If the variable \l\_\_enumext\_store\_active\_bool is true it will call the function \\_\_enumext\_parse\_store\_keys\_vii:n and reprocess the keys to pass them to the storage sequence.

```

3182 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3183 {
3184     \tl_if_novalue:nF {#1}
3185     {
3186         \str_clear:N \l__enumext_series_str
3187         \keys_set:nn { enumext / enumext* } {#1}
3188         \__enumext_parse_series_resume:n {#1}
3189         \bool_if:NT \l__enumext_store_active_bool
3190         {
3191             \__enumext_parse_store_keys_vii:n {#1}
3192         }
3193     }
3194 }

```

(End of definition for \\_\_enumext\_parse\_keys\_vii:n.)

\\_\_enumext\_parse\_store\_keys\_vii:n The function \\_\_enumext\_parse\_store\_keys\_vii:n searches for the values of the columns and columns-sep keys in the optional argument in enumext\* environment as long as the starred versions of the columns\* and columns-sep\* keys are not active. The captured values are stored in the variable \l\_\_enumext\_store\_opt\_vii\_tl which is used by the function \\_\_enumext\_store\_level\_open\_vii:.

```

3195 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3196 {
3197     \bool_if:NF \l__enumext_store_columns_vii_bool
3198     {
3199         \regex_match:nnT { \b columns\b } {#1}
3200         {
3201             \int_set_eq:NN
3202             \l__enumext_store_columns_vii_int
3203             \l__enumext_columns_vii_int
3204             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3205             {
3206                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3207             }
3208         }
3209     }

```

```

3209     }
3210     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3211     {
3212         \regex_match:nnT { \b columns-sep \b } {#1}
3213         {
3214             \dim_set_eq:NN
3215             \l__enumext_store_columns_sep_vii_dim
3216             \l__enumext_columns_sep_vii_dim
3217             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3218             {
3219                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3220             }
3221         }
3222     }
3223 }

```

(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`.

```

3224 \cs_new_protected:Nn \__enumext_before_list_vii:
3225 {
3226     \__enumext_vspace_above_vii:
3227     \__enumext_check_ans_exec: % need by chek-ans
3228     \__enumext_before_args_exec_vii:
3229     \__enumext_start_mini_vii:
3230 }

```

(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.

```

3231 \cs_new_protected:Nn \__enumext_after_list_vii:
3232 {
3233     \__enumext_stop_mini_vii:
3234     \__enumext_after_stop_list_vii:
3235     \__enumext_vspace_below_vii:
3236     \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
3237     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
3238     {
3239         \int_gset_eq:cN
3240         { g__enumext_resume_ \l__enumext_store_name_tl _int }
3241         { \value{enumXvii} }
3242     }
3243     \bool_lazy_and:nnT
3244     { \bool_if_p:N \g__enumext_starred_bool }
3245     { \bool_if_p:N \l__enumext_check_ans_bool }
3246     {
3247         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
3248         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
3249     }
3250     %\bool_gset_false:N \g__enumext_starred_bool
3251     \bool_set_false:N \l__enumext_starred_bool
3252 }

```

(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`.

```

3253 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3254 {
3255     \bool_if:NT \l__enumext_store_active_bool
3256     {
3257         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }

```

```

3258         {
3259             \__enumext_store_level_open_vii:
3260         }
3261     }
3262 }
3263 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3264 {
3265     \bool_if:NT \l__enumext_store_active_bool
3266     {
3267         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3268         {
3269             \__enumext_store_level_close_vii:
3270         }
3271     }
3272 }

```

(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.

```

3273 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3274 {
3275     \__enumext_stop_item_tmp_vii:
3276     \int_incr:N \l__enumext_item_column_pos_vii_int
3277     \int_gincr:N \g__enumext_item_count_all_vii_int
3278     \__enumext_item_peek_args_vii:
3279 }

```

(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).

```

3280 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3281 {
3282     \peek_meaning:NTF (
3283     { \__enumext_joined_item_vii:w }
3284     { \__enumext_joined_item_vii:w (1) }
3285 }

```

(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.

```

3286 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3287 {
3288     \__enumext_starred_joined_item_vii:n {#1}
3289     \peek_meaning_remove:NTF *
3290     { \__enumext_starred_item_vii:w }
3291     { \__enumext_standard_item_vii:w }
3292 }

```

(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 ].

```

3293 \cs_new_protected:Npn \__enumext_standard_item_vii:w

```

```

3294 {
3295   \bool_set_false:N \l__enumext_item_starred_vii_bool
3296   \peek_meaning:NTF [
3297     {
3298       \bool_set_eq:NN
3299       \l__enumext_wrap_label_vii_bool
3300       \l__enumext_wrap_label_opt_vii_bool
3301       \__enumext_start_item_vii:w
3302     }
3303     {
3304       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3305       \legacy_if_set_true:n { @noitemarg }
3306       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3307     }
3308   }

```

(End of definition for \\_\_enumext\_standard\_item\_vii:w.)

```

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

```

The function \\_\_enumext\_starred\_item\_vii:w together with the specified auxiliary functions aux\_i:w, aux\_ii:w, and aux\_iii:w execute \item\*, \item\*[\symbol] and \item\*[\symbol][\offset].

```

3309 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3310 {
3311   \bool_set_true:N \l__enumext_item_starred_vii_bool
3312   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3313   \peek_meaning:NTF [
3314     { \__enumext_starred_item_vii_aux_i:w }
3315     { \__enumext_starred_item_vii_aux_ii:w }
3316   }
3317   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3318   {
3319     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3320     \__enumext_starred_item_vii_aux_ii:w
3321   }
3322   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3323   {
3324     \peek_meaning:NTF [
3325       { \__enumext_starred_item_vii_aux_iii:w }
3326       {
3327         \dim_set_eq:NN
3328         \l__enumext_item_symbol_sep_vii_dim
3329         \l__enumext_labelsep_vii_dim
3330         \legacy_if_set_true:n { @noitemarg }
3331         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3332       }
3333     }
3334   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3335   {
3336     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3337     \legacy_if_set_true:n { @noitemarg }
3338     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3339   }

```

(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.

```

3340 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3341 {
3342   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3343   \legacy_if:nT { @noitemarg }
3344   {
3345     \legacy_if_set_false:n { @noitemarg }
3346     \legacy_if:nT { @nmbrrlist }
3347     {

```

```

3348         \bool_if:NT \l__enumext_hyperref_bool
3349         {
3350             \legacy_if_set_true:n { @hyper@item }
3351         }
3352         \refstepcounter{enumXvii}
3353         \bool_if:NT \l__enumext_check_ans_bool
3354         {
3355             \int_gincr:N \g__enumext_count_item_number_int
3356         }
3357     }
3358 }

```

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

```

3359     \group_begin:
3360     \lrbox{ \l__enumext_item_text_vii_box }
3361     \bool_if:NF \l__enumext_footnotes_key_bool
3362     {
3363         \__enumext_renew_footnote:
3364     }
3365     \bool_if:NT \l__enumext_item_starred_vii_bool
3366     {
3367         \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3368         {
3369             \tl_gset_eq:NN
3370             \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3371         }
3372         \mode_leave_vertical:
3373         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3374         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3375         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3376         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3377     }
3378     \group_begin:
3379     \tl_use:N \l__enumext_label_font_style_vii_tl
3380     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3381     {
3382         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3383         { \__enumext_wrapper_label_vii:n {#1} }
3384     }
3385     {
3386         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3387     }
3388     \group_end:
3389     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3390     \tl_use:N \l__enumext_after_list_args_vii_tl
3391     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3392     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3393     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3394     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3395 }

```

(End of definition for `\__enumext_start_item_vii:w`.)

`\__enumext_stop_item_vii:` The function `\__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its  $\langle contents \rangle$ . Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

3396 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3397 {
3398     \__enumext_endminipage:
3399     \endlrbox
3400     \group_end:
3401     \box_set_wd:Nn \l__enumext_item_text_vii_box
3402     {
3403         \l__enumext_joined_width_vii_dim
3404         + \l__enumext_labelwidth_vii_dim

```

```

3405         + \l__enumext_labelsep_vii_dim
3406     }
3407     \int_set:Nn \hbadness { 10000 }
3408     \box_use:N \l__enumext_item_text_vii_box
3409     \bool_if:NF \l__enumext_footnotes_key_bool
3410     {
3411         \__enumext_print_footnote:
3412     }
3413     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3414     {
3415         \par\noindent
3416         \int_zero:N \l__enumext_item_column_pos_vii_int
3417     }
3418     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3419 }

```

(End of definition for \\_\_enumext\_stop\_item\_vii:.)

\\_\_enumext\_remove\_extra\_parsep\_vii:

Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

3420 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3421 {
3422     \int_compare:nNnT
3423     {
3424         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3425     }
3426     =
3427     { \c_zero_int }
3428     {
3429         \par
3430         \vspace{ -\l__enumext_itemsep_vii_skip }
3431         \int_gzero:N \g__enumext_item_count_all_vii_int
3432     }
3433 }

```

(End of definition for \\_\_enumext\_remove\_extra\_parsep\_vii:.)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the “hook” function `\__enumext_after_env:nn`.

```

3434 \__enumext_after_env:nn {enumext*}
3435 {
3436     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3437     {
3438         \bool_if:NT \g__enumext_check_ans_show_h_bool
3439         {
3440             \__enumext_check_ans_show:
3441         }
3442         \bool_gset_false:N \g__enumext_starred_bool
3443         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3444         \tl_gclear:N \g__enumext_store_name_tl
3445     }
3446 }

```

## 10.36 The keyans\* environment

### 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.

```

3447 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3448 {
3449     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3450     {
3451         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3452         {
3453             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3454             / \l__enumext_columns_viii_int
3455         }
3456     }
3457     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3458     \dim_set:Nn \l__enumext_item_width_viii_dim
3459     {

```

```

3460         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3461         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3462         - \l__enumext_labelsep_viii_dim
3463     }
3464     \dim_zero_new:N \itemwidth
3465 }

```

(End of definition for \l\_\_enumext\_starred\_columns\_set\_viii:.)

\l\_\_enumext\_starred\_joined\_item\_viii:n

The function \l\_\_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.

```

3466 \cs_new_protected:Npn \l__enumext_starred_joined_item_viii:n #1
3467 {
3468     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3469     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3470     {
3471         \msg_warning:nnee { enumext } { item-joined }
3472         { \int_use:N \l__enumext_joined_item_viii_int }
3473         { \int_use:N \l__enumext_columns_viii_int }
3474         \int_set:Nn \l__enumext_joined_item_viii_int
3475         {
3476             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3477         }
3478     }
3479     \int_compare:nNnT
3480     { \l__enumext_joined_item_viii_int }
3481     >
3482     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3483     {
3484         \msg_warning:nnee { enumext } { item-joined-columns }
3485         { \int_use:N \l__enumext_joined_item_viii_int }
3486         {
3487             \int_eval:n
3488             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3489         }
3490         \int_set:Nn \l__enumext_joined_item_viii_int
3491         {
3492             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3493         }
3494     }
3495     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3496     {
3497         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3498         \int_decr:N \l__enumext_joined_item_aux_viii_int
3499         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3500         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3501         \dim_set:Nn \l__enumext_joined_width_viii_dim
3502         {
3503             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3504             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3505                 + \l__enumext_columns_sep_viii_dim
3506                 ) * \l__enumext_joined_item_aux_viii_int
3507         }
3508         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3509     }
3510     {
3511         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3512         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3513     }
3514 }

```

(End of definition for \l\_\_enumext\_starred\_joined\_item\_viii:n.)

\l\_\_enumext\_start\_mini\_viii:

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

\l\_\_enumext\_stop\_mini\_viii:

```

3515 \cs_new_protected:Nn \l__enumext_start_mini_viii:
3516 {
3517     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3518     {
3519         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3520         {
3521             \linewidth

```

```

3522         - \l__enumext_minipage_right_viii_dim
3523         - \l__enumext_minipage_hsep_viii_dim
3524     }
3525     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3526     \dim_gset_eq:NN
3527         \g__enumext_minipage_right_viii_dim
3528         \l__enumext_minipage_right_viii_dim
3529     \__enumext_mini_addvspace_viii:
3530     \nointerlineskip\noindent
3531     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3532 }
3533 }
3534 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3535 {
3536     \bool_if:NT \l__enumext_minipage_active_viii_bool
3537     {
3538         \end{__enumext_mini_env*}
3539         \hfill
3540         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3541     }
3542 }
3543 \__enumext_after_env:nn {keyans*}
3544 {
3545     \bool_if:NT \g__enumext_minipage_active_viii_bool
3546     {
3547         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3548         \par\addvspace { \g__enumext_minipage_right_skip }
3549         \bool_if:NF \g__enumext_minipage_center_viii_bool
3550         {
3551             \centering
3552         }
3553         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3554         \end{__enumext_mini_env*}
3555         \par\addvspace{ \g__enumext_minipage_after_skip }
3556     }
3557     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3558     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3559     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3560     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3561 }

```

(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.

```

3562 \NewDocumentEnvironment{keyans*}{ o }
3563 {
3564     \__enumext_safe_exec_viii:
3565     \__enumext_parse_keys_viii:n {#1}
3566     \__enumext_before_list_viii:
3567     \__enumext_start_list:nn { }
3568     {
3569         \__enumext_list_arg_two_viii:
3570         \__enumext_before_keys_exec_viii:
3571     }
3572     \__enumext_starred_columns_set_viii:
3573     \item[] \scan_stop:
3574     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3575     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3576 }
3577 {
3578     \__enumext_stop_item_tmp_viii:
3579     \__enumext_remove_extra_parsep_viii:
3580     \__enumext_keyans_check_ans:nn { item }{ keyans* }
3581     \__enumext_stop_list:
3582     \__enumext_after_list_viii:
3583 }

```

(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.

```

3584 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3585 {
3586   \int_incr:N \l__enumext_keyans_level_h_int
3587   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3588   {
3589     \msg_error:nn { enumext } { nested }
3590   }
3591   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3592   \bool_set_false:N \l__enumext_store_active_bool
3593   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3594   {
3595     \msg_error:nn { enumext } { keyans-wrong-level }
3596   }
3597 }

```

(End of definition for `\__enumext_safe_exec_viii:`)

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

```

3598 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3599 {
3600   \tl_if_novalue:nF {#1}
3601   {
3602     \keys_set:nn { enumext / keyans* } {#1}
3603   }
3604 }

```

(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`.

```

3605 \cs_new_protected:Nn \__enumext_before_list_viii:
3606 {
3607   \__enumext_vspace_above_viii:
3608   \__enumext_before_args_exec_viii:
3609   \__enumext_start_mini_viii:
3610 }

```

(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.

```

3611 \cs_new_protected:Nn \__enumext_after_list_viii:
3612 {
3613   \__enumext_stop_mini_viii:
3614   \__enumext_after_stop_list_viii:
3615   \__enumext_vspace_below_viii:
3616 }

```

(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 (`<number>`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `<label>` next to the [`<content>`] if it is present in the `<sequence>` and `<prop list>` defined by `save-ans` key for `\item*`, `\item* [<content>]`, `\item(<number>)*` and `\item(<number>)* [<content>]` commands.

`\__enumext_start_item_tmp_viii:` First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

3617 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3618 {
3619   \__enumext_stop_item_tmp_viii:
3620   \int_incr:N \l__enumext_item_column_pos_viii_int
3621   \int_gincr:N \g__enumext_item_count_all_viii_int
3622   \__enumext_item_peek_args_viii:
3623 }

```

(End of definition for `\__enumext_start_item_tmp_viii:`)

`\__enumext_item_peek_args_viii:`

The function `\__enumext_item_peek_args_viii:` will handle the `\item(<number>)`. Look for the argument “(”, if it is present we will call the function `\__enumext_joined_item_viii:w` (`<number>`), which is in charge of joining the item’s in the same row, in case they are not present we will set the default value (1).

```
3624 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3625 {
3626   \peek_meaning:NTF (
3627     { \__enumext_joined_item_viii:w }
3628     { \__enumext_joined_item_viii:w (1) }
3629   }
```

(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`.

```
3630 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3631 {
3632   \__enumext_starred_joined_item_viii:n {#1}
3633   \peek_meaning_remove:NTF *
3634     { \__enumext_starred_item_viii:w }
3635     { \__enumext_standard_item_viii:w }
3636 }
```

(End of definition for `\__enumext_joined_item_viii:w`)

`\__enumext_standard_item_viii:w`

The function `\__enumext_standard_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[<custom>]` by means of the function `\__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]`.

```
3637 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3638 {
3639   \bool_set_false:N \l__enumext_item_starred_viii_bool
3640   \peek_meaning:NTF [
3641     {
3642       \bool_set_eq:NN
3643         \l__enumext_wrap_label_viii_bool
3644         \l__enumext_wrap_label_opt_viii_bool
3645       \__enumext_start_item_viii:w
3646     }
3647     {
3648       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3649       \legacy_if_set_true:n { @noitemarg }
3650       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3651     }
3652 }
```

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

`\__enumext_starred_item_viii:w`

The function `\__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item* [ <content> ]`.

`\__enumext_starred_item_viii_aux_i:w`

`\__enumext_starred_item_viii_aux_ii:w`

```
3653 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3654 {
3655   \bool_set_true:N \l__enumext_item_starred_viii_bool
3656   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3657   \peek_meaning:NTF [
3658     { \__enumext_starred_item_viii_aux_i:w }
3659     { \__enumext_starred_item_viii_aux_ii:w }
3660 }
```

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

```

3661 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3662 {
3663   \tl_clear:N \__enumext_store_keyans_label_tl
3664   \tl_if_no_value:nF { #1 }
3665   {
3666     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3667     {
3668       \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
3669       \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
3670     }
3671     \tl_set:Ne \__enumext_keyans_item_opt_tl { #1 }
3672   }
3673   \__enumext_starred_item_viii_aux_ii:w
3674 }
3675 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3676 {
3677   \legacy_if_set_true:n { @noitemarg }
3678   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3679 }

```

(End of definition for `\__enumext_starred_item_viii:w`, `\__enumext_starred_item_viii_aux_i:w`, and `\__enumext_starred_item_viii_aux_ii:w`.)

`\__enumext_starred_item_exec:`

The function `\__enumext_starred_item_exec:` will be in charge of storing the current *(label)* for `\item*` followed by the *(content)* for `\item*[(content)]` if present in the *(sequence)* and *(prop list)* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

3680 \cs_new_protected:Nn \__enumext_starred_item_exec:
3681 {
3682   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3683   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3684   \__enumext_keyans_store_ref:
3685   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3686   \__enumext_keyans_addto_seq_link:
3687   \bool_if:NT \__enumext_show_answer_bool
3688   {
3689     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3690   }
3691   \bool_if:NT \__enumext_show_position_bool
3692   {
3693     \tl_set:Ne \__enumext_mark_answer_sym_tl
3694     {
3695       \group_begin:
3696       \exp_not:N \normalfont
3697       \exp_not:N \footnotesize [ \int_eval:n
3698       {
3699         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3700       }
3701       ]
3702       \group_end:
3703     }
3704     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3705   }
3706 }

```

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

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

`\__enumext_start_item_viii:w`

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

```

3707 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3708 {
3709   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3710   \legacy_if:nT { @noitemarg }
3711   {
3712     \legacy_if_set_false:n { @noitemarg }
3713     \legacy_if:nT { @nmbrrlist }
3714   }

```

```

3715         \bool_if:NT \l__enumext_hyperref_bool
3716         {
3717             \legacy_if_set_true:n { @hyper@item }
3718         }
3719         \refstepcounter{enumXviii}
3720     }
3721 }

```

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

```

3722 \group_begin:
3723 \lrbox{ \l__enumext_item_text_viii_box }
3724 \bool_if:NF \l__enumext_footnotes_key_bool
3725 {
3726     \__enumext_renew_footnote:
3727 }
3728 \bool_if:NT \l__enumext_item_starred_viii_bool
3729 {
3730     \__enumext_starred_item_exec:
3731 }
3732 \group_begin:
3733 \tl_use:N \l__enumext_label_font_style_viii_tl
3734 \bool_if:NTF \l__enumext_wrap_label_viii_bool
3735 {
3736     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3737     { \__enumext_wrapper_label_viii:n {#1} }
3738 }
3739 {
3740     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3741 }
3742 \group_end:
3743 \skip_horizontal:N \l__enumext_labelsep_viii_dim
3744 \tl_use:N \l__enumext_after_list_args_viii_tl
3745 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3746 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3747 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3748 \bool_if:NT \l__enumext_item_starred_viii_bool
3749 {
3750     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3751     \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3752 }
3753 {
3754     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3755 }
3756 }

```

(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.

```

3757 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3758 {
3759     \__enumext_endminipage:
3760     \endlrbox
3761     \group_end:
3762     \box_set_wd:Nn \l__enumext_item_text_viii_box
3763     {
3764         \l__enumext_joined_width_viii_dim
3765         + \l__enumext_labelwidth_viii_dim
3766         + \l__enumext_labelsep_viii_dim
3767     }
3768     \int_set:Nn \hbadness { 10000 }
3769     \box_use:N \l__enumext_item_text_viii_box
3770     \bool_if:NF \l__enumext_footnotes_key_bool
3771     {
3772         \__enumext_print_footnote:
3773     }
3774     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3775     {
3776         \par\noindent
3777         \int_zero:N \l__enumext_item_column_pos_viii_int

```

```

3778     }
3779     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3780 }

```

(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.

```

3781 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3782 {
3783   \int_compare:nNnT
3784   {
3785     \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3786   }
3787   =
3788   { \c_zero_int }
3789   {
3790     \par
3791     \vspace{ -\l__enumext_itemsep_viii_skip }
3792     \int_gzero:N \g__enumext_item_count_all_viii_int
3793   }
3794 }

```

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

```

3795 \NewDocumentCommand \getkeyans { m }
3796 {
3797   \exp_args:Ne \__enumext_getkeyans_aux:n
3798   { \tl_to_str:e { \text_expand:n {#1} } }
3799 }

```

(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  $\langle argument \rangle$  using “:”. If “:” is omitted it will return an error.

```

3800 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3801 {
3802   \str_if_in:nnTF {#1} { : }
3803   {
3804     \use:e
3805     {
3806       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3807       { {##1} {##2} }
3808     }
3809     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3810   }
3811   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3812 }

```

(End of definition for \\_\_enumext\_getkeyans\_aux:n.)

\\_\_enumext\_getkeyans:nn

The internal function `\__enumext_getkeyans:nn` will check for the existence of the  $\langle prop\ list \rangle$ , if it does not exist it will return an error message, then it will fetch the content specified by the second  $\langle argument \rangle$  from  $\langle prop\ list \rangle$ .

```

3813 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3814 {
3815   \prop_if_exist:cF { g__enumext_#1_prop }
3816   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3817   \group_begin:
3818     \prop_item:cn { g__enumext_#1_prop }{#2}
3819   \group_end:
3820 }

```

(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`.

```

3821 \keys_define:nn { keyanskey / print }
3822 {
3823   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3824                 {
3825                   \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3826                 },
3827   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3828   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3829                 {
3830                   \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3831                 },
3832   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3833   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3834                 {
3835                   \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3836                 },
3837   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3838   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3839                 {
3840                   \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3841                 },
3842   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3843   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3844                 {
3845                   \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3846                 },
3847   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3848 }

```

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

```

3849 \NewDocumentCommand \printkeyans { s O{} m }
3850 {
3851   \group_begin:
3852     \tl_use:N \l__enumext_print_keyans_i_tl
3853     \tl_use:N \l__enumext_print_keyans_ii_tl
3854     \tl_use:N \l__enumext_print_keyans_iii_tl
3855     \tl_use:N \l__enumext_print_keyans_iv_tl
3856     \tl_use:N \l__enumext_print_keyans_vii_tl
3857     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3858   \group_end:
3859 }

```

(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

```

3860 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3861 {
3862   \seq_if_exist:cTF { g__enumext_#3_seq }
3863   {
3864     \seq_if_empty:cF { g__enumext_#3_seq }
3865     {
3866       %%\seq_show:c { g__enumext_#3_seq }
3867       \bool_if:nTF {#1}
3868       {
3869         \begin{enumext*}[#2]
3870         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3871         \end{enumext*}
3872       }
3873     }

```

```

3874         \begin{enumext} [#2]
3875         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3876         \end{enumext}
3877     }
3878 }
3879 }
3880 {
3881     \msg_error:nnn { enumext } { undefined-storage-anskey } { #3 }
3882 }
3883 }

```

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

### 10.39 The command `\setenumext`

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

```

3884 \keys_define:nn { enumext / meta-families }
3885 {
3886     level-1 .code:n = { \keys_set:nn { enumext / level-1 } { #1 } },
3887     level-2 .code:n = { \keys_set:nn { enumext / level-2 } { #1 } },
3888     level-3 .code:n = { \keys_set:nn { enumext / level-3 } { #1 } },
3889     level-4 .code:n = { \keys_set:nn { enumext / level-4 } { #1 } },
3890     keyans .code:n = { \keys_set:nn { enumext / keyans } { #1 } },
3891     enumext* .code:n = { \keys_set:nn { enumext / enumext* } { #1 } },
3892     keyans* .code:n = { \keys_set:nn { enumext / keyans* } { #1 } },
3893     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = { #1 } } },
3894     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = { #1 } } },
3895     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = { #1 } } },
3896     print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = { #1 } } },
3897     print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = { #1 } } },
3898     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
3899 }

```

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

```

3900 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3901 {
3902     level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3903     keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3904 }

```

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

```

3905 \NewDocumentCommand \setenumext { o +m }
3906 {
3907     \tl_if_novalue:nTF { #1 }
3908     {
3909         \seq_map_inline:Nn \c__enumext_all_families_seq
3910     }
3911     {
3912         \seq_clear:N \l__enumext_setkey_tmpa_seq
3913         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq { #1 }
3914         \int_set:Nn \l__enumext_setkey_tmpa_int
3915         {
3916             \seq_count:N \l__enumext_setkey_tmpb_seq
3917         }
3918         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3919         {
3920             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3921             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3922             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3923             {
3924                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3925             }
3926         }
3927         {
3928             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n { #1 } }
3929         }
3930         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3931         { \seq_map_inline:Nn \c__enumext_all_families_seq }
3932         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3933     }
3934     {
3935         \keys_set:nn { enumext / meta-families } { ##1 = { #2 } }

```

```

3936     }
3937 }

```

(End of definition for `\setenumext`. This function is documented on page 6.)

```

\__enumext_set_parse:n
\__enumext_set_error:nn

```

Internal functions used by the `\setenumext` command.

```

3938 \cs_new_protected:Npn \__enumext_set_parse:n #1
3939 {
3940   \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3941   \int_step_inline:nnn { 0 } { 4 } {%<- max level
3942     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3943   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3944     {
3945       \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
3946         { \tl_trim_spaces:n {#1} }
3947     }
3948     { \__enumext_set_error:nn {#1} { } }
3949   }
3950 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3951 { \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.

```

3952 \msg_new:nnn { enumext } { package-load }
3953 {
3954   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3955 }
3956 \msg_new:nnn { enumext } { package-not-load }
3957 {
3958   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3959 }
3960 \msg_new:nnn { enumext } { package-load-foot }
3961 {
3962   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3963 }

```

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

```

3964 \msg_new:nnn { enumext } { counters }
3965 {
3966   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
3967   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3968 }

```

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

```

3969 \msg_new:nnn { enumext } { invalid-key }
3970 {
3971   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3972 }
3973 \msg_new:nnn { enumext } { unknown-key-family }
3974 {
3975   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3976 }

```

Messages used in length calculation.

```

3977 \msg_new:nnn { enumext } { width-negative }
3978 {
3979   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3980   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
3981 }
3982 \msg_new:nnn { enumext } { width-zero }
3983 {
3984   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3985   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
3986 }

```



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

```

3987 \msg_new:nnn { enumext } { list-lengths }
3988 {
3989   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
3990   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3991   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3992   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3993   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3994   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3995   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3996   \__enumext_show_length:nnn { skip } { topsep } {#1}
3997   \__enumext_show_length:nnn { skip } { parsep } {#1}
3998   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3999   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4000   *****
4001 }

```

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

```

4002 \msg_new:nnn { enumext } { list-lengths-not-nested }
4003 {
4004   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4005   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4006   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4007   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4008   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4009   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4010   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4011   \__enumext_show_length:nnn { skip } { topsep } {#1}
4012   \__enumext_show_length:nnn { skip } { parsep } {#1}
4013   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4014   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4015   *****
4016 }

```

Messages used by `save-ans` key.

```

4017 \msg_new:nnn { enumext } { save-ans-empty }
4018 {
4019   The ~ 'save-ans' ~ key ~ cannot ~ be ~ empty~ in ~ '#1'. ~ \msg_line_context:.
4020 }
4021 \msg_new:nnn { enumext } { save-ans-nested }
4022 {
4023   The ~ 'save-ans' ~ key ~ cannot ~ be ~ used ~ in ~ nested ~ '#1'. ~ \msg_line_context:.
4024 }

```

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

```

4025 \msg_new:nnn { enumext } { items-same-answer }
4026 {
4027   *****~Checking~answers~on~'#1'~OK~*****\\
4028   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4029   *****
4030   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4031 }
4032 \msg_new:nnn { enumext } { item-different-answer }
4033 {
4034   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4035   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4036 }

```

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

```

4037 \msg_new:nnn { enumext } { missing-starred }
4038 {
4039   Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4040 }

```

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

```

4041 \msg_new:nnn { enumext } { list-too-deep }
4042 {
4043   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4044   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4045 }

```

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

```

4046 \msg_new:nnn { enumext } { anskey-wrong-place }
4047 {
4048   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4049   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4050 }
4051 \msg_new:nnn { enumext } { anspic-wrong-place }
4052 {
4053   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4054   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4055 }
4056 \msg_new:nnn { enumext } { command-wrong-place }
4057 {
4058   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4059   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4060 }

```

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

```

4061 \msg_new:nnn { enumext } { keyans-nested }
4062 {
4063   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4064 }
4065 \msg_new:nnn { enumext } { keyans-wrong-level }
4066 {
4067   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4068   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4069 }
4070 \msg_new:nnn { enumext } { wrong-place }
4071 {
4072   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4073   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4074 }
4075 \msg_new:nnn { enumext } { keyanspic-nested }
4076 {
4077   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4078 }
4079 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4080 {
4081   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4082   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4083 }

```

Messages used by `\getkeyans` command.

```

4084 \msg_new:nnn { enumext } { undefined-storage-anskey }
4085 {
4086   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4087 }

```

Messages used by `\miniright` command.

```

4088 \msg_new:nnn { enumext } { missing-miniright }
4089 {
4090   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
4091   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4092 }
4093 \msg_new:nnn { enumext } { wrong-miniright-place }
4094 {
4095   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4096   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4097 }
4098 \msg_new:nnn { enumext } { wrong-miniright-use }
4099 {
4100   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4101   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4102 }

```

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

```

4103 \msg_new:nnn { enumext } { nested }
4104 {
4105   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4106 }
4107 \msg_new:nnn { enumext } { item-joined }
4108 {
4109   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.

```

```
4110     }  
4111 \msg_new:nnn { enumext } { item-joined-columns }  
4112 {  
4113     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4114 }
```

### 10.41 Finish package

Finish package implementation.

```
4115 \file_input_stop:  
4116 </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>	423
<code>\+</code>	196
<code>\-</code>	196
<code>\\</code>	204, 2930, 3966, 3979, 3984, 3989, 4004, 4027, 4028, 4043, 4048, 4053, 4058, 4067, 4072, 4081, 4090, 4095, 4100
A	
<code>above</code>	<u>1234</u>
<code>above*</code>	<u>1234</u>
<code>\addvspace</code>	881, 909, 1032, 1111, 1174, 1180, 1208, 1225, 2737, 2759, 2886, 2901, 3125, 3132, 3548, 3555
<code>after</code>	<u>719</u>
<code>align</code>	<u>377</u>
<code>\Alph</code>	31, 35
<code>\Alpha</code>	329, 506, 524, 537, 3842
<code>\alph</code>	31, 35
<code>\alpha</code>	330, 504, 3832
<code>\anskey</code>	11, 61, 1862
<code>\anspic</code>	13, 83, <u>2908</u>
<code>\arabic</code>	31, 32
<code>\arabic</code>	328, 503, 523, 3827, 3847
B	
<code>\b</code>	2615, 2628, 3199, 3212
<code>\baselineskip</code>	42
<code>\baselineskip</code>	1822, 1830
<code>before</code>	<u>719</u>
<code>before*</code>	<u>719</u>
<code>below</code>	<u>1234</u>
<code>below*</code>	<u>1234</u>
bool commands:	
<code>\bool_gset_false:N</code>	2790, 2791, 3134, 3250, 3442, 3443, 3557
<code>\bool_gset_true:N</code>	224, 235, 823, 2744, 3117, 3135, 3247, 3540, 3558
<code>\bool_if:NTF</code>	269, 281, 298, 1256, 1270, 1283, 1294, 1305, 1316, 1327, 1338, 1389, 1399, 1660, 1670, 1784, 1808, 1815, 1843, 1874, 1887, 1889, 1900, 1920, 2045, 2056, 2060, 2099, 2114, 2187, 2206, 2210, 2323, 2353, 2426, 2442, 2503, 2513, 2543, 2548, 2605, 2613, 2626, 2671, 2721, 2735, 2750, 2786, 2815, 2871, 2884, 2892, 2910, 3113, 3122, 3126, 3189, 3197, 3210, 3255, 3265, 3348, 3353, 3361, 3365, 3380, 3409, 3438, 3536, 3545, 3549, 3687, 3691, 3715, 3724, 3728, 3734, 3748, 3770
<code>\bool_if:nTF</code>	1209, 1226, 1928, 2364, 2398, 2462, 2931, 3867
<code>\bool_if_p:N</code>	1476, 1487, 1519, 1520, 1556, 1557, 1595, 1596, 1639, 1911, 1954, 1955, 1979, 1988, 1989, 2001, 2017, 2173, 2174, 2220, 2221, 2582, 2644, 2657, 2659, 2741, 2937, 2938, 3173, 3244, 3245
<code>\bool_lazy_all:nTF</code>	1474, 1485, 1637, 1977, 1986, 1999, 2015, 2580, 2642, 2655, 3171
<code>\bool_lazy_and:nnTF</code>	220, 231, 1518, 1555, 1594, 1910, 1953, 2172, 2740, 3243
<code>\bool_lazy_or:nnTF</code>	2219, 2936
<code>\bool_new:N</code>	25, 26, 27, 28, 29, 30, 31, 48, 69, 74, 75, 80, 81, 84, 104, 106, 108, 111, 112, 121, 122, 123, 124, 133, 134, 159, 170, 172, 1530
<code>\bool_not_p:n</code>	221, 232, 1912, 2004, 2019, 2645, 2646, 2658, 2742
<code>\bool_set_eq:NN</code>	2331, 2379, 3298, 3642
<code>\bool_set_false:N</code>	278, 1457, 1621, 1622, 2764, 2822, 2904, 2968, 2985, 3251, 3295, 3592, 3639
<code>\bool_set_true:N</code>	260, 264, 370, 647, 1240, 1245, 1417, 1418, 1544, 1746, 1753, 2327, 2357, 2375, 2387, 2579, 2588, 2651, 2664, 2690, 2820, 2846, 3102, 3170, 3179, 3304, 3311, 3312, 3525, 3648, 3655, 3656
box commands:	
<code>\box_dp:N</code>	928, 932, 936, 947, 951, 962, 971, 977, 987, 1000, 1006, 1012, 1043, 1044, 1045, 1048, 1058, 1062, 1071, 1078, 1083, 1091, 1120, 1121, 1124, 1131, 1144, 1152, 1158, 1166, 2997
<code>\box_new:N</code>	45, 165
<code>\box_set_wd:Nn</code>	3401, 3762
<code>\box_use:N</code>	3408, 3769
<code>\box_wd:N</code>	336
C	
<code>\c</code>	423, 424, 547, 549, 561, 563
<code>\cB</code>	424
<code>\cE</code>	424
<code>\centering</code>	1211, 1228, 3018, 3128, 3551
<code>check-ans</code>	<u>1614</u>
Document class:	
<code>article</code>	36
clist commands:	
<code>\clist_const:Nn</code>	177
<code>\clist_map_function:nN</code>	3005
<code>\clist_map_inline:Nn</code>	376, 589, 652, 718, 733, 814, 1250
<code>\clist_map_inline:nn</code>	36, 53, 59, 71, 83, 110, 142, 156, 176, 212, 401, 418, 657, 829, 1627, 1723, 1741, 1762, 1974, 2108, 2281, 2491, 2494, 2520, 2530, 2533, 2553
<code>\columnbreak</code>	62
<code>\columnbreak</code>	1914
<code>columns</code>	<u>798</u>
<code>columns*</code>	<u>1742</u>
<code>columns-sep</code>	<u>798</u>
<code>columns-sep*</code>	<u>1742</u>
<code>\columnsep</code>	78, 82
<code>\columnsep</code>	2715, 2868
<code>\columnseprule</code>	78, 82
<code>\columnseprule</code>	2719, 2870
Commands provide by enumext:	
<code>\anskey</code>	24, 25, 52, 56, 59–66, 68, 77, 90, 101, 102, 106
<code>\anspic*</code>	24, 65–68, 83, 84, 86, 101, 102
<code>\anspic</code>	59, 60, 83–85, 106
<code>\getkeyans</code>	59, 101, 106
<code>\item*</code>	24, 59, 60, 65–68, 70, 71, 92, 98, 99, 101, 102
<code>\itemwidth</code>	86, 87, 94, 95
<code>\item</code>	70, 71, 87, 91, 92, 95, 97, 98
<code>\miniright</code>	24, 40, 48, 49, 78, 79, 82, 83, 106
<code>\printkeyans</code>	25, 60, 102
<code>\setenumext</code>	24, 103, 104
Counters defined by enumext:	
<code>enumXiii</code>	23, 30
<code>enumXii</code>	23, 30

enumXiv ..... 23, 30  
enumXi ..... 23, 30  
enumXviii ..... 23, 30  
enumXvii ..... 23, 30, 92  
enumXvi ..... 23, 30  
enumXv ..... 23, 30  
cs commands:  
\cs\_generate\_variant:Nn 338, 354, 553, 569, 1767,  
1776, 1781, 1861, 2481, 3007  
\cs\_if\_exist:NTF ..... 308  
\cs\_new:Nn ..... 190  
\cs\_new:Npn ..... 200, 1424, 1433, 1442  
\cs\_new\_eq:NN 244, 245, 246, 250, 251, 283, 284, 287,  
288  
\cs\_new\_protected:Nn . 214, 255, 419, 439, 471, 734,  
738, 742, 746, 750, 754, 758, 762, 766, 770, 774, 778,  
782, 786, 790, 794, 830, 842, 866, 883, 894, 918, 993,  
1017, 1034, 1096, 1113, 1135, 1170, 1176, 1251, 1265,  
1279, 1290, 1301, 1312, 1323, 1334, 1387, 1397, 1407,  
1516, 1568, 1592, 1606, 1632, 1668, 1675, 1782, 1806,  
1813, 1841, 1848, 1965, 2097, 2112, 2140, 2170, 2215,  
2227, 2234, 2286, 2290, 2309, 2360, 2394, 2410, 2420,  
2436, 2574, 2640, 2669, 2676, 2699, 2729, 2748, 2813,  
2836, 2854, 2879, 2890, 2927, 2970, 2983, 3003, 3008,  
3024, 3092, 3111, 3163, 3224, 3231, 3253, 3263, 3280,  
3420, 3447, 3515, 3534, 3584, 3605, 3611, 3624, 3680,  
3781  
\cs\_new\_protected:Npn 182, 186, 291, 306, 323, 333,  
339, 427, 446, 540, 554, 1198, 1217, 1365, 1376, 1455,  
1472, 1497, 1532, 1576, 1768, 1777, 1897, 2042, 2054,  
2076, 2150, 2192, 2200, 2319, 2337, 2371, 2383, 2450,  
2484, 2523, 2591, 2611, 2832, 2978, 3043, 3182, 3195,  
3286, 3293, 3309, 3317, 3322, 3334, 3466, 3598, 3630,  
3637, 3653, 3661, 3675, 3800, 3813, 3860, 3938, 3950  
\cs\_new\_protected\_nopar:Nn . . . 3273, 3396, 3617,  
3757  
\cs\_new\_protected\_nopar:Npn . . . . . 3340, 3707  
\cs\_set:Nn ..... 2047  
\cs\_set:Npn ..... 1975, 2013, 3806  
\cs\_set\_eq:NN . . 3153, 3154, 3342, 3574, 3575, 3709  
\cs\_set\_protected:Nn . . . . 206, 658, 674, 686, 698  
\cs\_set\_protected:Npn 32, 46, 54, 66, 72, 100, 138,  
150, 157, 208, 355, 377, 406, 487, 507, 570, 590, 634,  
653, 710, 719, 798, 815, 1234, 1614, 1688, 1724, 1742,  
1967, 2101, 2270, 2482, 2521  
\cs\_to\_str:N ..... 325, 348

## D

\d ..... 196  
\DeclareDocumentEnvironment ..... 911  
dim commands:  
\dim\_abs:n ..... 2455, 2460  
\dim\_add:Nn ..... 2988  
\dim\_compare:nNnTF . 660, 676, 688, 700, 1200, 1219,  
2452, 2457, 2463, 2469, 2471, 2473, 2681, 2704, 2840,  
2858, 2980, 3026, 3094, 3449, 3517  
\dim\_compare:nTF ..... 1938  
\dim\_gset\_eq:NN ..... 3103, 3526  
\dim\_gzero:N ..... 3137, 3560  
\dim\_new:N 42, 49, 50, 51, 68, 94, 107, 117, 166, 167, 173  
\dim\_set:Nn . . 336, 648, 1754, 2351, 2455, 2460, 2462,  
2465, 2466, 2470, 2472, 2475, 2476, 2478, 2684, 2707,  
2842, 2860, 3010, 3028, 3035, 3078, 3096, 3336, 3451,  
3458, 3501, 3519  
\dim\_set\_eq:NN 494, 514, 530, 534, 2346, 2493, 2532,  
2630, 2715, 2868, 3085, 3088, 3089, 3214, 3327, 3508,

3511, 3512  
\dim\_use:N 661, 669, 1201, 1207, 1851, 1854, 1859, 2415,  
2417, 2682, 2687, 2688, 2695, 2705, 2709, 2710, 2712  
\dim\_zero:N ..... 2719, 2870, 2989, 2990, 2991  
\dim\_zero\_new:N ..... 3041, 3464  
\c\_zero\_dim 663, 677, 689, 701, 1201, 1219, 1940, 2452,  
2457, 2463, 2470, 2682, 2705, 2840, 2858, 3026, 3094,  
3449, 3517

## E

\end . . 1204, 1222, 1810, 1845, 2734, 2758, 2883, 2900, 3115,  
3131, 3538, 3554, 3871, 3876  
\endlist ..... 28  
\endlist ..... 245  
\endlrbox ..... 3399, 3760  
\endminipage ..... 28  
\endminipage ..... 251  
enumext ..... 5, 2554  
enumext internal commands:

\g\_enumext\_ \\_enumext\_store\_name\_tl  
\\_prop ..... 68  
\\_\_enumext\_add\_pre\_parsep: . . . 41, 840, 842, 842  
\\_\_enumext\_after\_args\_exec: . 39, 734, 746, 2567  
\\_\_enumext\_after\_args\_exec\_v: . 39, 40, 750, 762,  
2806  
\\_\_enumext\_after\_args\_exec\_vii: . . . 766, 790  
\\_\_enumext\_after\_args\_exec\_viii: . . . . . 794  
\\_\_enumext\_after\_env:n ..... 79  
\\_\_enumext\_after\_env:nn . . 80, 94, 186, 186, 2782,  
3120, 3434, 3543  
\\_\_enumext\_after\_hyperref: . . . 29, 253, 255, 255  
\\_\_enumext\_after\_list: 79, 90, 97, 2572, 2748, 2748  
\l\_enumext\_after\_list\_args\_v\_tl . . . . . 764  
\l\_enumext\_after\_list\_args\_vii\_tl 792, 3390  
\l\_enumext\_after\_list\_args\_viii\_tl 796, 3744  
\\_\_enumext\_after\_list\_v: . . 83, 2811, 2890, 2890  
\\_\_enumext\_after\_list\_vii: . . 3161, 3231, 3231  
\\_\_enumext\_after\_list\_viii: . . 3582, 3611, 3611  
\\_\_enumext\_after\_star\_env:nn ..... 88  
\\_\_enumext\_after\_stop\_list: . . . 38, 40, 734, 742,  
2762  
\\_\_enumext\_after\_stop\_list\_v: 39, 750, 758, 2905  
\l\_enumext\_after\_stop\_list\_v\_tl . . . . . 760  
\\_\_enumext\_after\_stop\_list\_vii: 766, 782, 3234  
\l\_enumext\_after\_stop\_list\_vii\_tl . . . 784  
\\_\_enumext\_after\_stop\_list\_viii: . 786, 3614  
\l\_enumext\_after\_stop\_list\_viii\_tl . . . 788  
\l\_enumext\_align\_label\_vii\_str . . 3382, 3386  
\l\_enumext\_align\_label\_viii\_str . 3736, 3740  
\l\_enumext\_align\_label\_X\_str ..... 157  
\c\_enumext\_all\_envs\_clist . . 177, 376, 589, 652,  
718, 733, 814, 1250  
\c\_enumext\_all\_families\_seq . . 103, 3900, 3909,  
3931  
\\_\_enumext\_anskey\_wrapper:n . . . . . 1692, 2052  
\\_\_enumext\_at\_begin\_document:n . . 28, 182, 182,  
242, 248  
\\_\_enumext\_before\_args\_exec: 38, 734, 734, 2679  
\\_\_enumext\_before\_args\_exec\_v: . . 39, 750, 750,  
2839  
\\_\_enumext\_before\_args\_exec\_vii: . . 766, 766,  
3228  
\\_\_enumext\_before\_args\_exec\_viii: 770, 3608  
\\_\_enumext\_before\_keys\_exec: 38, 734, 738, 2565

`\__enumext_before_keys_exec_v:` .. 39, [750](#), [754](#), [2804](#)  
`\__enumext_before_keys_exec_vii` ..... [766](#)  
`\__enumext_before_keys_exec_vii:` 39, [774](#), [3149](#)  
`\__enumext_before_keys_exec_viii:` .. 39, [778](#), [3570](#)  
`\__enumext_before_list:` ... [78](#), [2559](#), [2676](#), [2676](#)  
`\__enumext_before_list_v:` . [81](#), [2799](#), [2836](#), [2836](#)  
`\__enumext_before_list_vii:` 90, [3144](#), [3224](#), [3224](#)  
`\__enumext_before_list_viii:` .. 97, [3566](#), [3605](#), [3605](#)  
`\l__enumext_before_no_starred_key_v_tl` [756](#)  
`\l__enumext_before_no_starred_key_vii_-tl` ..... [776](#)  
`\l__enumext_before_no_starred_key_viii_-tl` ..... [780](#)  
`\l__enumext_before_starred_key_v_tl` ... [752](#)  
`\l__enumext_before_starred_key_vii_tl` . [768](#)  
`\l__enumext_before_starred_key_viii_tl` [772](#)  
`\__enumext_calc_hspace:NNNNNN` [73](#), [2450](#), [2450](#), [2481](#), [2486](#), [2525](#)  
`\l__enumext_check_ans_bool` ... [70](#), [71](#), [121](#), [1618](#), [1622](#), [1670](#), [1889](#), [2187](#), [2323](#), [2353](#), [2741](#), [3245](#), [3353](#)  
`\__enumext_check_ans_exec:` .. 57, [78](#), [1668](#), [1668](#), [2680](#), [3227](#)  
`\g__enumext_check_ans_item_tl` .. 67, [121](#), [2186](#), [2194](#), [2198](#)  
`\__enumext_check_ans_set:` . 57, [1632](#), [1632](#), [1672](#)  
`\__enumext_check_ans_show:` 58, [1675](#), [1675](#), [2788](#), [3440](#)  
`\g__enumext_check_ans_show_bool` [79](#), [121](#), [2744](#), [2786](#), [2791](#)  
`\g__enumext_check_ans_show_h_bool` [121](#), [3247](#), [3438](#), [3443](#)  
`\l__enumext_columns_sep_v_dim` [2858](#), [2860](#), [2868](#)  
`\l__enumext_columns_sep_vii_dim` .. 3026, [3028](#), [3037](#), [3082](#), [3216](#), [3418](#)  
`\l__enumext_columns_sep_viii_dim` . 3449, [3451](#), [3460](#), [3505](#), [3779](#)  
`\l__enumext_columns_v_int` 1039, [2856](#), [2864](#), [2876](#), [2881](#)  
`\l__enumext_columns_vii_int` .. 3031, [3034](#), [3038](#), [3046](#), [3050](#), [3053](#), [3059](#), [3065](#), [3069](#), [3203](#), [3413](#), [3424](#)  
`\l__enumext_columns_viii_int` . 3454, [3457](#), [3461](#), [3469](#), [3473](#), [3476](#), [3482](#), [3488](#), [3492](#), [3774](#), [3785](#)  
`\g__enumext_count_item_anskey_int` .. 67, [121](#), [1678](#), [1686](#), [1891](#), [2189](#)  
`\g__enumext_count_item_number_int` [121](#), [1643](#), [1648](#), [1651](#), [1654](#), [1662](#), [1678](#), [1685](#), [2325](#), [2355](#), [3355](#)  
`\g__enumext_count_item_with_ans_int` .... 62  
`\l__enumext_counter_i_tl` ..... [32](#), [315](#)  
`\l__enumext_counter_ii_tl` ..... [32](#), [316](#)  
`\l__enumext_counter_iii_tl` ..... [32](#), [317](#)  
`\l__enumext_counter_iv_tl` ..... [32](#), [318](#)  
`\l__enumext_counter_style_for_ref_vii_-tl` ..... [454](#), [464](#), [475](#), [477](#)  
`\l__enumext_counter_style_for_ref_viii_-tl` ..... [481](#), [483](#)  
`\l__enumext_counter_style_for_ref_X_tl` [146](#)  
`\c__enumext_counter_style_tl` .... [32](#), [146](#), [421](#)  
`\g__enumext_counter_styles_tl` . 23, [31](#), [42](#), [326](#), [344](#)  
`\l__enumext_counter_v_tl` ..... [32](#), [319](#)  
`\l__enumext_counter_vi_tl` ..... [32](#), [320](#)  
`\l__enumext_counter_vii_tl` ..... [32](#), [321](#), [451](#)  
`\l__enumext_counter_viii_tl` ..... [32](#), [322](#), [461](#)  
`\__enumext_current_env:` 27, [214](#), [214](#), [2556](#), [3141](#)  
`\l__enumext_current_widest_dim` 23, [42](#), [350](#), [495](#), [515](#), [531](#), [535](#)  
`\__enumext_default_item:n` ... [2319](#), [2319](#), [2368](#)  
`\__enumext_define_counters:Nn` 23, [306](#), [306](#), [315](#), [316](#), [317](#), [318](#), [319](#), [320](#), [321](#), [322](#)  
`\__enumext_endminipage:` . 28, [248](#), [251](#), [917](#), [3020](#), [3398](#), [3759](#)  
`\__enumext_fake_item:` ..... [658](#), [658](#), [2512](#)  
`\l__enumext_fake_item_indent_v_dim` [677](#), [682](#)  
`\l__enumext_fake_item_indent_v_tl` [679](#), [2376](#), [2380](#), [2388](#)  
`\l__enumext_fake_item_indent_vii_dim` [689](#), [694](#)  
`\l__enumext_fake_item_indent_vii_tl` [691](#), [3394](#)  
`\l__enumext_fake_item_indent_viii_dim` . 701, [706](#), [3751](#)  
`\l__enumext_fake_item_indent_viii_tl` .. 703, [3750](#), [3754](#)  
`\l__enumext_fake_item_indent_X_tl` ..... 72  
`\__enumext_fake_item_vii:` .... [658](#), [686](#), [2542](#)  
`\__enumext_fake_item_viii:` .... [658](#), [698](#), [2547](#)  
`\__enumext_filter_series:n` 53, [1424](#), [1424](#), [1465](#), [1483](#), [1494](#)  
`\__enumext_filter_series_key:n` 53, [1424](#), [1429](#), [1433](#)  
`\__enumext_filter_series_pair:nn` .. 53, [1424](#), [1430](#), [1442](#)  
`\g__enumext_footnote_arg_seq` . [143](#), [2292](#), [2305](#), [2315](#)  
`\g__enumext_footnote_int` . [143](#), [2299](#), [2302](#), [2304](#), [2306](#)  
`\g__enumext_footnote_int_seq` . [143](#), [2293](#), [2306](#), [2311](#), [2314](#)  
`\__enumext_footnotes_key_bool` ..... 29  
`\l__enumext_footnotes_key_bool` 25, 29, 93, [133](#), [264](#), [269](#), [278](#), [3361](#), [3409](#), [3724](#), [3770](#)  
`\__enumext_footnotetext:nn` ... [2286](#), [2286](#), [2316](#)  
`\__enumext_getkeyans:nn` .. 101, [3809](#), [3813](#), [3813](#)  
`\__enumext_getkeyans_aux:n` 101, [3797](#), [3800](#), [3800](#)  
`\l__enumext_hyperref_bool` 25, 29, [133](#), [260](#), [281](#), [298](#), [1955](#), [2174](#), [3348](#), [3715](#)  
`\__enumext_hypertarget:nn` 29, [255](#), [283](#), [287](#), [303](#)  
`\__enumext_if_is_int:n` ..... 194  
`\__enumext_if_is_int:nTF` ..... 194, [542](#), [556](#)  
`\l__enumext_item_column_pos_vii_int` 91, [3053](#), [3059](#), [3065](#), [3069](#), [3076](#), [3276](#), [3413](#), [3416](#)  
`\l__enumext_item_column_pos_viii_int` ... 97, [3476](#), [3482](#), [3488](#), [3492](#), [3499](#), [3620](#), [3774](#), [3777](#)  
`\l__enumext_item_column_pos_X_int` ..... 157  
`\g__enumext_item_count_all_vii_int` 91, [3077](#), [3277](#), [3424](#), [3431](#)  
`\g__enumext_item_count_all_viii_int` 97, [3500](#), [3621](#), [3785](#), [3792](#)  
`\g__enumext_item_count_all_X_int` ..... 157  
`\__enumext_item_peek_args_vii:` 91, [3278](#), [3280](#), [3280](#)  
`\__enumext_item_peek_args_viii:` . 97, 98, [3622](#), [3624](#), [3624](#)  
`\__enumext_item_starred:` .. 72, [2410](#), [2410](#), [2428](#)  
`\l__enumext_item_starred_vii_bool` [3295](#), [3311](#), [3365](#)  
`\l__enumext_item_starred_viii_bool` [3639](#), [3655](#),



3728, 3748	2501
\l__enumext_item_starred_X_bool . . . . . 157	\__enumext_keyans_mini_addvspace: 46, 82, 1096, 1096, 2848
\__enumext_item_std:w 28, 70–72, 85, 242, 246, 2328, 2334, 2358, 2376, 2380, 2388, 3001	\__enumext_keyans_mini_right_cmd:n 49, 1194, 1217, 1217
\g__enumext_item_symbol_aux_vii_tl 3319, 3367, 3370, 3374, 3376	\__enumext_keyans_mini_set_vskip: . 45, 1034, 1034, 1098
\g__enumext_item_symbol_aux_X_tl . . . . . 157	\__enumext_keyans_multi_addvspace: . 82, 883, 894, 2873
\l__enumext_item_symbol_sep_vii_dim . . 3328, 3336, 3373, 3375	\__enumext_keyans_multi_set_vskip: . 42, 883, 883, 896
\g__enumext_item_symbol_tl 23, 70, 37, 2343, 2416, 2433	\__enumext_keyans_multicols_start: 82, 2852, 2854, 2854
\l__enumext_item_symbol_vii_tl . . . . . 3370	\__enumext_keyans_multicols_stop: . 82, 1221, 2879, 2879, 2903
\l__enumext_item_text_vii_box 3360, 3401, 3408	\__enumext_keyans_parse_keys:n 2798, 2832, 2832
\l__enumext_item_text_viii_box 3723, 3762, 3769	\l__enumext_keyans_pic_above_int . 116, 3011, 3012, 3014
\l__enumext_item_text_X_box . . . . . 157	\l__enumext_keyans_pic_above_skip . . 85, 116, 2956, 2995
\l__enumext_item_width_vii_dim . . 3035, 3080, 3088, 3089	\__enumext_keyans_pic_arg_two: 85, 2954, 2983, 2983
\l__enumext_item_width_viii_dim . . 3458, 3503, 3511, 3512	\l__enumext_keyans_pic_below_int . 116, 3011, 3012, 3015
\l__enumext_item_width_X_dim . . . . . 157	\l__enumext_keyans_pic_body_seq . . 84, 86, 116, 2922, 2961, 3019
\l__enumext_itemindent_X_dim . . . . . 46	\__enumext_keyans_pic_do:n 86, 2961, 2963, 3003, 3003, 3007
\l__enumext_itemsep_vii_skip . . . . . 3430	\l__enumext_keyans_pic_level_int . . 20, 1184, 1882, 2079, 2118, 2153, 2236, 2972, 2973
\l__enumext_itemsep_viii_skip . . . . . 3791	\__enumext_keyans_pic_row:n 86, 3005, 3008, 3008
\l__enumext_joined_item_aux_vii_int . . 3074, 3075, 3076, 3077, 3083	\__enumext_keyans_pic_safe_exec: . . 85, 2950, 2970, 2970
\l__enumext_joined_item_aux_viii_int . 3497, 3498, 3499, 3500, 3506	\__enumext_keyans_pic_skip_abs:N . . 85, 2978, 2978, 2994
\l__enumext_joined_item_aux_X_int . . . 157	\l__enumext_keyans_pic_width_dim . 116, 3010, 3017
\__enumext_joined_item_vii:w . . 91, 3283, 3284, 3286, 3286	\__enumext_keyans_redefine_item: . . 72, 2394, 2394, 2500
\l__enumext_joined_item_vii_int . . 3045, 3046, 3049, 3051, 3057, 3062, 3067, 3072, 3074, 3080	\__enumext_keyans_safe_exec: . 2797, 2813, 2813
\__enumext_joined_item_viii:w . 98, 3627, 3628, 3630, 3630	\__enumext_keyans_show_ans: . . 2200, 2208, 2227
\l__enumext_joined_item_viii_int . 3468, 3469, 3472, 3474, 3480, 3485, 3490, 3495, 3497, 3503	\__enumext_keyans_show_item_opt: . 2200, 2215, 2388, 2946, 3751
\l__enumext_joined_item_X_int . . . . . 157	\__enumext_keyans_show_left:n . 72, 2200, 2200, 2386, 2941
\l__enumext_joined_width_vii_dim . 3078, 3085, 3088, 3391, 3403	\__enumext_keyans_show_pos: . . 2200, 2212, 2234
\l__enumext_joined_width_viii_dim 3501, 3508, 3511, 3745, 3764	\__enumext_keyans_starred_item:n . . 72, 2383, 2383, 2402
\l__enumext_joined_width_X_dim . . . . . 157	\__enumext_keyans_store_ref: . . 66, 2097, 2097, 2391, 2934, 3684
\__enumext_keyans_addto_prop:n 65, 2076, 2076, 2390, 2933	\__enumext_keyans_store_ref_aux_i: 66, 2097, 2109, 2112
\__enumext_keyans_addto_seq:n . 67, 2150, 2150, 2392, 2935	\__enumext_keyans_store_ref_aux_ii: 67, 2097, 2138, 2140
\__enumext_keyans_addto_seq_link: 2150, 2168, 2170, 3686	\l__enumext_keyans_tmpa_dim . . . . . 84
\__enumext_keyans_anspic_code:nnn . 84, 2924, 2927, 2927	\l__enumext_keyans_tmpa_tl 24, 99, 84, 2385, 2389
\__enumext_keyans_check_ans:nn . . 67, 68, 2192, 2192, 2809, 2965, 3580	\l__enumext_keyans_tmpb_tl . . . . . 99, 84
\__enumext_keyans_default_item:n . . 71, 2371, 2371, 2406	\__enumext_keyans_wrapper_opt:n . . 1695, 2223
\l__enumext_keyans_env_bool 20, 2645, 2658, 2820, 2904	\l__enumext_label_copy_i_tl . . 2009, 2116, 2121, 2126, 2131
\__enumext_keyans_fake_item: . . 658, 674, 2502	\l__enumext_label_copy_v_tl . . . . . 2126
\l__enumext_keyans_item_opt_tl 84, 2204, 2217, 2223, 3671	\l__enumext_label_copy_vi_tl . . . . . 2121
\l__enumext_keyans_level_h_int 20, 2128, 3586, 3587	\l__enumext_label_copy_vii_tl 1984, 1995, 2026, 2116
\l__enumext_keyans_level_int . . 20, 1192, 1878, 2123, 2819, 2823, 2918	\l__enumext_label_copy_viii_tl . . . . . 2131
\__enumext_keyans_make_label: 31, 73, 2436, 2436,	

<code>\l__enumext_label_copy_X_tl</code> . . . . .	<a href="#">135</a>	<code>\__enumext_list_arg_two_vii:</code> . . . . .	<a href="#">2521</a> , <a href="#">3148</a>
<code>\l__enumext_label_fill_left_v_tl</code> . . . . .	<a href="#">2440</a>	<code>\__enumext_list_arg_two_viii:</code> . . . . .	<a href="#">2521</a> , <a href="#">3569</a>
<code>\l__enumext_label_fill_left_X_tl</code> . . . . .	<a href="#">72</a>	<code>\l__enumext_listoffset_v_dim</code> . . . . .	<a href="#">2865</a>
<code>\l__enumext_label_fill_right_v_tl</code> . . . . .	<a href="#">2447</a>	<code>\l__enumext_listparindent_vii_dim</code> . . . . .	<a href="#">3392</a>
<code>\l__enumext_label_fill_right_X_tl</code> . . . . .	<a href="#">72</a>	<code>\l__enumext_listparindent_viii_dim</code> . . . . .	<a href="#">3746</a>
<code>\l__enumext_label_font_style_v_tl</code> . . . . .	<a href="#">2441</a> , <a href="#">2945</a>	<code>\__enumext_make_label:</code> . . . . .	<a href="#">31</a> , <a href="#">70</a> , <a href="#">71</a> , <a href="#">73</a> , <a href="#">2420</a> , <a href="#">2420</a> , <a href="#">2510</a>
<code>\l__enumext_label_font_style_vii_tl</code> . . . . .	<a href="#">3379</a>	<code>\l__enumext_mark_answer_sym_tl</code> . . . . .	<a href="#">61</a> , <a href="#">111</a> , <a href="#">1701</a> , <a href="#">1856</a> , <a href="#">2062</a> , <a href="#">2238</a> , <a href="#">2251</a> , <a href="#">3693</a>
<code>\l__enumext_label_font_style_viii_tl</code> . . . . .	<a href="#">3733</a>	<code>\l__enumext_mark_position_str</code> . . . . .	<a href="#">111</a> , <a href="#">1705</a> , <a href="#">1706</a> , <a href="#">1729</a> , <a href="#">1730</a> , <a href="#">1854</a>
<code>\l__enumext_label_i_tl</code> . . . . .	<a href="#">487</a>	<code>\l__enumext_mark_ref_sym_tl</code> . . . . .	<a href="#">111</a> , <a href="#">1715</a> , <a href="#">1960</a> , <a href="#">2182</a>
<code>\l__enumext_label_ii_tl</code> . . . . .	<a href="#">487</a>	<code>\__enumext_mini_addvspace:</code> . . . . .	<a href="#">45</a> , <a href="#">78</a> , <a href="#">1017</a> , <a href="#">1017</a> , <a href="#">2692</a>
<code>\l__enumext_label_iii_tl</code> . . . . .	<a href="#">487</a>	<code>\__enumext_mini_addvspace_vii:</code> . . . . .	<a href="#">47</a> , <a href="#">1170</a> , <a href="#">1170</a> , <a href="#">3106</a>
<code>\l__enumext_label_iv_tl</code> . . . . .	<a href="#">487</a>	<code>\__enumext_mini_addvspace_viii:</code> . . . . .	<a href="#">47</a> , <a href="#">1170</a> , <a href="#">1176</a> , <a href="#">3529</a>
<code>\__enumext_label_style:Nnn</code> . . . . .	<a href="#">23</a> , <a href="#">31</a> , <a href="#">339</a> , <a href="#">339</a> , <a href="#">354</a> , <a href="#">492</a> , <a href="#">512</a> , <a href="#">528</a> , <a href="#">532</a>	<code>\__enumext_mini_env*</code> . . . . .	<a href="#">911</a>
<code>\l__enumext_label_v_tl</code> . . . . .	<a href="#">65</a> , <a href="#">67</a> , <a href="#">525</a> , <a href="#">2084</a> , <a href="#">2158</a> , <a href="#">2229</a> , <a href="#">2263</a> , <a href="#">2385</a> , <a href="#">2389</a> , <a href="#">2801</a> , <a href="#">2940</a> , <a href="#">2942</a>	<code>\__enumext_mini_right_cmd:n</code> . . . . .	<a href="#">48</a> , <a href="#">49</a> , <a href="#">1196</a> , <a href="#">1198</a> , <a href="#">1198</a>
<code>\l__enumext_label_vi_tl</code> . . . . .	<a href="#">65</a> , <a href="#">67</a> , <a href="#">525</a> , <a href="#">2081</a> , <a href="#">2155</a> , <a href="#">2940</a> , <a href="#">2942</a> , <a href="#">2946</a>	<code>\__enumext_mini_set_vskip:</code> . . . . .	<a href="#">43</a> , <a href="#">918</a> , <a href="#">918</a> , <a href="#">1019</a>
<code>\l__enumext_label_vii_tl</code> . . . . .	<a href="#">507</a> , <a href="#">3306</a> , <a href="#">3331</a> , <a href="#">3338</a>	<code>\__enumext_mini_set_vskip_vii:</code> . . . . .	<a href="#">46</a> , <a href="#">1113</a> , <a href="#">1113</a> , <a href="#">1172</a>
<code>\l__enumext_label_viii_tl</code> . . . . .	<a href="#">507</a> , <a href="#">3650</a> , <a href="#">3678</a> , <a href="#">3682</a>	<code>\__enumext_mini_set_vskip_viii:</code> . . . . .	<a href="#">46</a> , <a href="#">1113</a> , <a href="#">1135</a> , <a href="#">1178</a>
<code>\l__enumext_label_width_by_box</code> . . . . .	<a href="#">42</a> , <a href="#">335</a> , <a href="#">336</a>	<code>\__enumext_minipage:w</code> . . . . .	<a href="#">28</a> , <a href="#">248</a> , <a href="#">250</a> , <a href="#">913</a> , <a href="#">3017</a> , <a href="#">3391</a> , <a href="#">3745</a>
<code>\__enumext_label_width_by_box:Nn</code> . . . . .	<a href="#">30</a> , <a href="#">333</a> , <a href="#">333</a> , <a href="#">338</a> , <a href="#">350</a> , <a href="#">566</a>	<code>\l__enumext_minipage_active_v_bool</code> . . . . .	<a href="#">82</a> , <a href="#">83</a> , <a href="#">2846</a> , <a href="#">2871</a> , <a href="#">2884</a> , <a href="#">2892</a>
<code>\l__enumext_labelsep_i_dim</code> . . . . .	<a href="#">2231</a> , <a href="#">2267</a> , <a href="#">3689</a> , <a href="#">3704</a>	<code>\g__enumext_minipage_active_vii_bool</code> . . . . .	<a href="#">88</a> , <a href="#">3117</a> , <a href="#">3122</a> , <a href="#">3134</a>
<code>\l__enumext_labelsep_v_dim</code> . . . . .	<a href="#">2863</a>	<code>\l__enumext_minipage_active_vii_bool</code> . . . . .	<a href="#">3102</a> , <a href="#">3113</a>
<code>\l__enumext_labelsep_vii_dim</code> . . . . .	<a href="#">3030</a> , <a href="#">3039</a> , <a href="#">3081</a> , <a href="#">3329</a> , <a href="#">3389</a> , <a href="#">3405</a>	<code>\g__enumext_minipage_active_viii_bool</code> . . . . .	<a href="#">3540</a> , <a href="#">3545</a> , <a href="#">3557</a>
<code>\l__enumext_labelsep_viii_dim</code> . . . . .	<a href="#">3453</a> , <a href="#">3462</a> , <a href="#">3504</a> , <a href="#">3743</a> , <a href="#">3766</a>	<code>\l__enumext_minipage_active_viii_bool</code> . . . . .	<a href="#">3525</a> , <a href="#">3536</a>
<code>\l__enumext_labelwidth_i_dim</code> . . . . .	<a href="#">2231</a> , <a href="#">2266</a> , <a href="#">3689</a> , <a href="#">3704</a>	<code>\g__enumext_minipage_active_X_bool</code> . . . . .	<a href="#">157</a>
<code>\l__enumext_labelwidth_v_dim</code> . . . . .	<a href="#">2863</a>	<code>\l__enumext_minipage_active_X_bool</code> . . . . .	<a href="#">60</a>
<code>\l__enumext_labelwidth_vii_dim</code> . . . . .	<a href="#">3030</a> , <a href="#">3038</a> , <a href="#">3081</a> , <a href="#">3382</a> , <a href="#">3386</a> , <a href="#">3404</a>	<code>\g__enumext_minipage_after_skip</code> . . . . .	<a href="#">60</a> , <a href="#">1117</a> , <a href="#">1129</a> , <a href="#">3132</a> , <a href="#">3555</a>
<code>\l__enumext_labelwidth_viii_dim</code> . . . . .	<a href="#">3453</a> , <a href="#">3461</a> , <a href="#">3504</a> , <a href="#">3736</a> , <a href="#">3740</a> , <a href="#">3765</a>	<code>\l__enumext_minipage_after_skip</code> . . . . .	<a href="#">43</a> , <a href="#">44</a> , <a href="#">79</a> , <a href="#">83</a> , <a href="#">60</a> , <a href="#">934</a> , <a href="#">949</a> , <a href="#">969</a> , <a href="#">985</a> , <a href="#">1000</a> , <a href="#">1006</a> , <a href="#">1012</a> , <a href="#">1026</a> , <a href="#">1036</a> , <a href="#">1045</a> , <a href="#">1048</a> , <a href="#">1060</a> , <a href="#">1078</a> , <a href="#">1089</a> , <a href="#">1105</a> , <a href="#">1137</a> , <a href="#">1150</a> , <a href="#">1164</a> , <a href="#">2759</a> , <a href="#">2901</a>
<code>\l__enumext_leftmargin_tmp_v_bool</code> . . . . .	<a href="#">85</a> , <a href="#">2985</a>	<code>\g__enumext_minipage_center_vii_bool</code> . . . . .	<a href="#">3126</a> , <a href="#">3135</a>
<code>\l__enumext_leftmargin_tmp_X_bool</code> . . . . .	<a href="#">46</a>	<code>\g__enumext_minipage_center_viii_bool</code> . . . . .	<a href="#">3549</a> , <a href="#">3558</a>
<code>\l__enumext_leftmargin_tmp_X_dim</code> . . . . .	<a href="#">46</a>	<code>\g__enumext_minipage_center_X_bool</code> . . . . .	<a href="#">157</a>
<code>\l__enumext_leftmargin_X_dim</code> . . . . .	<a href="#">46</a>	<code>\l__enumext_minipage_hsep_v_dim</code> . . . . .	<a href="#">81</a> , <a href="#">2844</a>
<code>\__enumext_level:</code> . . . . .	<a href="#">190</a> , <a href="#">190</a> , <a href="#">430</a> , <a href="#">432</a> , <a href="#">433</a> , <a href="#">441</a> , <a href="#">443</a> , <a href="#">661</a> , <a href="#">665</a> , <a href="#">669</a> , <a href="#">736</a> , <a href="#">740</a> , <a href="#">744</a> , <a href="#">748</a> , <a href="#">832</a> , <a href="#">834</a> , <a href="#">836</a> , <a href="#">838</a> , <a href="#">871</a> , <a href="#">873</a> , <a href="#">875</a> , <a href="#">877</a> , <a href="#">881</a> , <a href="#">921</a> , <a href="#">924</a> , <a href="#">943</a> , <a href="#">952</a> , <a href="#">958</a> , <a href="#">963</a> , <a href="#">967</a> , <a href="#">978</a> , <a href="#">982</a> , <a href="#">983</a> , <a href="#">988</a> , <a href="#">1024</a> , <a href="#">1028</a> , <a href="#">1201</a> , <a href="#">1207</a> , <a href="#">1254</a> , <a href="#">1256</a> , <a href="#">1258</a> , <a href="#">1261</a> , <a href="#">1268</a> , <a href="#">1270</a> , <a href="#">1272</a> , <a href="#">1275</a> , <a href="#">1786</a> , <a href="#">1794</a> , <a href="#">1798</a> , <a href="#">1802</a> , <a href="#">2047</a> , <a href="#">2050</a> , <a href="#">2051</a> , <a href="#">2327</a> , <a href="#">2328</a> , <a href="#">2332</a> , <a href="#">2333</a> , <a href="#">2334</a> , <a href="#">2341</a> , <a href="#">2343</a> , <a href="#">2347</a> , <a href="#">2348</a> , <a href="#">2351</a> , <a href="#">2357</a> , <a href="#">2358</a> , <a href="#">2412</a> , <a href="#">2415</a> , <a href="#">2417</a> , <a href="#">2424</a> , <a href="#">2425</a> , <a href="#">2426</a> , <a href="#">2429</a> , <a href="#">2432</a> , <a href="#">2562</a> , <a href="#">2564</a> , <a href="#">2613</a> , <a href="#">2618</a> , <a href="#">2619</a> , <a href="#">2620</a> , <a href="#">2622</a> , <a href="#">2626</a> , <a href="#">2631</a> , <a href="#">2632</a> , <a href="#">2633</a> , <a href="#">2635</a> , <a href="#">2651</a> , <a href="#">2664</a> , <a href="#">2671</a> , <a href="#">2682</a> , <a href="#">2684</a> , <a href="#">2687</a> , <a href="#">2688</a> , <a href="#">2690</a> , <a href="#">2695</a> , <a href="#">2702</a> , <a href="#">2705</a> , <a href="#">2707</a> , <a href="#">2709</a> , <a href="#">2710</a> , <a href="#">2711</a> , <a href="#">2712</a> , <a href="#">2715</a> , <a href="#">2721</a> , <a href="#">2726</a> , <a href="#">2732</a> , <a href="#">2735</a> , <a href="#">2737</a> , <a href="#">2750</a>	<code>\l__enumext_minipage_hsep_vii_dim</code> . . . . .	<a href="#">3100</a>
<code>\l__enumext_level_h_int</code> . . . . .	<a href="#">20</a> , <a href="#">222</a> , <a href="#">449</a> , <a href="#">473</a> , <a href="#">1478</a> , <a href="#">1488</a> , <a href="#">1640</a> , <a href="#">1657</a> , <a href="#">2003</a> , <a href="#">2020</a> , <a href="#">2584</a> , <a href="#">3165</a> , <a href="#">3166</a> , <a href="#">3174</a>	<code>\l__enumext_minipage_hsep_viii_dim</code> . . . . .	<a href="#">3523</a>
<code>\l__enumext_level_int</code> . . . . .	<a href="#">20</a> , <a href="#">192</a> , <a href="#">233</a> , <a href="#">844</a> , <a href="#">995</a> , <a href="#">1188</a> , <a href="#">1477</a> , <a href="#">1489</a> , <a href="#">1634</a> , <a href="#">1980</a> , <a href="#">1990</a> , <a href="#">1996</a> , <a href="#">2002</a> , <a href="#">2010</a> , <a href="#">2018</a> , <a href="#">2025</a> , <a href="#">2515</a> , <a href="#">2576</a> , <a href="#">2577</a> , <a href="#">2583</a> , <a href="#">2596</a> , <a href="#">2603</a> , <a href="#">2649</a> , <a href="#">2662</a> , <a href="#">2717</a> , <a href="#">2784</a> , <a href="#">2827</a> , <a href="#">2914</a> , <a href="#">3175</a> , <a href="#">3257</a> , <a href="#">3267</a> , <a href="#">3436</a> , <a href="#">3593</a>	<code>\l__enumext_minipage_left_skip</code> . . . . .	<a href="#">43</a> , <a href="#">82</a> , <a href="#">60</a> , <a href="#">926</a> , <a href="#">941</a> , <a href="#">960</a> , <a href="#">975</a> , <a href="#">1022</a> , <a href="#">1032</a> , <a href="#">1037</a> , <a href="#">1043</a> , <a href="#">1052</a> , <a href="#">1069</a> , <a href="#">1081</a> , <a href="#">1101</a> , <a href="#">1111</a> , <a href="#">1115</a> , <a href="#">1120</a> , <a href="#">1124</a> , <a href="#">1138</a> , <a href="#">1142</a> , <a href="#">1156</a> , <a href="#">1174</a> , <a href="#">1180</a>
<code>\__enumext_list_arg_two_i:</code> . . . . .	<a href="#">2482</a>	<code>\l__enumext_minipage_left_v_dim</code> . . . . .	<a href="#">81</a> , <a href="#">2842</a> , <a href="#">2850</a>
<code>\__enumext_list_arg_two_ii:</code> . . . . .	<a href="#">2482</a>	<code>\l__enumext_minipage_left_vii_dim</code> . . . . .	<a href="#">3096</a> , <a href="#">3108</a>
<code>\__enumext_list_arg_two_iii:</code> . . . . .	<a href="#">2482</a>	<code>\l__enumext_minipage_left_viii_dim</code> . . . . .	<a href="#">3519</a> , <a href="#">3531</a>
<code>\__enumext_list_arg_two_iv:</code> . . . . .	<a href="#">2482</a>	<code>\l__enumext_minipage_left_X_dim</code> . . . . .	<a href="#">60</a>
<code>\__enumext_list_arg_two_v:</code> . . . . .	<a href="#">72</a> , <a href="#">2482</a> , <a href="#">2803</a> , <a href="#">2986</a>	<code>\g__enumext_minipage_right_skip</code> . . . . .	<a href="#">60</a> , <a href="#">1116</a> , <a href="#">1121</a> , <a href="#">1125</a> , <a href="#">3125</a> , <a href="#">3548</a>



`\l__enumext_minipage_right_skip` .. 43, 60, 930, 945, 965, 980, 1038, 1044, 1056, 1074, 1085, 1139, 1146, 1160, 1208, 1225  
`\l__enumext_minipage_right_v_dim` .. 81, 1219, 1224, 2840, 2844  
`\g__enumext_minipage_right_vii_dim` 87, 3104, 3124, 3137  
`\l__enumext_minipage_right_vii_dim` 87, 3094, 3099, 3105  
`\g__enumext_minipage_right_viii_dim` .. 3527, 3547, 3560  
`\l__enumext_minipage_right_viii_dim` .. 3517, 3522, 3528  
`\g__enumext_minipage_right_X_dim` ..... 157  
`\g__enumext_minipage_right_X_skip` .... 157  
`\g__enumext_minipage_stat_int` . 78, 82, 60, 1213, 1230, 2691, 2752, 2757, 2847, 2894, 2899  
`\g__enumext_miniright_code_vii_tl` . 88, 3130, 3136  
`\g__enumext_miniright_code_viii_tl` 3553, 3559  
`\g__enumext_miniright_code_X_tl` ..... 157  
`\__enumext_multi_addvspace:` ... 42, 79, 866, 866, 2723  
`\__enumext_multi_set_vskip:` .. 41, 830, 830, 868  
`\l__enumext_multicols_above_ii_skip` ... 849  
`\l__enumext_multicols_above_iii_skip` .. 855  
`\l__enumext_multicols_above_iv_skip` ... 861  
`\l__enumext_multicols_above_v_skip` 885, 899, 909  
`\l__enumext_multicols_above_X_skip` .... 54  
`\l__enumext_multicols_below_v_skip` 889, 903, 2886  
`\l__enumext_multicols_below_X_skip` .... 54  
`\__enumext_multicols_start:` 78, 2697, 2699, 2699  
`\__enumext_multicols_stop:` 79, 1203, 2729, 2729, 2761  
`\__enumext_newlabel:nn` 25, 29, 64, 291, 291, 2036, 2144  
`\l__enumext_newlabel_arg_one_tl` 25, 29, 64, 66, 135, 1959, 2029, 2037, 2133, 2145, 2180  
`\l__enumext_newlabel_arg_two_tl` 25, 29, 63, 135, 1983, 1993, 2007, 2023, 2038, 2120, 2125, 2130, 2146  
`\__enumext_parse_keys:n` ..... 2558, 2591, 2591  
`\__enumext_parse_keys_parse_keys:n` .... 53  
`\__enumext_parse_keys_vii:n` 53, 3143, 3182, 3182  
`\__enumext_parse_keys_viii:n` . 3565, 3598, 3598  
`\__enumext_parse_series_resume:n` .. 53, 1455, 1455, 2599, 3188  
`\__enumext_parse_store_keys:n` 76, 77, 2607, 2611, 2611  
`\__enumext_parse_store_keys_vii:n` . 89, 3191, 3195, 3195  
`\l__enumext_parsep_i_skip` . 847, 849, 998, 1046  
`\l__enumext_parsep_ii_skip` .... 853, 855, 1004  
`\l__enumext_parsep_iii_skip` ... 859, 861, 1010  
`\l__enumext_parsep_vii_skip` ..... 3393  
`\l__enumext_parsep_viii_skip` ..... 3747  
`\l__enumext_partopsep_v_skip` .. 901, 905, 1072, 1076, 1083, 1087, 1103, 1107  
`\l__enumext_partopsep_viii_skip` ..... 1148  
`\__enumext_phantomsection:` 29, 255, 284, 288, 304  
`\__enumext_print_footnote:` ... 2286, 2309, 3411, 3772  
`\__enumext_print_keyans_box:NN` 61, 1848, 1848, 1861, 2049, 2231, 2265, 3689, 3704  
`\l__enumext_print_keyans_i_tl` .... 3823, 3852  
`\l__enumext_print_keyans_ii_tl` ... 3828, 3853  
`\l__enumext_print_keyans_iii_tl` .. 3833, 3854  
`\l__enumext_print_keyans_iv_tl` ... 3838, 3855  
`\l__enumext_print_keyans_vii_tl` .. 3843, 3856  
`\l__enumext_print_keyans_X_tl` ..... 100  
`\__enumext_printkeyans:nnn` 102, 3857, 3860, 3860  
`\__enumext_redefine_item:` . 71, 2360, 2360, 2509  
`\l__enumext_ref_aux_tl` 146, 430, 432, 435, 451, 453, 456, 461, 463, 466  
`\l__enumext_ref_key_arg_tl` .. 146, 424, 429, 436, 448, 457, 467  
`\__enumext_regex_label_ref_key:` .. 32, 33, 419, 419, 431, 452, 462  
`\__enumext_register_counter_style:Nn` .. 323, 323, 328, 329, 330, 331, 332  
`\__enumext_remove_extra_parsep_vii:` .. 3158, 3420, 3420  
`\__enumext_remove_extra_parsep_viii:` . 3579, 3781, 3781  
`\__enumext_renew_footnote:` ... 2286, 2290, 3363, 3726  
`\l__enumext_resume_bool` ..... 23  
`\__enumext_resume_counter:` 54, 1497, 1501, 1507, 1516, 1572  
`\__enumext_resume_counter_new:n` .. 1502, 1508, 1511, 1532  
`\__enumext_resume_counter_series:n` 54, 1351, 1497, 1497  
`\__enumext_resume_counter_series_vii:n` 1361, 1576, 1576  
`\__enumext_resume_counter_vii:` ... 1576, 1580, 1585, 1592, 1610  
`\g__enumext_resume_int` 23, 80, 37, 1522, 1527, 1528, 1535, 1539, 1540, 1559, 1565, 1566, 2765, 2766, 2769  
`\l__enumext_resume_name_bool` . 1457, 1530, 1544  
`\l__enumext_resume_name_tl` ... 1531, 1543, 1550, 2776, 2777, 2779  
`\__enumext_resume_series_default:n` 53, 1455, 1460, 1472  
`\__enumext_resume_starred:` ... 1352, 1497, 1568  
`\__enumext_resume_starred_vii:` 1362, 1576, 1606  
`\g__enumext_resume_vii_int` .. 90, 37, 1598, 1603, 1604, 3236  
`\__enumext_resumext_series_default:n` ... 54  
`\__enumext_safe_exec:` ..... 2557, 2574, 2574  
`\__enumext_safe_exec_vii:` ... 3142, 3163, 3163  
`\__enumext_safe_exec_viii:` ... 3564, 3584, 3584  
`\g__enumext_series_standar_default_tl` .. 37, 1482, 1483, 1570, 1573  
`\g__enumext_series_starred_default_tl` .. 37, 1493, 1494, 1608, 1611  
`\l__enumext_series_str` .. 1349, 1359, 1458, 1463, 1464, 1466, 1468, 2595, 2772, 2774, 3186  
`\__enumext_set_error:nn` ..... 3938, 3948, 3950  
`\__enumext_set_label_ref:n` ... 32, 427, 427, 499  
`\__enumext_set_label_ref_h:n` . 33, 446, 446, 519  
`\__enumext_set_parse:n` ..... 3921, 3938, 3938  
`\l__enumext_setkey_tmpa_int` ... 95, 3914, 3918  
`\l__enumext_setkey_tmpa_seq` 95, 3912, 3922, 3928, 3930, 3932, 3945  
`\l__enumext_setkey_tmpa_tl` .... 95, 3920, 3924

```

\l__enumext_setkey_tmpb_seq 95, 3913, 3916, 3920,
    3921
\l__enumext_setkey_tmpb_tl 95, 3940, 3942, 3943
\l__enumext_show_answer_bool . 111, 1709, 1733,
    2056, 2206, 2220, 2937, 3687
\__enumext_show_length:nnn . . 38, 200, 200, 3990,
    3991, 3992, 3993, 3994, 3995, 3996, 3997, 3998, 3999,
    4005, 4006, 4007, 4008, 4009, 4010, 4011, 4012, 4013,
    4014
\l__enumext_show_position_bool 111, 1712, 1736,
    2060, 2210, 2221, 2938, 3691
\g__enumext_standar_bool . 27, 20, 221, 224, 1476,
    1660, 2582, 2790
\l__enumext_standar_bool . 20, 1988, 2001, 2017,
    2579, 2764
\g__enumext_standar_keyans_pic_star_env-
    int . . . . . 132
\g__enumext_standar_keyans_star_env_int 131
\l__enumext_standar_level_one_bool 20, 1389,
    1519, 1556, 2588
\g__enumext_standar_star_env_int . . 128, 225
\__enumext_standard_item_vii:w 91, 3291, 3293,
    3293
\__enumext_standard_item_viii:w 98, 3635, 3637,
    3637
\g__enumext_starred_bool 27, 89, 90, 20, 232, 235,
    1487, 1639, 1979, 1989, 2019, 2114, 2646, 2659, 2742,
    3173, 3244, 3250, 3442
\l__enumext_starred_bool . 89, 90, 20, 1912, 1920,
    2004, 2045, 3170, 3251
\__enumext_starred_columns_set_vii: . . 3024,
    3024, 3151
\__enumext_starred_columns_set_viii: . 3447,
    3447, 3572
\__enumext_starred_item:nn . . 2337, 2337, 2366
\__enumext_starred_item_exec: . 99, 3680, 3680,
    3730
\__enumext_starred_item_vii:w 91, 92, 3290, 3309,
    3309
\__enumext_starred_item_vii_aux_i:w . . 3309,
    3314, 3317
\__enumext_starred_item_vii_aux_ii:w . 3309,
    3315, 3320, 3322
\__enumext_starred_item_vii_aux_iii:w 3309,
    3325, 3334
\__enumext_starred_item_viii:w 98, 3634, 3653,
    3653
\__enumext_starred_item_viii_aux_i:w . 3653,
    3658, 3661
\__enumext_starred_item_viii_aux_ii:w 3653,
    3659, 3673, 3675
\__enumext_starred_joined_item_vii:n . 87, 91,
    3043, 3043, 3288
\__enumext_starred_joined_item_viii:n 95, 98,
    3466, 3466, 3632
\g__enumext_starred_keyans_star_env_int 130
\l__enumext_starred_level_one_bool 20, 1399,
    1595, 3179
\g__enumext_starred_star_env_int . . 129, 236
\__enumext_start_from:NNn 35, 540, 540, 553, 575
\l__enumext_start_i_int . 1528, 1540, 1552, 1560,
    1566
\__enumext_start_item_tmp_vii: 88, 3154, 3273,
    3273
\__enumext_start_item_tmp_viii: 96, 3575, 3617,
    3617
\__enumext_start_item_vii:w . 91, 92, 3301, 3306,
    3331, 3338, 3340, 3340
\__enumext_start_item_viii:w . . 98, 3645, 3650,
    3678, 3707, 3707
\__enumext_start_list:nn 28, 74, 85, 242, 244, 2561,
    2800, 2951, 3146, 3567
\__enumext_start_mini_vii: . 90, 3092, 3092, 3229
\__enumext_start_mini_viii: 97, 3515, 3515, 3609
\__enumext_start_store_level: . 77, 2560, 2640,
    2640
\__enumext_start_store_level_vii: . 90, 3145,
    3253, 3253
\l__enumext_start_vii_int . . . . . 1604
\l__enumext_start_X_int . . . . . 72, 570
\__enumext_stop_item_tmp_vii: . 88, 91, 92, 3153,
    3157, 3275, 3342
\__enumext_stop_item_tmp_viii: . . 96, 97, 3574,
    3578, 3619, 3709
\__enumext_stop_item_vii: 92, 93, 3342, 3396, 3396
\__enumext_stop_item_viii: 100, 3709, 3757, 3757
\__enumext_stop_list: . . 28, 242, 245, 2570, 2810,
    2964, 3159, 3581
\__enumext_stop_mini_vii: 88, 90, 3111, 3111, 3233
\__enumext_stop_mini_viii: . 97, 3515, 3534, 3613
\__enumext_stop_store_level: . . 77, 2571, 2640,
    2669
\__enumext_stop_store_level_vii: . . 90, 3160,
    3253, 3263
\l__enumext_store_active_bool 24, 52, 76, 89, 84,
    1417, 1520, 1557, 1596, 1874, 2605, 2644, 2657, 2815,
    2822, 2910, 2968, 3189, 3255, 3265, 3592
\__enumext_store_addto_prop:n 59, 65, 1767, 1768,
    1776, 1899, 2095, 3683
\__enumext_store_addto_seq:n 60, 67, 1777, 1777,
    1781, 1788, 1802, 1810, 1819, 1837, 1845, 1963, 2185
\l__enumext_store_ans_bool 121, 1418, 1621, 1784,
    1808, 1815, 1843, 1887
\l__enumext_store_anskey_arg_tl 24, 62, 63, 84,
    1905, 1914, 1916, 1922, 1930, 1933, 1943, 1948, 1951,
    1957, 1963
\__enumext_store_anskey_code:nnnn . 62, 1893,
    1897, 1897
\__enumext_store_anskey_show_left:n 65, 1904,
    2054, 2054
\__enumext_store_anskey_show_wrap:n 65, 2042,
    2042, 2058, 2073
\l__enumext_store_columns_break_bool . 1868,
    1911
\l__enumext_store_columns_join_int 84, 1919,
    1924
\l__enumext_store_columns_sep_vii_bool 3210
\l__enumext_store_columns_sep_vii_dim 3215,
    3219
\l__enumext_store_columns_sep_X_bool . . 100
\l__enumext_store_columns_sep_X_dim . . . 100
\l__enumext_store_columns_vii_bool . . . 3197
\l__enumext_store_columns_vii_int 3202, 3206
\l__enumext_store_columns_X_bool . . . . 100
\l__enumext_store_columns_X_int . . . . 100
\__enumext_store_internal_ref: . . 62, 63, 1902,
    1965, 1965
\l__enumext_store_item_symbol_sep_dim 1866,
    1940, 1945

```

`\l__enumext_store_item_symbol_tl` . 1864, 1931, 1935  
`\l__enumext_store_keyans_item_opt_sep_tl` . . . . 1698, 2089, 2091, 2162, 2164, 3666, 3668  
`\l__enumext_store_keyans_item_opt_tl` . . . 84  
`\l__enumext_store_keyans_label_tl` 24, 65, 67, 84, 2078, 2081, 2084, 2091, 2093, 2095, 2152, 2155, 2158, 2164, 2166, 2176, 2185, 2186, 3663, 3668, 3669, 3682, 3683, 3685  
`\__enumext_store_level_close:` . 60, 1782, 1806, 2673  
`\__enumext_store_level_close_vii:` 1813, 1841, 3269  
`\__enumext_store_level_open:` . . 59, 60, 77, 1782, 1782, 2652, 2665  
`\__enumext_store_level_open_vii:` . . 89, 1813, 1813, 3259  
`\g__enumext_store_name_tl` 24, 79, 84, 1680, 1683, 2745, 2792, 3248, 3444  
`\l__enumext_store_name_tl` 24, 52, 84, 1367, 1368, 1378, 1379, 1409, 1411, 1413, 1415, 1419, 1421, 1524, 1562, 1600, 1770, 1772, 1779, 2031, 2032, 2068, 2135, 2136, 2244, 2257, 2745, 2767, 2769, 3237, 3240, 3248, 3699  
`\l__enumext_store_opt_vii_tl` . 1817, 1827, 1833, 1837, 3204, 3217  
`\l__enumext_store_opt_X_tl` . . . . . 100  
`\l__enumext_store_ref_key_bool` 62, 1718, 1900, 1954, 2099, 2173  
`\l__enumext_store_upper_level_X_bool` . . 100  
`\l__enumext_store_write_aux_file_tl` 25, 64, 67, 135, 2034, 2040, 2142, 2148  
`\__enumext_storing_exec:` 1365, 1391, 1401, 1407  
`\__enumext_storing_set:n` . . 52, 1347, 1365, 1365  
`\__enumext_storing_set_vii:n` . . . . . 1357, 1376  
`\__enumext_storing_standar:` . . . . . 1373, 1387  
`\__enumext_storing_starred:` . . . . . 1384, 1397  
`\l__enumext_the_counter_vii_tl` . . . . . 453  
`\l__enumext_the_counter_viii_tl` . . . . . 463  
`\l__enumext_the_counter_X_tl` . . . . . 146  
`\__enumext_tmp:n` 32, 36, 46, 53, 54, 59, 66, 71, 72, 83, 100, 110, 138, 142, 150, 156, 157, 176, 208, 212, 653, 657, 1614, 1631, 1688, 1723, 1724, 1741, 1967, 1974, 1975, 1996, 2010, 2013, 2025, 2101, 2108, 2482, 2520, 2521, 2553  
`\__enumext_tmp:nn` 355, 376, 377, 405, 406, 418, 570, 589, 634, 652, 710, 718, 719, 733, 798, 814, 815, 829, 1234, 1250, 1742, 1766, 2270, 2285  
`\__enumext_tmp:nnn` 487, 503, 504, 505, 506, 507, 523, 524  
`\__enumext_tmp:nnnnn` 590, 615, 618, 621, 623, 625, 628, 631  
`\__enumext_tmp:w` . . . . . 3806, 3809  
`\l__enumext_tmpa_vii_int` . . . . . 3034, 3037  
`\l__enumext_tmpa_viii_int` . . . . . 3457, 3460  
`\l__enumext_tmpa_X_int` . . . . . 157  
`\l__enumext_topsep_v_skip` 887, 891, 1041, 1054, 1062, 1067, 1087, 1091, 2967, 2998  
`\l__enumext_topsep_vii_skip` . . 1118, 1127, 1131  
`\l__enumext_topsep_viii_skip` . 1140, 1162, 1166  
`\__enumext_use_key_ref:` . . . 33, 439, 439, 2511  
`\__enumext_use_key_ref_h:` . . 33, 471, 471, 2539  
`\l__enumext_vspace_a_star_v_bool` . . . . 1283  
`\l__enumext_vspace_a_star_vii_bool` . . . 1305  
`\l__enumext_vspace_a_star_viii_bool` . . . 1316  
`\l__enumext_vspace_a_star_X_bool` . . . . . 72  
`\__enumext_vspace_above:` . . 49, 1251, 1251, 2678  
`\__enumext_vspace_above_v:` . 50, 1279, 1279, 2838  
`\l__enumext_vspace_above_v_skip` . . 1281, 1285, 1287  
`\__enumext_vspace_above_vii:` . . 50, 1301, 1301, 3226  
`\l__enumext_vspace_above_vii_skip` 1303, 1307, 1309  
`\__enumext_vspace_above_viii:` . 50, 1301, 1312, 3607  
`\l__enumext_vspace_above_viii_skip` 1314, 1318, 1320  
`\l__enumext_vspace_b_star_v_bool` . . . . . 1294  
`\l__enumext_vspace_b_star_vii_bool` . . . 1327  
`\l__enumext_vspace_b_star_viii_bool` . . . 1338  
`\l__enumext_vspace_b_star_X_bool` . . . . . 72  
`\__enumext_vspace_below:` . . 50, 1265, 1265, 2763  
`\__enumext_vspace_below_v:` . 50, 1290, 1290, 2906  
`\l__enumext_vspace_below_v_skip` . . 1292, 1296, 1298  
`\__enumext_vspace_below_vii:` . . 51, 1323, 1323, 3235  
`\l__enumext_vspace_below_vii_skip` 1325, 1329, 1331  
`\__enumext_vspace_below_viii:` . 51, 1323, 1334, 3615  
`\l__enumext_vspace_below_viii_skip` 1336, 1340, 1342  
`\__enumext_widest_from:nnnn` . . 35, 554, 554, 569, 581  
`\g__enumext_widest_label_tl` 23, 31, 42, 343, 347, 351  
`\l__enumext_wrap_label_opt_v_bool` . . . . 2379  
`\l__enumext_wrap_label_opt_vii_bool` 91, 3300  
`\l__enumext_wrap_label_opt_viii_bool` 98, 3644  
`\l__enumext_wrap_label_opt_X_bool` . . . . . 72  
`\l__enumext_wrap_label_v_bool` 2375, 2379, 2387, 2442  
`\l__enumext_wrap_label_vii_bool` 91, 3299, 3304, 3312, 3380  
`\l__enumext_wrap_label_viii_bool` . . 98, 3643, 3648, 3656, 3734  
`\l__enumext_wrap_label_X_bool` . . . . . 72  
`\__enumext_wrapper_label_v:n` . . . . . 2444, 2946  
`\__enumext_wrapper_label_vii:n` . . . . . 3383  
`\__enumext_wrapper_label_viii:n` . . . . . 3737  
`\__enumext_zero_count_level:` . . . . . 206, 206  
`\__enumext_zero_parsep:` . . . . . 44, 938, 993, 993  
enumext\* . . . . . 5, 3139  
enumXi . . . . . 315  
enumXii . . . . . 315  
enumXiii . . . . . 315  
enumXiv . . . . . 315  
enumXv . . . . . 315  
enumXvi . . . . . 315  
enumXvii . . . . . 315  
enumXviii . . . . . 315  
Environments provide by enumext:  
enumext\* 22, 23, 25–27, 30, 32–34, 37–40, 46, 47, 50, 51, 53, 54, 56, 58–62, 64, 66, 69, 75–77, 89, 90, 92, 94, 95, 97, 99, 102, 105, 106  
enumext 22, 23, 25, 27, 30, 31, 33–36, 38–46, 48–51, 53, 54, 56–62, 64, 66, 69–74, 76, 77, 80, 81, 85–87, 90, 102, 105

keyans*	22–24, 26, 27, 30, 32–34, 37–40, 46, 47, 50–52, 56, 58, 59, 66, 69, 75, 97, 105, 106
keyanspic . .	22–25, 30, 31, 34, 48, 52, 56, 59, 60, 65–68, 83–85, 106
keyans	22–25, 27, 30, 31, 34–36, 38–40, 42, 45, 46, 48–50, 52, 56, 58–60, 65–68, 72–74, 80, 81, 83–85, 87, 97, 105, 106
Environments:	
list . . . . .	27, 28, 73, 74, 76
lrbox . . . . .	86, 93, 100
minipage . . . . .	27, 28, 40, 42, 43, 83–86, 93, 100
multicols . . . . .	41–43, 48, 78, 79, 82, 83
exp commands:	
\exp_after:wN . . . . .	3809
\exp_args:Ne . . . . .	2602, 3797
\exp_not:N 154, 346, 435, 456, 466, 667, 681, 682, 693, 694, 705, 706, 1959, 2065, 2066, 2178, 2241, 2242, 2254, 2255, 3696, 3697, 3806	
\exp_not:n 435, 436, 456, 457, 466, 467, 668, 1440, 1453, 1750, 1757, 1924, 1935, 1945, 1959, 1960, 2037, 2145, 2180, 2182, 2622, 2635, 3206, 3219	
F	
\fbox . . . . .	1693
file commands:	
\file_input_stop: . . . . .	4115
first . . . . .	719
font . . . . .	355
\footnote . . . . .	69
\footnote . . . . .	69, 2294
\footnotemark . . . . .	2304
\footnotesize . . . . .	2066, 2242, 2255, 3697
\footnotetext . . . . .	2288
G	
\getkeyans . . . . .	13, 101, 3795
group commands:	
\group_begin: . . . . .	1886, 2064, 2240, 2253, 3359, 3378, 3695, 3722, 3732, 3817, 3851
\group_end: 1895, 2071, 2247, 2260, 3388, 3400, 3702, 3742, 3761, 3819, 3858	
H	
\hbadness . . . . .	3407, 3768
hbox commands:	
\hbox_set:Nn . . . . .	335
\hfill 385, 389, 394, 395, 1205, 1223, 1959, 2178, 3116, 3539	
hook commands:	
\hook_gput_code:nnn . . . . .	9, 184, 188, 253
\hook_gset_rule:nnnn . . . . .	254
\hspace . . . . .	3418, 3779
\hyperlink . . . . .	63, 67
\hyperlink . . . . .	1959, 2178
\hypertarget . . . . .	29
\hypertarget . . . . .	283
I	
\IfHyperBoolean . . . . .	261
\IfPackageLoadedTF . . . . .	11, 257, 271
\ignorespaces . . . . .	670
\inputlineno . . . . .	225, 236
int commands:	
\int_add:Nn . . . . .	3076, 3499
\int_case:nn . . . . .	844, 995, 1634, 1657
\int_compare:nNnTF 449, 473, 920, 1039, 1184, 1188, 1192, 1677, 1878, 1882, 2079, 2118, 2123, 2128, 2153, 2236, 2577, 2596, 2649, 2662, 2701, 2717, 2731, 2752, 2784, 2823, 2827, 2856, 2881, 2894, 2914, 2918, 2973, 3046, 3056, 3072, 3166, 3257, 3267, 3413, 3422, 3436, 3469, 3479, 3495, 3587, 3593, 3774, 3783, 3918	
\int_compare_p:nNn 222, 233, 1477, 1478, 1488, 1489, 1640, 1980, 1990, 2002, 2003, 2018, 2020, 2583, 2584, 3174, 3175	
\int_decr:N . . . . .	3075, 3498
\int_eval:n 1772, 2032, 2066, 2136, 2242, 2255, 2497, 2538, 3064, 3487, 3697	
\int_from_alph:n . . . . .	548, 562
\int_from_roman:n . . . . .	550, 564
\int_gadd:Nn . . . . .	3077, 3500
\int_gdecr:N . . . . .	1643, 1648, 1651, 1654, 1662
\int_gincr:N 1527, 1535, 1539, 1565, 1603, 1891, 2189, 2325, 2355, 2691, 2847, 3277, 3355, 3621	
\int_gset:Nn . . . . .	225, 236, 1522, 1559, 1598, 2302
\int_gset_eq:NN 2299, 2765, 2769, 2774, 2779, 3236, 3239	
\int_gzero:N 210, 1213, 1230, 1685, 1686, 2757, 2899, 3431, 3792	
\int_if_exist:NTF 1419, 1466, 1545, 2767, 2777, 3237	
\int_incr:N 2576, 2819, 2972, 3165, 3276, 3586, 3620	
\int_mod:nn . . . . .	3424, 3785
\int_new:N 20, 21, 22, 23, 24, 37, 38, 60, 76, 88, 97, 105, 118, 119, 126, 127, 128, 129, 130, 131, 132, 143, 160, 161, 162, 163, 164, 1421, 1468	
\int_set:Nn 544, 548, 550, 1548, 1560, 1747, 1919, 3011, 3012, 3034, 3045, 3051, 3067, 3407, 3457, 3468, 3474, 3490, 3768, 3914	
\int_set_eq:NN . 1528, 1540, 1552, 1566, 1604, 2617, 3074, 3201, 3497	
\int_step_function:nnN . . . . .	1996, 2010, 2025
\int_step_inline:nnn . . . . .	3013, 3941
\int_to_roman:n . . . . .	192, 1976, 2014
\int_use:N . . 921, 1524, 1550, 1562, 1600, 2497, 2515, 2538, 2603, 2702, 2711, 2726, 2732, 3049, 3050, 3062, 3472, 3473, 3485	
\int_zero:N . . . . .	3416, 3777
\c_one_int . 3034, 3053, 3059, 3065, 3069, 3072, 3457, 3476, 3482, 3488, 3492, 3495	
\l_tmpa_int . . . . .	1548, 1552
\c_zero_int . . 222, 233, 1980, 1990, 2002, 2003, 2018, 2020, 3257, 3267, 3427, 3788	
\item . . . . .	28, 39, 40, 60, 70, 83, 85, 86, 88, 96
\item 70, 71, 91, 92, 97, 99, 246, 1790, 1796, 1821, 1829, 1916, 2155, 2158, 2362, 2396, 3152, 3154, 3573, 3575, 3685	
\item* . . . . .	6, 12, 2394
item-pos* . . . . .	2270
item-sym* . . . . .	2270
\itemindent . . . . .	23, 74
\itemindent . . . . .	73
itemindent . . . . .	634
\itemsep . . . . .	84, 85
\itemsep . . . . .	2987, 2993
\itemwidth . . . . .	3041, 3085, 3089, 3464, 3508, 3512
K	
keyans . . . . .	11, 2795
keyans* . . . . .	11, 3562
keyanspic . . . . .	12, 2948
Keys for environments provide by enumext:	
above*	24, 49, 50
above . . . . .	24, 49, 50, 78, 81, 90, 97
after . . . . .	38–40, 80, 83, 90, 97

align	24, 31, 32, 72, 93
before*	38, 39, 78, 90, 97
before	38, 39, 81
below*	24, 49-51
below	24, 49-51, 80, 83, 90, 97
check-ans	24, 25, 27, 56, 62, 67, 68, 70, 71, 78-80, 94, 105
columns-sep*	25, 59, 77, 89
columns-sep	40, 60, 77, 78, 82, 89
columns*	25, 59, 77, 89
columns	23, 40, 43, 49, 60, 77, 78, 82, 89
first	38-40, 93
font	31, 72, 93
item-pos*	61, 63, 69
item-sym*	23, 61, 63, 69, 70
item*-sep	70
itemindent	24, 36, 37, 72, 93
itemsep	36, 75
labelsep	31, 70, 74, 93
labelwidth	30, 31, 34, 35, 74
label	23, 30, 31, 34, 35, 86
lisparindent	75
list-indent	23, 36, 37, 85
list-offset	36, 37
listparindent	36, 93
mark-ans	25, 58, 65
mark-pos	58
mark-ref	25, 58, 63
mini-env	24, 40, 43, 48, 49, 69, 78, 81, 87, 90, 95, 97
mini-sep	24, 40, 78, 81
miniright*	24, 40
miniright	24, 40, 47, 88
minirigth*	27
minirigth	27
no-store	25, 56, 57
noitemsep	36, 44
nosep	36, 44
parindent	75
parsep	36, 75, 93
partopsep	36
ref	26, 32, 33
resume*	51, 53, 54
resume	23, 51-54, 74, 80, 90
rightmargin	36
save-ans	24, 51-54, 59, 60, 62, 65, 67, 71, 80, 81, 83, 90, 97, 99, 101, 102, 105
save-key	25, 53
save-ref	25, 29, 58, 62, 63, 66, 67, 72, 99
save-sep	58
series	51, 53, 54
show-ans	25, 58, 61, 62, 65, 72, 99
show-length	27, 38, 74, 105
show-pos	25, 58, 61, 62, 65, 68, 72, 99
start	24, 27, 35, 74
store-brk	61, 62
topsep	36
widest	23, 27, 35
wrap-ans	58, 61, 65
wrap-label*	31, 70, 72, 91, 93, 98
wrap-label	31, 72, 91, 93, 98
wrap-opt	58
keys commands:	
\keys_define:nn	357, 379, 408, 489, 509, 525, 572, 592, 636, 655, 712, 721, 800, 817, 1236, 1345, 1355, 1616, 1690, 1726, 1744, 1862, 2272, 3821, 3884
\l_keys_key_str	3975
\keys_set:nn	371, 824, 1241, 1246, 1509, 1573, 1586, 1611, 1908, 2598, 2602, 2834, 3187, 3602, 3886, 3887, 3888, 3889, 3890, 3891, 3892, 3893, 3894, 3895, 3896, 3897, 3935
keyval commands:	
\keyval_parse:NNn	1428
L	
label	487, 507, 525
Labels provide by enumext:	
\Alph*	30, 31
\Roman*	30, 31
\alph*	30, 31
\arabic*	30-32
\roman*	30, 31
\labelsep	85
\labelsep	2988, 2991
labelsep	355
\labelwidth	30, 85
\labelwidth	2988, 2989
labelwidth	355
\leftmargin	23, 74
\leftmargin	73, 2988
legacy commands:	
\legacy_if:nTF	3343, 3346, 3710, 3713
\legacy_if_gset_false:n	914
\legacy_if_set_false:n	3345, 3712
\legacy_if_set_true:n	3305, 3330, 3337, 3350, 3649, 3677, 3717
\linewidth	78, 81
\linewidth	2686, 2844, 3010, 3037, 3098, 3460, 3521
\list	28
\list	244
list-indent	634
list-offset	634
\listparindent	2990
listparindent	634
\lrbox	3360, 3723
M	
\makebox	86
\makebox	1852, 1854, 2416, 3374, 3382, 3386, 3736, 3740
\makelabel	70, 72, 73, 86
\makelabel	72, 73, 2422, 2438
\makesavenoteenv	277
mark-ans	1688
mark-pos	1688, 1724
mark-ref	1688
mini-env	798
mini-sep	798
\minipage	28
\minipage	250
\miniright	10, 48, 1182, 2755, 2897
\miniright*	10
mode commands:	
\mode_if_vertical:TF	869, 897, 1020, 1099
\mode_leave_vertical:	667, 681, 693, 705, 1821, 1829, 1850, 2414, 3372
msg commands:	
\msg_error:nn	2825, 2829, 2916, 2975, 3168, 3589, 3595, 3898
\msg_error:nnn	1186, 1190, 1215, 1232, 1370, 1381, 1513, 1589, 3811, 3816, 3881, 3951
\msg_error:nnnn	1876, 1880, 1884, 2817, 2912, 2920
\msg_fatal:nn	2578



\msg_fatal:nnn	309	\prg_replicate:nn	203, 4030
\msg_info:nnn	13, 16, 259, 273	\prg_return_false:	198
\msg_line_context:	3979, 3984, 3989, 4004, 4019, 4023, 4035, 4039, 4043, 4048, 4053, 4058, 4063, 4067, 4072, 4077, 4081, 4086, 4090, 4095, 4100, 4105, 4109, 4113	\prg_return_true:	197
\msg_new:nnn	3952, 3956, 3960, 3964, 3969, 3973, 3977, 3982, 3987, 4002, 4017, 4021, 4025, 4032, 4037, 4041, 4046, 4051, 4056, 4061, 4065, 4070, 4075, 4079, 4084, 4088, 4093, 4098, 4103, 4107, 4111	\printkeyans	13, 102, 3849
\msg_term:nnn	1680	prop commands:	
\msg_term:nnnn	2505, 2515, 2544, 2549	\prop_count:N	1772, 2032, 2068, 2136, 2244, 2257, 3699
\msg_warning:nn	2754, 2896	\prop_gput_if_not_in:Nnn	1767, 1770
\msg_warning:nnn	1394, 1404, 1683	\prop_if_exist:NTF	1409, 3815
\msg_warning:nnnn	2196, 2454, 2459, 3048, 3061, 3471, 3484	\prop_item:Nn	3818
\multicolsep	78, 82	\prop_new:N	1411
\multicolsep	2716, 2869	\ProvidesExplPackage	4
<b>N</b>			
\NeedsTeXFormat	3	<b>R</b>	
\newcounter	312	\raggedcolumns	2725, 2875
\NewDocumentCommand	1182, 1872, 2908, 3795, 3849, 3905	\ref	63, 66
\NewDocumentEnvironment	2554, 2795, 2948, 3139, 3562	ref	487, 507
\newlabel	29	\refstepcounter	3352, 3719
\newlabel	295	regex commands:	
no-store	1614	\regex_match:nnTF	196, 547, 549, 561, 563, 2615, 2628, 3199, 3212
\noindent	88, 96	\regex_replace_once:nnN	423
\noindent	2693, 2849, 3107, 3153, 3415, 3530, 3574, 3776	\renewcommand	435, 456, 466
\nointerlineskip	2693, 2849, 3107, 3530	\RenewDocumentCommand	2294, 2362, 2396, 2422, 2438
noitemsep	590	\RequirePackage	17
\nopagebreak	880, 908, 1031, 1110, 1173, 1179	resume	1345
\normalfont	2065, 2241, 2254, 3696	resume*	1345
nosep	590	rightmargin	634
<b>P</b>			
Packages:		\Roman	31, 35
enumext	22, 51, 73, 74, 83, 104	\Roman	331
enumitem	30	\roman	31, 35
expl3	86	\roman	332, 505, 3837
footnotehyper	29	<b>S</b>	
hyperref	25, 27–29, 33, 63, 67, 92, 104	save-ans	1345
lua-visual-debug	43	save-ref	1688
multicol	22, 104	save-sep	1688
shortlst	86	scan commands:	
\par	880, 908, 1031, 1110, 1173, 1179, 1208, 1225, 2044, 2737, 2759, 2886, 2901, 3022, 3125, 3132, 3415, 3429, 3548, 3555, 3776, 3790	\scan_stop:	85, 3001, 3152, 3573, 3806, 3809
\parindent	3392, 3746	seq commands:	
\parsep	41, 44, 84, 85	\seq_clear:N	3912
\parsep	1822, 1830, 2535, 2987, 2994, 2999	\seq_const_from_clist:Nn	3900
parsep	590	\seq_count:N	2961, 3916
\parskip	3393, 3747	\seq_gclear:N	2292, 2293
\partopsep	85	\seq_gput_right:Nn	1779, 2305, 2306
\partopsep	2536, 2992	\seq_if_empty:NTF	2311, 3864, 3930
partopsep	590	\seq_if_exist:NTF	1413, 3862
peek commands:		\seq_item:Nn	3019
\peek_meaning:NTF	3282, 3296, 3313, 3324, 3626, 3640, 3657	\seq_map_function:NN	3921
\peek_meaning_remove:NTF	3289, 3633	\seq_map_inline:Nn	3870, 3875, 3909, 3931, 3932
\peek_remove_spaces:n	2400	\seq_map_pairwise_function:NNN	2313
\phantomsection	29	\seq_new:N	98, 99, 116, 144, 145, 1415
\phantomsection	284	\seq_pop_left:NN	3920
prg commands:		\seq_put_right:Nn	2922, 3928, 3945
\prg_do_nothing:	288	\seq_set_from_clist:Nn	3913
\prg_new_protected_conditional:Npnn	194	\seq_set_map_e:NNn	3922
		\seq_show:N	3866
		series	1345
		\setcounter	558, 562, 564, 2497, 2538, 2966
		\setenumext	6–9, 103, 3825, 3830, 3835, 3840, 3845, 3905
		\setlength	1823, 1831
		show-ans	1688, 1724
		show-length	710
		skip commands:	
		\skip_add:Nn	849, 855, 861, 871, 875, 899, 903, 1000, 1006, 1012, 1022, 1026, 1048, 1101, 1105, 2987

<code>\skip_eval:n</code>	1822, 1830
<code>\skip_gset:Nn</code>	1121, 1125, 1129
<code>\skip_gzero_new:N</code>	1116, 1117
<code>\skip_horizontal:N</code>	682, 694, 706, 3375, 3389, 3743
<code>\skip_horizontal:n</code>	668, 1851, 1859, 2415, 2417, 3373, 3751
<code>\skip_if_eq:nnTF</code>	847, 853, 859, 923, 957, 998, 1004, 1010, 1041, 1046, 1067, 1118, 1140, 1253, 1267, 1281, 1292, 1303, 1314, 1325, 1336
<code>\skip_new:N</code>	56, 57, 61, 62, 63, 64, 65, 120, 174
<code>\skip_set:Nn</code>	832, 836, 885, 889, 926, 930, 934, 941, 945, 949, 960, 965, 969, 975, 980, 985, 1043, 1044, 1045, 1052, 1056, 1060, 1069, 1074, 1078, 1081, 1085, 1089, 1120, 1124, 1142, 1146, 1150, 1156, 1160, 1164, 2981, 2995
<code>\skip_set_eq:NN</code>	2495, 2534, 2535, 3392, 3393, 3746, 3747
<code>\skip_use:N</code>	834, 838, 873, 877, 881, 901, 905, 924, 943, 952, 958, 963, 967, 978, 982, 983, 988, 1024, 1028, 1054, 1254, 1258, 1261, 1268, 1272, 1275, 2737
<code>\skip_zero:N</code>	2536, 2716, 2869, 2992, 2993
<code>\skip_zero_new:N</code>	1036, 1037, 1038, 1115, 1137, 1138, 1139
<code>\c_zero_skip</code>	847, 853, 859, 924, 958, 998, 1004, 1010, 1041, 1046, 1067, 1118, 1140, 1254, 1268, 1281, 1292, 1303, 1314, 1325, 1336
<code>\small</code>	3827, 3832, 3837, 3842, 3847
<code>\star</code>	2276
<code>start</code>	570
<code>\stepcounter</code>	2298, 2929
str commands:	
<code>\c_backslash_str</code>	4039, 4048, 4049, 4053, 4054, 4058, 4059, 4090, 4091, 4095, 4100, 4101
<code>\c_colon_str</code>	2031, 2135, 3806
<code>\str_case:nn</code>	216
<code>\str_case:nnTF</code>	1435, 1444
<code>\str_clear:N</code>	2595, 3186
<code>\str_count:n</code>	203, 4030
<code>\str_if_empty:N</code>	1458
<code>\str_if_eq:nnTF</code>	2498, 2540
<code>\str_if_in:nnTF</code>	3802
<code>\str_new:N</code>	115, 169
<code>\str_set:Nn</code>	411, 412, 413, 1705, 1706, 1729, 1730
<code>\string</code>	277
<code>\strutbox</code>	928, 932, 936, 947, 951, 962, 971, 977, 987, 1000, 1006, 1012, 1043, 1044, 1045, 1048, 1058, 1062, 1071, 1078, 1083, 1091, 1120, 1121, 1124, 1131, 1144, 1152, 1158, 1166, 2997

T

TeX and  $\LaTeX$  commands:

<code>\@auxout</code>	293
<code>\@currentenv</code>	216
<code>\protected@write</code>	293
text commands:	
<code>\text_expand:n</code>	3798
<code>\textasteriskcentered</code>	1702, 1716
<code>\thepage</code>	299
tl commands:	
<code>\c_space_tl</code>	2223, 3989, 4004
<code>\tl_clear:N</code>	384, 390, 1905, 2078, 2152, 3663
<code>\tl_clear_new:N</code>	341
<code>\tl_const:Nn</code>	146, 325
<code>\tl_gclear:N</code>	1482, 1493, 2198, 2433, 2792, 3136, 3376, 3444, 3559

<code>\tl_gclear_new:N</code>	1463
<code>\tl_gput_right:Nn</code>	326
<code>\tl_greplace_all:Nnn</code>	347
<code>\tl_gset:Nn</code>	1464, 1483, 1494, 2186, 2745, 3248, 3319
<code>\tl_gset_eq:NN</code>	343, 2343, 3369
<code>\tl_if_blank:nTF</code>	3367
<code>\tl_if_empty:N</code>	441, 475, 481, 1368, 1379, 1570, 1608, 1786, 1817, 1931, 2089, 2162, 2194, 2217, 2412, 2772, 3666, 3943
<code>\tl_if_empty:nTF</code>	1499, 1537, 1578
<code>\tl_if_exist:N</code>	1505, 1583
<code>\tl_if_novalue:nTF</code>	1906, 1917, 2086, 2160, 2202, 2296, 2321, 2339, 2344, 2373, 2593, 2959, 3184, 3600, 3664, 3907
<code>\tl_map_inline:Nn</code>	344, 421
<code>\tl_new:N</code>	34, 39, 40, 41, 43, 44, 77, 78, 79, 85, 86, 87, 89, 90, 91, 92, 93, 95, 96, 102, 103, 113, 114, 125, 135, 136, 137, 140, 148, 149, 152, 153, 168, 171, 1531
<code>\tl_put_left:Nn</code>	1794, 1827, 1914, 2229, 2263, 3682, 3685
<code>\tl_put_right:Nn</code>	342, 433, 454, 464, 1748, 1755, 1798, 1833, 1916, 1922, 1930, 1933, 1943, 1948, 1951, 1957, 1983, 1993, 2007, 2023, 2029, 2034, 2081, 2084, 2091, 2093, 2120, 2125, 2130, 2133, 2142, 2155, 2158, 2164, 2166, 2176, 2620, 2633, 3204, 3217, 3668, 3669, 3823, 3828, 3833, 3838, 3843
<code>\tl_remove_all:Nn</code>	3942
<code>\tl_remove_once:Nn</code>	1971, 2105
<code>\tl_replace_all:Nnn</code>	346
<code>\tl_reverse:N</code>	1970, 1972, 2104, 2106
<code>\tl_set:Nn</code>	154, 311, 385, 389, 394, 395, 429, 448, 665, 679, 691, 703, 1367, 1378, 1543, 2062, 2204, 2238, 2251, 2341, 3671, 3693, 3940
<code>\tl_set_eq:NN</code>	352, 430, 432, 451, 453, 461, 463, 1969, 2103, 2116, 2385, 2389, 2940, 2942
<code>\tl_to_str:n</code>	1505, 1510, 1545, 1583, 1587, 3798
<code>\tl_trim_spaces:n</code>	342, 3928, 3940, 3946
<code>\tl_use:N</code>	348, 351, 443, 477, 483, 736, 740, 744, 748, 752, 756, 760, 764, 768, 772, 776, 780, 784, 788, 792, 796, 1856, 1976, 1984, 1995, 2009, 2014, 2026, 2328, 2334, 2358, 2376, 2380, 2388, 2424, 2425, 2432, 2440, 2441, 2447, 2562, 2801, 2945, 3130, 3379, 3390, 3394, 3553, 3733, 3744, 3750, 3754, 3852, 3853, 3854, 3855, 3856, 3924

token commands:

<code>\token_to_str:N</code>	295
<code>\topsep</code>	1823, 1831
<code>topsep</code>	590
<code>\typeout</code>	226, 237, 263, 266, 276, 277, 1481, 1492, 1644, 1663, 2587, 3178

U

<code>\u</code>	424
use commands:	
<code>\use:N</code>	204, 2429, 2564
<code>\use:n</code>	1426, 3804
<code>\use_none:nn</code>	287
<code>\usecounter</code>	2496, 2537

V

<code>\value</code>	2765, 2774, 2779, 3236, 3241
<code>\vspace</code>	915, 1258, 1261, 1272, 1275, 1285, 1287, 1296, 1298, 1307, 1309, 1318, 1320, 1329, 1331, 1340, 1342, 1822, 1830, 2956, 2967, 3430, 3791

W

widest .....	<a href="#">570</a>	wrap-label .....	<a href="#">355</a>
wrap-ans .....	<a href="#">1688</a>	wrap-label* .....	<a href="#">355</a>
		wrap-opt .....	<a href="#">1688</a>