

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-21<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>2</b>	<b>4</b>	<b>The storage system</b>	<b>9</b>
1.1	Description and usage	3	4.1	Keys for storage	9
1.2	The concept of left margin	3	4.2	Keys for internal label and ref	10
1.3	User interface	3	4.3	Keys for debugging and checking	10
1.3.1	Internal counters	3	4.4	The command <code>\anskey</code>	10
1.3.2	Support for <code>multicol</code>	4	4.5	The environment <code>keyans</code>	11
1.3.3	Support for <code>minipage</code>	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	12
1.3.5	Support for <code>\footnote</code>	4	4.6.1	The command <code>\anspic</code>	12
<b>2</b>	<b>The environment <code>enumext</code></b>	<b>4</b>	4.7	Printing stored content	13
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	13
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	13
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>5</b>	<b>5</b>	<b>Full examples</b>	<b>14</b>
3.1	Keys for <code>label</code> and <code>ref</code>	6	<b>6</b>	<b>The way of non-enumerated lists</b>	<b>16</b>
3.2	Keys for spaces	6	<b>7</b>	<b>References</b>	<b>18</b>
3.2.1	Vertical spaces	7	<b>8</b>	<b>Change history</b>	<b>18</b>
3.2.2	Horizontal spaces	7	<b>9</b>	<b>Index of Documentation</b>	<b>19</b>
3.3	Keys for <code>add code</code>	8	<b>10</b>	<b>Implementation</b>	<b>21</b>
3.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	8	<b>11</b>	<b>Index of Implementation</b>	<b>110</b>
3.5	Keys for <code>multicols</code>	9			
3.6	Keys for <code>minipage</code>	9			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

## Motivation and acknowledgments

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

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

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all  $\text{\LaTeX}$  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 2024-05-21.

<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 (l<sup>pp</sup>l), version 1.3 or later (<https://www.latex-project.org/l<sup>pp</sup>l.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[3] package, need to have a modern T<sub>E</sub>X distribution such as T<sub>E</sub>X Live or MiK<sub>T</sub>E<sub>X</sub>. 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 (C) value

(B) correct (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 (D) I and III only

(B) II only (E) I, II, and III

(C) I and II only

★ 3. Third type of questions


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

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


(A) value (D) value

(B) value (E) value


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




(A)




(B)



(C)



(D)



(E)
5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value



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

1. (B),  $x = 5$

2. (D)

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

5. (D), “other note”

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

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

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.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

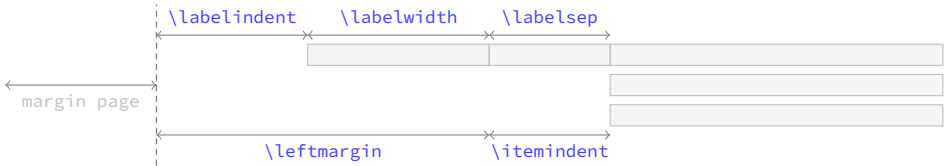


Figure 1: Representation of horizontal lengths in `enumitem`.

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

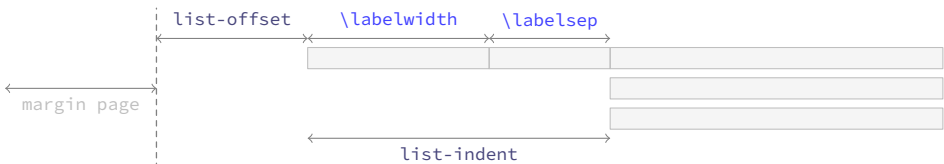


Figure 2: Representation of horizontal lengths concept in `enumext`.

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



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

1.3 User interface

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

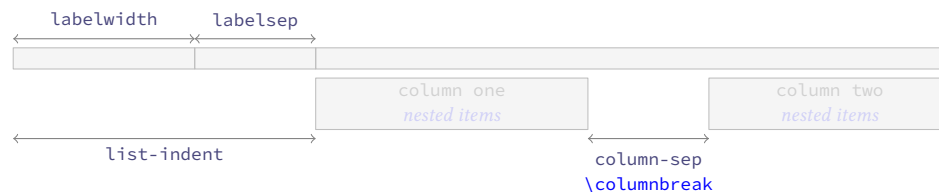


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 `\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 lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and *keyans* and *keyanspic* environments. All *keys* related to vertical or horizontal spacing accept a “*skip*” or “*dim*” expression if passed between braces, i.e. you do not need to use `\dimeval` or `\dimexpr` to perform calculations.

### 3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3.2 Keys for spaces

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

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



### 3.2.1 Vertical spaces

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

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

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

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

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

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

- The value of this parameter also affects the *inner levels* and the `keyans` environment. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T<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, series and resume

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

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

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

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

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

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

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

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

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

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

### 3.5 Keys for multicols

`columns = {⟨integer⟩}` default: 1

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

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

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

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

### 3.6 Keys for minipage

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

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

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

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

#### 3.6.1 The command `\miniright`

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

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

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

## 4 The storage system

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

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

### 4.1 Keys for storage

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

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

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

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

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

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

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

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

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

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

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

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

## 4.2 Keys for internal label and ref

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

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

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

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

## 4.3 Keys for debugging and checking

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

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

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

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

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

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

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

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

## 4.4 The command \anskey

---

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

---

The `\anskey` command takes a mandatory argument and is triggered by `save-ans` key. The “content” are “stored” in *<store name>* set by `save-ans` key. The command does “not support” verbatim content and must NOT be nested. By design it is assumed that each `\item` or `\item*` will have a “single” occurrence of the command unless a nested level is opened or the `no-store` key is used. If `save-ref` key are active and the `hyperref`[7] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by L<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;">* first answer</li> <li>2. Text containing our instructions or questions.</li> <li style="margin-left: 20px;">(a) Question.</li> <li style="margin-left: 40px;">* second answer</li> </ul> | <ul style="list-style-type: none"> <li>3. Text containing our instructions or questions.</li> <li style="margin-left: 20px;">* third answer</li> <li>4. Text containing our instructions or questions.</li> <li style="margin-left: 20px;">* fourth answer</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 \begin{keyans}[\key = val] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}
keyans* \begin{keyans*}[\key = val] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans*}
```

---

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

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

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

### 4.5.1 The `\item*` in `keyans`

---

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

---

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

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

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

### Example

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

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

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

1. Text containing a question.

(A) Choice

\* (B) Correct choice

(C) Choice


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

(A) Choice

(B) Choice

(C) Choice

(D) Choice

\* (E) [note] Correct choice
- 

Some text

4.6 The environment keyanspic

keyanspic

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

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

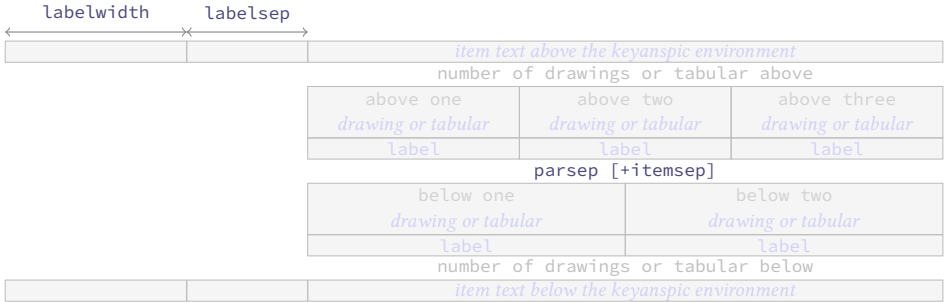


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic


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

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


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

Example


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

1. Question with images.
- 


(A)




(B)



(C)



(D)



\* (E)[note]
- ©2024 by Pablo González L
- 13 / 122

4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

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

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1]  $3(x + y + z)$

2. True False

(a) ~~LaTeXe~~ is cool?

[2] Very True!

3. Related to Linux

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

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- \*

\*

\*

\*

\*



5 Full examples

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

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

and then you can use the excellent [arara](#)<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

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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


C

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

D

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

Example 2

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

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

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

☒ A

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

B

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

C

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

D

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

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

☒ A

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

B

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

C


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

D

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

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

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



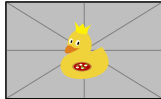
D



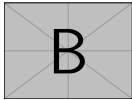
B



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 .

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

Here we set the label key using the default settings in L<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 2024-05-21** – First public release.

9 Index of Documentation

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

C

Document class:

article2

book2

exam3

letter2

report2

\columnbreak5

\columnsep10

Commands provide by enumext:

\anskey4, 10–12

\anspic\*4, 10, 11, 13

\anspic11, 13

\getkeyans4, 11, 14

\item\*4–7, 10–12

\item6, 7, 9–12

\miniright4, 5, 10

\printkeyans4, 6, 11, 14

\setenumext4, 6, 7, 10–12, 14

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

enumext\*4, 5, 9, 10

enumext4–6, 9–12, 14, 17

keyans\*4, 5, 10, 11

keyanspic4, 7, 10, 11, 13, 17

keyans4–13, 17

Environments:

enumerate1, 3, 4, 6, 19

list4, 9, 19

minipage3–5, 10, 19

multicols3, 5, 10

I

\item4, 5

\itemsep8

K

Keys for environments provide by enumext:

above\*8

above8

after9, 10

align7, 18

before\*9

before9

below\*8

below8

check-ans11

columns-sep5, 10

columns5, 8, 10

first9

font7

L

\label5

Labels provide by enumext:

\Alph\*7, 12

\Roman\*7

\alph\*7

\arabic\*7

\roman\*7

\labelsep4, 7

\labelwidth4, 7

\linewidth10

\listparindent8

P

Packages:

enumerate18

enumext1–4, 13, 18, 19

enumitem4, 5, 9, 18, 19

item-pos\*6

item-sym\*6

itemindent8

itemsep8, 13

labelsep4, 6–11, 18

labelwidth4, 6, 7, 9–11, 18

label7, 9, 11, 12, 14, 17, 18

list-indent4, 8, 9

list-offset4, 8, 18

listparindent8

mark-ans11

mark-pos11

mark-ref11

mini-env5, 8, 10

mini-sep5, 10

miniright\*10

miniright10

no-store11

noitemsep8

nosep8, 17

parsep8, 13

partopsep8

ref5, 7

resume\*9

resume\*9

resume9

rightmargin8

save-ans5, 9–14

save-key9

save-ref5, 7, 11, 14

save-sep11

series9

show-ans11

show-length7

show-pos11, 14

start9

topsep8

widest7

wrap-ans10

wrap-label\*7, 18

wrap-label7

wrap-opt10



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

## 10 Implementation

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

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a  $\text{\TeX}$ pert 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 ( $\text{\LaTeX}$ 3 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  $\text{\LaTeX}$  to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-21}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

### 10.4 Definition of variables

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

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

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

```

27 \bool_new:N \l__enumext_starred_first_level_bool
28 \bool_new:N \l__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_first_level_bool
31 \bool_new:N \l__enumext_keyans_env_bool

```

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

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
\l__enumext_store_ans_bool
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\l__enumext_check_start_line_env_tl
\g__enumext_check_start_line_env_tl
\g__enumext_check_starred_cmd_int
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int
```

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

```
134 \bool_new:N \l__enumext_store_ans_bool
135 \bool_new:N \l__enumext_check_ans_bool
136 \bool_new:N \g__enumext_check_ans_show_bool
137 \bool_new:N \g__enumext_check_ans_show_h_bool
138 \tl_new:N \l__enumext_check_start_line_env_tl
139 \tl_new:N \g__enumext_check_start_line_env_tl
140 \int_new:N \g__enumext_check_starred_cmd_int
141 \int_new:N \g__enumext_count_item_anskey_int
142 \int_new:N \g__enumext_count_item_number_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`.

```
143 \bool_new:N \l__enumext_hyperref_bool
144 \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.

```

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

```

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

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

Internal variables used for redefinition of `\footnote`.

```

153 \int_new:N \g__enumext_footnote_int
154 \seq_new:N \g__enumext_footnote_arg_seq
155 \seq_new:N \g__enumext_footnote_int_seq

```

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

```

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

```

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

```

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

```

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

```

(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.

```

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

```

(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.

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

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

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

```
189 \cs_new:Nn \__enumext_level:
190 {
191   \int_to_roman:n { \__enumext_level_int }
192 }
```

(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`

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

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

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

```
199 \cs_new_protected:Nn \__enumext_regex_counter_style:
200 {
201   \tl_map_inline:Nn \c__enumext_counter_style_tl
202   {
203     \regex_replace_once:nnN { \c{##1}\* }
204     { \c{##1}\cB{\u{\__enumext_ref_the_count_tl}\cE} } \__enumext_ref_key_arg_tl
205   }
206 }
```

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

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

```
207 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
208 {
209   * ~ #2
210   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
211   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
212 }
```

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

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

```
213 \cs_set_protected:Nn \__enumext_zero_count_level:
214 {
215   \cs_set_protected:Npn \__enumext_tmp:n ##1
216   {
217     \int_gzero:c { g__enumext_count_level_##1_int }
218   }
219   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
220 }
```

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

`\__enumext_is_not_nested:` The function `\__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “true” only if the environments `enumext` and `enumext*` are nested in each other.

```

221 \cs_new_protected:Nn \__enumext_is_not_nested:
222 {
223   \str_case:en { \@currenvir }
224   {
225     {enumext}
226     {
227       \bool_lazy_and:nnT
228       { \bool_not_p:n { \g__enumext_standar_bool } }
229       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
230       {
231         \bool_gset_true:N \g__enumext_standar_bool
232         \typeout{working-on-enumext}
233       }
234     }
235     {enumext*}
236     {
237       \bool_lazy_and:nnT
238       { \bool_not_p:n { \g__enumext_starred_bool } }
239       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
240       {
241         \bool_gset_true:N \g__enumext_starred_bool
242         \typeout{working-on-enumext*}
243       }
244     }
245   }
246 }

```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standard_first_level_bool` and `\l__enumext_standard_first_level_bool` to “true” only if the environment is not nested and we are in the “first level” of it . We will also save the start line number of each environment in the variable `\g__enumext_check_start_line_env_tl` to use in messages related to the `check-ans` key.

```

247 \cs_new_protected:Nn \__enumext_is_on_first_level:
248 {
249   \bool_lazy_all:nT
250   {
251     { \bool_if_p:N \g__enumext_standar_bool }
252     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
253     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
254   }
255   {
256     \bool_set_true:N \l__enumext_standar_first_level_bool
257     \tl_gset:Nx \g__enumext_check_start_line_env_tl
258     {
259       in ~ 'enumext' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
260     }
261   }
262   \bool_lazy_all:nT
263   {
264     { \bool_if_p:N \g__enumext_starred_bool }
265     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
266     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
267   }
268   {
269     \bool_set_true:N \l__enumext_starred_first_level_bool
270     \tl_gset:Nx \g__enumext_check_start_line_env_tl
271     {
272       in ~ 'enumext*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
273     }
274   }
275 }

```

(End of definition for `\__enumext_is_not_nested:` and `\__enumext_is_on_first_level:`)

`\__enumext_keyans_save_start_line:` The function `\__enumext_keyans_save_start_line:` will save the start line number of the environments `keyans`, `keyans*` and `keyanspic` in the variable `\l__enumext _check_start_line_env_tl` to use in the `\__enumext_check_starred_cmd:n` function.

```

276 \cs_new_protected:Nn \__enumext_keyans_save_start_line:

```

```

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

```

(End of definition for `\__enumext_keyans_save_start_line:`)

## 10.6 Copying list and minipage environments

The `list` environment provided by  $\TeX$  has the following plain form:

```

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

```

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

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

```

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

```

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

The `minipage` environment provided by  $\TeX$  has the following (simplified) plain form:

```

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

```

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

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

```

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

```

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

## 10.7 Compatibility with hyperref and footnotehyper

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

```

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

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```
\__enumext_newlabel:nn
```

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

```

#1: \l__enumext_newlabel_arg_one_tl
#2: \l__enumext_newlabel_arg_two_tl

```

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

```

355 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
356 {
357   \protected@write \@auxout { }
358   {
359     \token_to_str:N \newlabel {#1}

```

```

360         {
361             {#2}
362             \bool_if:NT \l__enumext_hyperref_bool
363             { { \thepage } {#2} {#1} }
364             { }
365         }
366     }
367     \__enumext_hypertarget:nn {#1} { }
368     \__enumext_phantomsection:
369 }

```

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

## 10.8 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

#2 : The counter’s name.

```

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

```

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

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

```

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

```

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

## 10.9 Definition of labels

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

```

\__enumext_register_counter_style:Nn

```

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

```

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

```

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

```

\__enumext_label_width_by_box:Nn
\__enumext_label_width_by_box:cv

```

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

```

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

```

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

```

\__enumext_label_style:Nnn
\__enumext_label_style:cvn

```

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

```

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

```

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

## 10.10 Setting keys associated with label

```

font
labelsep
labelwidth
wrap-label
wrap-label*

```

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

```

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

```

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

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

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

```

441 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
442 {

```



```

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

470 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
471 {
472   \keys_define:nn { enumext / #1 }
473   {
474     align .choice:,
475     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
476     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
477     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
478     align .initial:n = left,
479     align .value_required:n = true,
480   }
481 }
482 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for *align*.)

## 10.11 Setting label and ref keys

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

### 10.11.1 Define and set label and ref keys for enumext environment

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

```

label
ref
\l__enumext_label_i_tl 483 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 484 {
\l__enumext_label_iii_tl 485   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 486   {
487     label .code:n = {
488       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
489       { l__enumext_counter_#2_tl } {##1}
490       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
491       \l__enumext_current_widest_dim
492     },
493     label .initial:n = #3,
494     label .value_required:n = true,
495     ref .code:n = \__enumext_standar_ref:n {##1},
496     ref .value_required:n = true,
497   }
498 }
499 \__enumext_tmp:nnn { level-1 } { i } { \arabic*.}
500 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*.}
501 \__enumext_tmp:nnn { level-3 } { iii } { \roman*.}

```

```
502 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }
```

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

```
\__enumext_standar_ref:n
\__enumext_standar_ref:
```

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

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

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

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

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

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

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

```
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
```

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

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

```
\__enumext_starred_ref:n
\__enumext_starred_ref:
```

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

```
548 \cs_new_protected:Npn \__enumext_starred_ref:n #1
549 {
```

```

550 \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
551 \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
552 {
553   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
554   {
555     \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
556   }
557   {
558     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
559     \__enumext_regex_counter_style:
560     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
561     \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
562     {
563       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
564       { \exp_not:V \l__enumext_ref_key_arg_tl }
565     }
566   }
567 }
568 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
569 {
570   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
571   {
572     \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
573   }
574   {
575     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
576     \__enumext_regex_counter_style:
577     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
578     \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
579     {
580       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
581       { \exp_not:V \l__enumext_ref_key_arg_tl }
582     }
583   }
584 }
585 }

```

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

```

586 \cs_new_protected:Nn \__enumext_starred_ref:
587 {
588   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
589   {
590     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
591     {
592       \tl_use:N \l__enumext_renew_the_count_vii_tl
593     }
594   }
595   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
596   {
597     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
598     {
599       \tl_use:N \l__enumext_renew_the_count_viii_tl
600     }
601   }
602 }

```

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

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

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

```

\l__enumext_label_v_tl 603 \keys_define:nn { enumext / keyans }
\l__enumext_label_vi_tl 604 {
605   label .code:n = {
606     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
607     { \l__enumext_counter_v_tl } {#1}
608     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
609     \l__enumext_current_widest_dim
610     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
611     { \l__enumext_counter_vi_tl } {#1}

```

```

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

```

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

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

```

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

```

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

```

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

```

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

## 10.12 Setting start and widest keys

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

```

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

```

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

```

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

```

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

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

```

__enumext_widest_from:nNNn
__enumext_widest_from:nccn
#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.

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

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

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

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

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

### 10.13 Setting keys for vertical spaces

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

```
topsep parttopsep parsep noitemsep nosep
695 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
696 {
697   \keys_define:nn { enumext / #1 }
698   {
699     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
700     topsep .initial:n = {#3},
701     topsep .value_required:n = true,
702     parttopsep .skip_set:c = { l__enumext_parttopsep_#2_skip },
703     parttopsep .initial:n = {#4},
704     parttopsep .value_required:n = true,
705     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
706     parsep .initial:n = {#5},
707     parsep .value_required:n = true,
708     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
709     itemsep .initial:n = {#6},
710     itemsep .value_required:n = true,

```

```

711         noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
712         noitemsep .value_forbidden:n = true,
713         nosepep    .meta:n      = {
714             itemsep = 0pt, parsep= 0pt,
715             topsep = 0pt, partopsep = 0pt,
716         },
717         nosepep    .value_forbidden:n = true,
718     }
719 }

```

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

```

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

```

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

## 10.14 Setting keys for horizontal spaces

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

```

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

```

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

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

```

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

```

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

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `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.

```

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

```

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

## 10.15 Setting show-length key

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

```

815 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
816 {
817   \keys_define:nn { enumext / #1 }
818   {
819     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
820     show-length .initial:n = false,
821   }
822 }
823 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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



## 10.16 Setting before, after and first keys

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

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

(End of definition for before and others.)

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

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

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

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

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

The function \\_\_enumext\_after\_stop\_list: executes the  $\{\langle code \rangle\}$  set by the after key “after” the enumext environment has finished.

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

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

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

(End of definition for \\_\_enumext\_before\_args\_exec: and others.)

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

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

```
855 \cs_new_protected:Nn \__enumext_before_args_exec_v:
856 {
857   \tl_use:N \l__enumext_before_starred_key_v_tl
858 }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 10.17 Setting keys for multicols and minipage

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

mini-sep

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

columns

```

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

```

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

```

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

```

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

## 10.18 Adjustment of vertical spaces for multicols

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

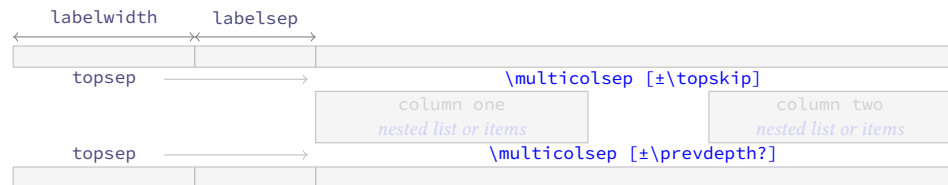


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

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

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

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

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

```

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

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

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

```

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

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

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

```

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

```

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

```

(End of definition for `\__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`.

```

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

```

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

### 10.19 Adjustment of vertical spaces for minipage

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

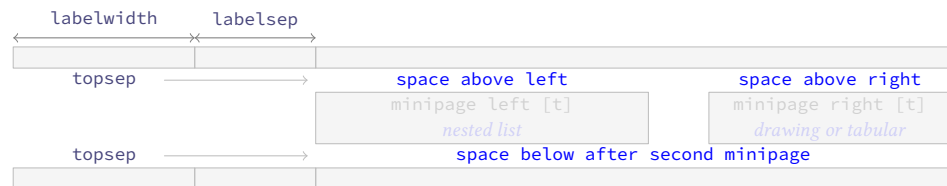


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

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

```

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

```

(End of definition for `__enumext_mini_env*`.)

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

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

We will set the default values taking into account that  $\TeX$  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`.

```

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

```

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

```

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

```

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

```

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

```

```

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

```

(End of definition for \l\_\_enumext\_mini\_set\_vskip:.)

\l\_\_enumext\_zero\_parsep: The function \l\_\_enumext\_zero\_parsep: “adjusted” the value of \l\_\_enumext\_minipage\_after\_skip detecting the value of \l\_\_enumext\_parsep from the previous level. This is necessary since \l\_\_enumext\_parsep from the previous level affects the *vertical spaces* and this is noticeable when using the `nosep` or `noitemsep` keys.

```

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

```

(End of definition for \l\_\_enumext\_zero\_parsep:.)

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

```

1122 \cs_new_protected:Nn \l__enumext_mini_addvspace:
1123 {

```



```

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

```

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

### 10.19.2 Adjustment of vertical spaces for minipage in keyans

\\_\_enumext\_keyans\_mini\_set\_vskip:

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

```

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

```

```

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

```

(End of definition for `\__enumext_keyans_mini_set_vskip:`)

`\__enumext_keyans_mini_addvspace:`

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

```

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

```

(End of definition for `\__enumext_keyans_mini_addvspace:`)

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

`\__enumext_mini_set_vskip_vii:`

`\__enumext_mini_set_vskip_viii:`

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

```

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

```

```

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

```

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

`\__enumext_mini_addvspace_vii:`  
`\__enumext_mini_addvspace_viii:`

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

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

```

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

```

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

#### 10.19.4 The command `\miniright`

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

`\miniright`

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

```

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

```

```

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

```

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

\\_\_enumext\_mini\_right\_cmd:n

The function \\_\_enumext\_mini\_right\_cmd:n takes as argument the *starred version* ‘\*’ of the \miniright command in the enumext environment. We check if the mini-env key is active via the variable \\_\_enumext\_minipage\_right\_X\_dim, if so we close the multicols environment with the \_\_enumext\_\_mini\_env\* environment on the “left side”, then we open the \_\_enumext\_mini\_env\* environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the \centering command when the starred argument ‘\*’ is not present and set zero \g\_\_enumext\_minipage\_stat\_int, otherwise we return an error.

```

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

```

(End of definition for \\_\_enumext\_mini\_right\_cmd:n.)

\\_\_enumext\_keyans\_mini\_right\_cmd:n

The function \\_\_enumext\_keyans\_mini\_right\_cmd:n takes as argument the *starred version* ‘\*’ of the \miniright command in the keyans environment. The implementation of this function is the same as that of the \\_\_enumext\_mini\_right\_cmd:n function of the enumext environment.

```

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

```

(End of definition for \\_\_enumext\_keyans\_mini\_right\_cmd:n.)

## 10.20 Setting above and below keys

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

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

```

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

```

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

### 10.20.1 Functions for above and below keys in enumext

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

```

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

```

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

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

```

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

```

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

### 10.20.2 Functions for above and below keys in keyans

`\__enumext_vspace_above_v:`

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

```

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

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

`\__enumext_vspace_below_v:`

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

```

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

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

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

`\__enumext_vspace_above_vii:`

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

`\__enumext_vspace_above_viii:`

```

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

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

`\__enumext_vspace_below_vii:`

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

`\__enumext_vspace_below_viii:`

```

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

```

```

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

```

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

### 10.21 Setting series, resume and resume\* keys

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

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

```

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

```

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

#### 10.21.1 Internal functions for series key

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

```

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

```

The function `\__enumext_filter_series_key:n` will be responsible for filtering the  $\langle keys \rangle$  that are passed “without value” by excluding the `resume` and `resume*` keys.

```

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

```

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

```

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

```



```

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

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

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

```

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

```

The function `\__enumext_resume_last:n` will be in charge of saving the filtering *(keys)* when the `series` key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the `enumext` environment and in the variable `\g__enumext_starred_series_tl` for the `enumext*` environment. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the `series` key is not being used.

```

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

```

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

### 10.21.2 Internal function to save counter value

```
\__enumext_resume_save_counter:
```

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

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

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

```

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

```

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

### 10.21.3 Internal functions for resume key

`\__enumext_resume_series:n`

The function `\__enumext_resume_series:n` will handle the argument passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function `\__enumext__resume_counter:` is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is not present, if the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key, otherwise it

will verify that the `\g__enumext_series_⟨series name⟩_tl` variable set by the `series` key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```

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

```

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

```

\__enumext_resume_counter:n
\__enumext_resume_counter:
  \__enumext_resume_counter_series:
  \__enumext_resume_counter_save_ans:

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

#### 10.21.4 Internal function for `resume*` key

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

```

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

```

```

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

```

(End of definition for \\_\_enumext\_resume\_starred:.)

## 10.22 Setting save-ans key

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

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

```

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

```

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

### 10.22.1 Internal functions for `save-ans` key

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

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

```

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

```

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

```

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

```

```

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

```

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

## 10.23 The check answer mechanism

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

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

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

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

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

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

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

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

### 10.23.1 Setting check-ans key

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

```

check-ans no-store
1752 \cs_set_protected:Npn \__enumext_tmp:n #1
1753 {
1754   \keys_define:nn { enumext / #1 }
1755   {
1756     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1757     check-ans .initial:n = false,
1758     check-ans .value_required:n = true,
1759     no-store .code:n = {
1760       \bool_set_false:N \l__enumext_store_ans_bool
1761       \bool_set_false:N \l__enumext_check_ans_bool
1762     },
1763     no-store .value_forbidden:n = true,
1764   }
1765 }
1766 \clist_map_inline:nn
1767 {
1768   level-1, level-2, level-3, level-4, enumext*
1769 }
1770 { \__enumext_tmp:n {#1} }

```

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

### 10.23.2 Set-up check answer mechanism

`\__enumext_check_ans_gdecr:` The function `\__enumext_check_ans_gdecr:` 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.

```

1771 \cs_new_protected:Nn \__enumext_check_ans_gdecr:
1772 {
1773   \int_case:nn { \l__enumext_level_int }
1774   {
1775     { 1 }{
1776       \bool_lazy_all:NT
1777       {
1778         { \bool_if_p:N \g__enumext_starred_bool }
1779         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1780       }
1781       {
1782         \int_gdecr:N \g__enumext_count_item_number_int
1783         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1784       }
1785     }
1786     { 2 }{
1787       \int_gdecr:N \g__enumext_count_item_number_int
1788     }
1789     { 3 }{
1790       \int_gdecr:N \g__enumext_count_item_number_int
1791     }
1792     { 4 }{
1793       \int_gdecr:N \g__enumext_count_item_number_int
1794     }
1795   }
1796   \int_case:nn { \l__enumext_level_h_int }
1797   {
1798     { 1 }{
1799       \bool_if:NT \g__enumext_standar_bool
1800       {
1801         \int_gdecr:N \g__enumext_count_item_number_int
1802         \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1803       }
1804     }
1805   }
1806 }

```

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

`\__enumext_check_ans_exec:` The function `\__enumext_check_ans_exec:` will count the number of times the `\item` and `\item*` commands appears per level within the `enumext` environment. The boolean variable `\l__enumext_store_ans_bool` controlled by the `no-store` key will increment the integer variable of the level counter by `1` to preserve the equality that we will use in the final comparison of the process.

```

1807 \cs_new_protected:Nn \__enumext_check_ans_exec:
1808 {
1809   \bool_if:NT \l__enumext_check_ans_bool
1810   {
1811     \tl_if_empty:NTF \l__enumext_store_name_tl
1812     {
1813       \bool_if:NT \l__enumext_standar_first_level_bool
1814       {
1815         \msg_error:nnnn { enumext } { need-save-ans }{ check-ans } { enumext }
1816       }
1817       \bool_if:NT \l__enumext_starred_first_level_bool
1818       {
1819         \msg_error:nnnn { enumext } { need-save-ans }{ check-ans } { enumext* }
1820       }
1821     }
1822     {
1823       \__enumext_check_ans_gdecr:
1824     }
1825   }
1826 }

```

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



`\__enumext_check_ans_to_hook:` The function `\__enumext_check_ans_to_hook:` will count the number of times the `\item` and `\item*` commands appears per level within the `enumext` environment.

```

1827 \cs_new_protected:Nn \__enumext_check_ans_to_hook:
1828 {
1829   \bool_lazy_and:nnT
1830   { \bool_if_p:N \__enumext_check_ans_bool }
1831   { \bool_if_p:N \g__enumext_standar_bool }
1832   {
1833     \bool_gset_true:N \g__enumext_check_ans_show_bool
1834     \tl_gset:NV \g__enumext_store_name_tl \__enumext_store_name_tl
1835   }
1836   \bool_lazy_and:nnT
1837   { \bool_if_p:N \__enumext_check_ans_bool }
1838   { \bool_if_p:N \g__enumext_starred_bool }
1839   {
1840     \bool_gset_true:N \g__enumext_check_ans_show_h_bool
1841     \tl_gset:NV \g__enumext_store_name_tl \__enumext_store_name_tl
1842   }
1843 }

```

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

`\__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.

```

1844 \cs_new_protected:Nn \__enumext_check_ans_show:
1845 {
1846   \int_compare:nNnTF
1847   { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1848   {
1849     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1850   }
1851   {
1852     \msg_warning:nnVV
1853     { enumext } { item-different-answer } \g__enumext_store_name_tl \g__enumext_check_start_line_env_tl
1854   }
1855   \int_gzero:N \g__enumext_count_item_number_int
1856   \int_gzero:N \g__enumext_count_item_anskey_int
1857   %\tl_gclear:N \g__enumext_check_start_line_env_tl
1858 }

```

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

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

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

```

1859 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1860 {
1861   \int_compare:nNnT
1862   { \g__enumext_check_starred_cmd_int } = { 0 }
1863   {
1864     \msg_warning:nnnV
1865     { enumext } { missing-starred } { #1 } \__enumext_check_start_line_env_tl
1866   }
1867   \int_compare:nNnT
1868   { \g__enumext_check_starred_cmd_int } > { 1 }
1869   {
1870     \msg_warning:nnnV
1871     { enumext } { many-starred } { #1 } \__enumext_check_start_line_env_tl
1872   }
1873   \int_gzero:N \g__enumext_check_starred_cmd_int
1874   \tl_clear:N \__enumext_check_start_line_env_tl
1875 }

```

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

## 10.24 Keys and functions associated with storage

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

```

1876 \cs_set_protected:Npn \__enumext_tmp:n #1
1877 {
1878   \keys_define:nn { enumext / #1 }
1879   {
1880     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1881     wrap-ans .initial:n = \fbox{##1},
1882     wrap-ans .value_required:n = true,
1883     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1884     wrap-opt .initial:n = [{##1}],
1885     wrap-opt .value_required:n = true,
1886     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1887     save-sep .initial:n = {, ~ },
1888     save-sep .value_required:n = true,
1889     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
1890     mark-ans .initial:n = \textasteriskcentered,
1891     mark-ans .value_required:n = true,
1892     mark-pos .choice:,
1893     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1894     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1895     mark-pos .initial:n = right,
1896     mark-pos .value_required:n = true,
1897     show-ans .bool_set:N = \__enumext_show_answer_bool,
1898     show-ans .initial:n = false,
1899     show-ans .value_required:n = true,
1900     show-pos .bool_set:N = \__enumext_show_position_bool,
1901     show-pos .initial:n = false,
1902     show-pos .value_required:n = true,
1903     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
1904     mark-ref .initial:n = \textasteriskcentered,
1905     mark-ref .value_required:n = true,
1906     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
1907     save-ref .initial:n = false,
1908     save-ref .value_required:n = true,
1909   }
1910 }
1911 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1912 \cs_set_protected:Npn \__enumext_tmp:n #1
1913 {
1914   \keys_define:nn { enumext / #1 }
1915   {
1916     mark-pos .choice:,
1917     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1918     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1919     mark-pos .initial:n = right,
1920     mark-pos .value_required:n = true,
1921     show-ans .bool_set:N = \__enumext_show_answer_bool,
1922     show-ans .initial:n = false,
1923     show-ans .value_required:n = true,
1924     show-pos .bool_set:N = \__enumext_show_position_bool,
1925     show-pos .initial:n = false,
1926     show-pos .value_required:n = true,
1927   }
1928 }
1929 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1930 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1931 {
1932   \keys_define:nn { enumext / #1 }
1933   {
1934     columns*      .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1935                   \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1936                   \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1937                   {
1938                     columns = \exp_not:v { l__enumext_store_columns_#2_int },
1939                   },
1940     columns*      .value_required:n = true,
1941     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1942                   \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1943                   \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1944                   {
1945                     columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1946                   },
1947     columns-sep* .value_required:n = true,
1948   }
1949 }
1950 \clist_map_inline:nn
1951 {
1952   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1953 }
1954 { \__enumext_tmp:nn #1 }

```

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

#### 10.24.1 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

1955 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1956 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1957 {
1958   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1959   {
1960     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1961   }
1962   { #1 }
1963 }
1964 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

#### 10.24.2 Function for storing content in sequence

```

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

```

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

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

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

```

1965 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1966 {
1967   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1968 }
1969 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

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

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

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

```

1970 \cs_new_protected:Nn \__enumext_store_level_open:
1971 {
1972   \bool_if:NT \l__enumext_store_ans_bool
1973   {
1974     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1975     {
1976       \__enumext_store_addto_seq:n
1977       {
1978         \item \begin{enumext}
1979       }
1980     }
1981     {
1982       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1983       {
1984         \item \begin{enumext} [
1985       }
1986       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1987       {
1988         ]
1989       }
1990       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1991     }
1992   }
1993 }
1994 \cs_new_protected:Nn \__enumext_store_level_close:
1995 {
1996   \bool_if:NT \l__enumext_store_ans_bool
1997   {
1998     \__enumext_store_addto_seq:n { \end{enumext} }
1999   }
2000 }

```

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

\\_\_enumext\_store\_level\_open\_vii:  
 \\_\_enumext\_store\_level\_close\_vii:

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

```

2001 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2002 {
2003   \bool_if:NT \l__enumext_store_ans_bool
2004   {
2005     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
2006     {
2007       \__enumext_store_addto_seq:n
2008       {
2009         \item \mode_leave_vertical:
2010         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2011         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2012       }
2013     }
2014     {
2015       \tl_put_left:Nn \l__enumext_store_opt_vii_tl
2016       {
2017         \item \mode_leave_vertical:
2018         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2019         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2020       }
2021       \tl_put_right:Nn \l__enumext_store_opt_vii_tl
2022       {
2023         ]
2024       }
2025       \__enumext_store_addto_seq:v \l__enumext_store_opt_vii_tl
2026     }
2027   }
2028 }
2029 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2030 {
2031   \bool_if:NT \l__enumext_store_ans_bool
2032   {

```

```

2033     \__enumext_store_addto_seq:n { \end{enumext*} }
2034   }
2035 }

```

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

#### 10.24.4 Function for show marks and position

```

\__enumext_print_keyans_box:NN
\__enumext_print_keyans_box:cc

```

The function \\_\_enumext\_print\_keyans\_box:NN print a box in the left margin with \l\_\_enumext\_mark\_answer\_sym\_tl used by the wrap-ans, show-ans and show-pos keys. The function takes two arguments:

#1: \l\_\_enumext\_labelwidth\_X\_dim

#2: \l\_\_enumext\_labelsep\_X\_dim

```

2036 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2037 {
2038   \mode_leave_vertical:
2039   \skip_horizontal:n { -\dim_use:N #2 }
2040   \makebox[0pt][ r ]
2041   {
2042     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2043     {
2044       \tl_use:N \l__enumext_mark_answer_sym_tl
2045     }
2046   }
2047   \skip_horizontal:n { \dim_use:N #2 }
2048 }
2049 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

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

#### 10.25 The command \anskey and internal label and ref

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

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

```

2050 \keys_define:nn { enumext / anskey }
2051 {
2052   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2053   item-sym* .value_required:n = true,
2054   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2055   item-pos* .value_required:n = true,
2056   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2057   store-brk .default:n = true,
2058   store-brk .value_forbidden:n = true,
2059 }

```

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*.

```

2060 \NewDocumentCommand \anskey { d() s o +m }
2061 {
2062   \bool_if:NF \l__enumext_store_active_bool
2063   {
2064     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2065   }
2066   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2067   {
2068     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2069   }
2070   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2071   {
2072     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2073   }
2074   \group_begin:
2075     \bool_if:NT \l__enumext_store_ans_bool
2076     {
2077       \bool_if:NT \l__enumext_check_ans_bool

```

```

2078         {
2079             \int_gincr:N \g__enumext_count_item_anskey_int
2080         }
2081         \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2082     }
2083     \group_end:
2084 }

```

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

`\__enumext_store_anskey_code:nnnn`

The internal function `\__enumext_store_anskey_code:nnnn` first we pass the command *⟨argument⟩* to the *⟨prop list⟩*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `save-ref` key and will call the function `\__enumext_store_internal_ref:` for the internal “*label and ref*” system. Followed by this if the `show-ans` or `show-pos` keys are active we will show the “*wrapped*” *⟨argument⟩* passed to the command.

```

2085 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2086 {
2087     \__enumext_store_addto_prop:n {#4}
2088     \bool_if:NT \l__enumext_store_ref_key_bool
2089     {
2090         \__enumext_store_internal_ref:
2091     }
2092     \__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`.

```

2093     \tl_clear:N \l__enumext_store_anskey_arg_tl
2094     \tl_if_novalue:nF {#3}
2095     {
2096         \keys_set:nn { enumext / anskey } {#3}
2097     }
2098     \bool_lazy_and:nnT
2099     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2100     { \bool_not_p:n { \l__enumext_starred_bool } }
2101     {
2102         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2103     }
2104     \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).

```

2105     \tl_if_novalue:nF {#1}
2106     {
2107         \int_set:Nn \l__enumext_store_columns_join_int {#1}
2108         \bool_if:NT \l__enumext_starred_bool
2109         {
2110             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2111             {
2112                 ( \exp_not:V \l__enumext_store_columns_join_int )
2113             }
2114         }
2115     }

```

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`.

```

2116     \bool_if:nTF {#2}
2117     {
2118         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2119         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2120         {
2121             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2122             {
2123                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2124             }
2125         }
2126         \dim_compare:nT
2127         {
2128             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim

```

```

2129     }
2130     {
2131         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2132         {
2133             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2134         }
2135     }
2136     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2137 }
2138 {
2139     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2140 }

```

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`.

```

2141 \bool_lazy_and:nnT
2142 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2143 { \bool_if_p:N \l__enumext_hyperref_bool }
2144 {
2145     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2146     {
2147         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2148         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2149     }
2150 }
2151 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2152 }

```

(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`.

```

2153 \cs_new_protected:Nn \__enumext_store_internal_ref:
2154 {
2155     \cs_set_protected:Npn \__enumext_tmp:n ##1
2156     {
2157         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2158         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2159         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2160         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2161     }
2162     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2163     \cs_set:Npn \__enumext_tmp:n ##1
2164     { . \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.

```

2165 \bool_lazy_all:nT
2166 {
2167     { \bool_if_p:N \g__enumext_starred_bool }
2168     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2169 }
2170 {
2171     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2172     { \tl_use:N \l__enumext_label_copy_vii_tl }
2173 }
2174 \bool_lazy_all:nT
2175 {
2176     { \bool_if_p:N \l__enumext_standar_bool }
2177     { \bool_if_p:N \g__enumext_starred_bool }
2178     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2179 }
2180 {
2181     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2182     {
2183         \tl_use:N \l__enumext_label_copy_vii_tl
2184         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2185     }
2186 }

```



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.

```

2187 \bool_lazy_all:nT
2188 {
2189   { \bool_if_p:N \l__enumext_standar_bool }
2190   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2191   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2192   { \bool_not_p:n { \l__enumext_starred_bool } }
2193 }
2194 {
2195   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2196   {
2197     \tl_use:N \l__enumext_label_copy_i_tl
2198     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2199   }
2200 }
2201 \cs_set:Npn \__enumext_tmp:n ##1
2202 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2203 \bool_lazy_all:nT
2204 {
2205   { \bool_if_p:N \l__enumext_standar_bool }
2206   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2207   { \bool_not_p:n { \g__enumext_starred_bool } }
2208   { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2209 }
2210 {
2211   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2212   {
2213     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2214     . \tl_use:N \l__enumext_label_copy_vii_tl
2215   }
2216 }

```

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

```

2217 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2218 {
2219   \l__enumext_store_name_tl \c_colon_str
2220   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2221 }

```

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.

```

2222 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2223 {
2224   \__enumext_newlabel:nn
2225   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2226   { \l__enumext_newlabel_arg_two_tl }
2227 }
2228 \l__enumext_store_write_aux_file_tl
2229 }

```

(End of definition for `\__enumext_store_internal_ref:`)

`\__enumext_store_anskey_show_wrap:n`

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

```

2230 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2231 {
2232   \par
2233   \bool_if:NT \l__enumext_starred_bool
2234   {
2235     \cs_set:Nn \__enumext_level: { vii }
2236   }
2237   \__enumext_print_keyans_box:cc
2238   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2239   { \l__enumext_labelsep_ \__enumext_level: _dim }
2240   \__enumext_anskey_wrapper:n { #1 }
2241 }

```

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

`\__enumext_store_anskey_show_left:n`

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

```

2242 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2243 {
2244   \bool_if:NT \l__enumext_show_answer_bool
2245   {
2246     \__enumext_store_anskey_show_wrap:n { #1 }
2247   }
2248   \bool_if:NT \l__enumext_show_position_bool
2249   {
2250     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2251     {
2252       \group_begin:
2253       \exp_not:N \normalfont
2254       \exp_not:N \footnotesize [ \int_eval:n
2255       {
2256         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2257       }
2258       ]
2259       \group_end:
2260     }
2261     \__enumext_store_anskey_show_wrap:n { #1 }
2262   }
2263 }

```

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

## 10.26 Common functions for `keyans`, `keyans*` and `keyanspic`

### 10.26.1 Storing content in `prop list`

`\__enumext_keyans_addto_prop:n`

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

```

2264 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2265 {
2266   \tl_clear:N \l__enumext_store_keyans_label_tl
2267   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2268   {
2269     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2270   }
2271   {
2272     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2273   }
2274   \tl_if_novalue:nF { #1 }
2275   {
2276     % Set save-sep
2277     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2278     {
2279       \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2280     }
2281     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { #1 }
2282   }
2283   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2284 }

```

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

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

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

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

```

2285 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2286 {
2287   \bool_if:NT \l__enumext_store_ref_key_bool
2288   {
2289     \cs_set_protected:Npn \__enumext_tmp:n ##1
2290     {
2291       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2292       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2293       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2294       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2295     }
2296     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2297     \__enumext_keyans_store_ref_aux_i:
2298   }
2299 }

```

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

```

2300 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2301 {
2302   \bool_if:NT \g__enumext_starred_bool
2303   {
2304     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2305   }
2306   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2307   {
2308     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2309     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2310   }
2311   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2312   {
2313     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2314     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2315   }
2316   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2317   {
2318     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2319     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2320   }
2321   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2322   {
2323     \l__enumext_store_name_tl \c_colon_str
2324     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2325   }
2326   \__enumext_keyans_store_ref_aux_ii:
2327 }

```

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.

```

2328 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2329 {
2330   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2331   {
2332     \__enumext_newlabel:nn
2333     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2334     { \l__enumext_newlabel_arg_two_tl }
2335   }
2336   \l__enumext_store_write_aux_file_tl
2337 }

```

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

### 10.26.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n` 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.

```

2338 \cs_new_protected:Npn \l__enumext_keyans_addto_seq:n #1
2339 {
2340   \tl_clear:N \l__enumext_store_keyans_label_tl
2341   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2342   {
2343     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2344   }
2345   {
2346     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2347   }
2348   \tl_if_no_value:nF { #1 }
2349   {
2350     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2351     {
2352       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2353     }
2354     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2355   }
2356   \l__enumext_keyans_addto_seq_link:
2357 }

```

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

```

2358 \cs_new_protected:Nn \l__enumext_keyans_addto_seq_link:
2359 {
2360   \bool_lazy_and:nnT
2361   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2362   { \bool_if_p:N \l__enumext_hyperref_bool }
2363   {
2364     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2365     {
2366       \hfill \exp_not:N \hyperlink
2367       {
2368         \exp_not:V \l__enumext_newlabel_arg_one_tl
2369       }
2370       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2371     }
2372   }
2373   \l__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2374   \bool_if:NT \l__enumext_check_ans_bool
2375   {
2376     \int_gincr:N \g__enumext_count_item_anskey_int
2377   }
2378 }

```

(End of definition for `\l__enumext_keyans_addto_seq:n` and `\l__enumext_keyans_addto_seq_link:.`)

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

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

`\l__enumext_keyans_show_left:n` Common function to show *starred commands* `\item*` and *(position)* of stored content in *(prop list)* for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_` `\l__enumext_store_name_tl` `_prop` for `show-pos` key.

```

2379 \cs_new_protected:Npn \l__enumext_keyans_show_left:n #1
2380 {
2381   \tl_if_no_value:nF { #1 }
2382   {
2383     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2384   }
2385   \bool_if:NT \l__enumext_show_answer_bool
2386   {
2387     \l__enumext_keyans_show_ans:
2388   }
2389   \bool_if:NT \l__enumext_show_position_bool

```

```

2390     {
2391         \__enumext_keyans_show_pos:
2392     }
2393 }
2394 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2395 {
2396     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2397     {
2398         \bool_lazy_or:nnT
2399         { \bool_if_p:N \l__enumext_show_answer_bool }
2400         { \bool_if_p:N \l__enumext_show_position_bool }
2401         {
2402             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2403         }
2404     }
2405 }
2406 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2407 {
2408     \tl_put_left:Nn \l__enumext_label_v_tl
2409     {
2410         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2411     }
2412 }
2413 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2414 {
2415     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2416     {
2417         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2418         {
2419             \group_begin:
2420             \exp_not:N \normalfont
2421             \exp_not:N \footnotesize [ \int_eval:n
2422                 {
2423                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2424                 }
2425             ]
2426             \group_end:
2427         }
2428     }
2429     {
2430         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2431         {
2432             \group_begin:
2433             \exp_not:N \normalfont
2434             \exp_not:N \footnotesize [ \int_eval:n
2435                 {
2436                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2437                 }
2438             ]
2439             \group_end:
2440         }
2441     }
2442     \tl_put_left:Nn \l__enumext_label_v_tl
2443     {
2444         \__enumext_print_keyans_box:NN
2445         \l__enumext_labelwidth_i_dim
2446         \l__enumext_labelsep_i_dim
2447     }
2448 }

```

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

## 10.27 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` it is best to define a couple of keys that allow us to control and set by default the *symbol* and its *offset*.

`item-sym*` Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```

item-pos* 2449 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
          2450 {
          2451     \keys_define:nn { enumext / #1 }
          2452     {

```

```

2453     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2454     item-sym* .value_required:n = true,
2455     item-sym* .initial:n = {${\star$}},
2456     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2457     item-pos* .value_required:n = true,
2458   }
2459 }
2460 \clist_map_inline:nn
2461 {
2462   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2463 }
2464 { \__enumext_tmp:nn #1 }

```

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

## 10.28 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

2465 \cs_new_protected:Nn \__enumext_footnotetext:nn
2466 {
2467   \footnotetext[#1]{#2}
2468 }
2469 \cs_new_protected:Nn \__enumext_renew_footnote:
2470 {
2471   \seq_gclear:N \g__enumext_footnote_arg_seq
2472   \seq_gclear:N \g__enumext_footnote_int_seq
2473   \RenewDocumentCommand \footnote { o +m }
2474   {
2475     \tl_if_novalue:nTF {##1}
2476     {
2477       \stepcounter{footnote}
2478       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2479     }
2480     {
2481       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2482     }
2483     \footnotemark [ \g__enumext_footnote_int ]
2484     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2485     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2486   }
2487 }
2488 \cs_new_protected:Nn \__enumext_print_footnote:
2489 {
2490   \seq_if_empty:NF \g__enumext_footnote_int_seq
2491   {
2492     \seq_map_pairwise_function:NNN
2493     \g__enumext_footnote_int_seq
2494     \g__enumext_footnote_arg_seq
2495     \__enumext_footnotetext:nn
2496   }
2497 }

```

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

## 10.29 Redefining `\item` command

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

### 10.29.1 The `\item` command in `enumext`

```

\__enumext_default_item:n

```

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

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

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

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

```

2498 \cs_new_protected:Npn \__enumext_default_item:n #1
2499 {
2500   \tl_if_novalue:nTF {#1}
2501   {
2502     \bool_if:NT \__enumext_check_ans_bool
2503     {
2504       \int_gincr:N \g__enumext_count_item_number_int
2505     }
2506     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2507     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2508   }
2509   {
2510     \bool_set_eq:cc
2511     { l__enumext_wrap_label_ \__enumext_level: _bool }
2512     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2513     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2514   }
2515 }

```

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

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

#1: \l\_\_enumext\_item\_symbol\_X\_tl

#2: \l\_\_enumext\_item\_symbol\_sep\_X\_dim

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

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

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

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

```

2516 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2517 {
2518   \tl_if_novalue:nF {#1}
2519   {
2520     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2521   }
2522   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2523   \tl_if_novalue:nTF {#2}
2524   {
2525     \dim_set_eq:cc
2526     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2527     { l__enumext_labelsep_ \__enumext_level: _dim }
2528   }
2529   {
2530     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2531   }
2532   \bool_if:NT \__enumext_check_ans_bool
2533   {
2534     \int_gincr:N \g__enumext_count_item_number_int
2535   }
2536   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2537   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2538 }

```

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

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



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

```

2539 \cs_new_protected:Nn \__enumext_redefine_item:
2540 {
2541   \RenewDocumentCommand \item { s o o }
2542   {
2543     \bool_if:nTF {##1}
2544     {
2545       \__enumext_starred_item:nn {##2} {##3}
2546     }
2547     { \__enumext_default_item:n {##2} }
2548   }
2549 }

```

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

### 10.29.2 The `\item` command in `keyans`

The `\item*` and `\item*[\langle content \rangle]` commands *store* the current  $\langle label \rangle$  next to the  $[\langle content \rangle]$  if it is present in the  $\langle sequence \rangle$  and  $\langle prop list \rangle$  defined by `save-ans` key.

`\__enumext_keyans_default_item:n`

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

```

2550 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2551 {
2552   \tl_if_novalue:nTF { #1 }
2553   {
2554     \bool_set_true:N \__enumext_wrap_label_v_bool
2555     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2556   }
2557   {
2558     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2559     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2560   }
2561 }

```

(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  $\langle label \rangle$ .

```

2562 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2563 {
2564   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2565   \__enumext_keyans_show_left:n { #1 }
2566   \bool_set_true:N \__enumext_wrap_label_v_bool
2567   \__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  $\langle sequence \rangle$ .

```

2568   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2569   \__enumext_keyans_addto_prop:n { #1 }
2570   \__enumext_keyans_store_ref:
2571   \__enumext_keyans_addto_seq:n { #1 }
2572   \int_gincr:N \g__enumext_check_starred_cmd_int
2573 }

```

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

`\item*`

`\__enumext_keyans_redefine_item:`

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

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

```

2574 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2575 {
2576   \RenewDocumentCommand \item { s o }
2577   {
2578     \bool_if:nTF {##1}

```

```

2579         {
2580             \peek_remove_spaces:n
2581             {
2582                 \__enumext_keyans_starred_item:n {##2}
2583             }
2584         }
2585         {
2586             \__enumext_keyans_default_item:n {##2}
2587         }
2588     }
2589 }

```

(End of definition for `\item*` and `\__enumext_keyans_redefine_item:`. This function is documented on page 11.)

### 10.30 Redefining `\makeLabel` command

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

#### 10.30.1 Redefining `\makeLabel` for `enumext`

`\__enumext_item_starred:` The function `\__enumext_item_starred:` will be responsible for executing `\item*` for the `enumext` environment.

```

2590 \cs_new_protected:Nn \__enumext_item_starred:
2591 {
2592     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2593     {
2594         \mode_leave_vertical:
2595         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2596         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2597         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2598     }
2599 }

```

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

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

```

2600 \cs_new_protected:Nn \__enumext_make_label:
2601 {
2602     \RenewDocumentCommand \makeLabel { m }
2603     {
2604         \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2605         \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2606         \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2607         {
2608             \__enumext_item_starred:
2609             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2610         }
2611         { ##1 }
2612         \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2613         \tl_gclear:N \g__enumext_item_symbol_tl
2614     }
2615 }

```

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

#### 10.30.2 Redefining `\makeLabel` for `keyans`

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

```

2616 \cs_new_protected:Nn \__enumext_keyans_make_label:
2617 {
2618     \RenewDocumentCommand \makeLabel { m }
2619     {
2620         \tl_use:N \__enumext_label_fill_left_v_tl
2621         \tl_use:N \__enumext_label_font_style_v_tl
2622         \bool_if:NTF \__enumext_wrap_label_v_bool
2623         {
2624             \__enumext_wrapper_label_v:n { ##1 }
2625         }
2626     }

```

```

2626     { ##1 }
2627     \tl_use:N \l__enumext_label_fill_right_v_tl
2628   }
2629 }

```

(End of definition for `\__enumext_keyans_make_label:`)

### 10.31 Second argument of the lists

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

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

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

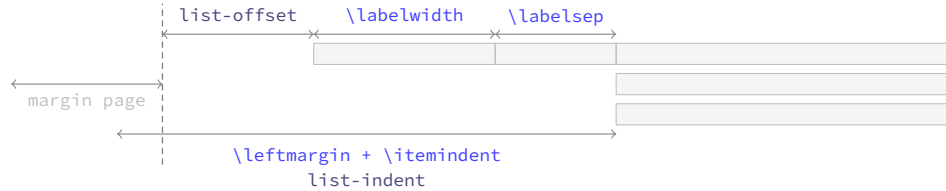


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

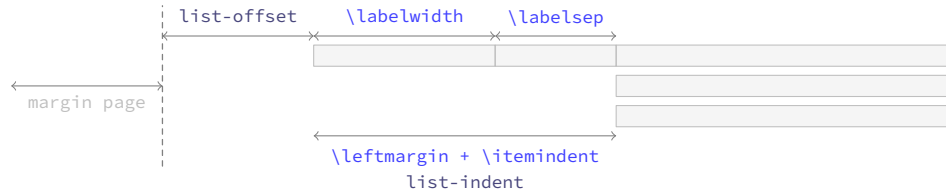


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

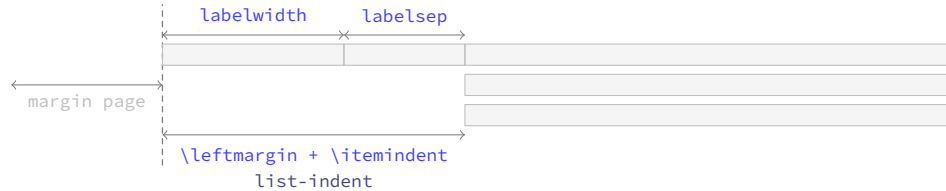


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

```

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

```

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

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

```

2630 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2631 {
2632   \dim_compare:nNtT { #1 } < { \c_zero_dim }
2633   {
2634     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2635     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2636   }
2637   \dim_compare:nNtT { #2 } < { \c_zero_dim }
2638   {
2639     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2640     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2641   }

```

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

```
2642 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2643 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2644 {
2645   \dim_set:Nn #6 { #1 + #2 - #4 }
2646   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2647 }
2648 {
2649   \dim_compare:nNnT { #4 } = { #1 + #2 }
2650   { \dim_set:Nn #6 { \c_zero_dim } }
2651   \dim_compare:nNnT { #4 } < { #1 + #2 }
2652   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2653   \dim_compare:nNnT { #4 } > { #1 + #2 }
2654   {
2655     \dim_set:Nn #6 { -#1 - #2 + #4 }
2656     \dim_set:Nn #6 { #6*-1 }
2657   }
2658   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2659 }
2660 }
2661 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

### 10.31.2 Setting second argument of the lists

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

```
2662 \cs_set_protected:Npn \__enumext_tmp:n #1
2663 {
2664   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
2665   {
2666     \__enumext_calc_hspace:ccccccc
2667     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2668     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2669     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2670     { \__enumext_leftmargin_tmp_#1_bool }
2671   }
2672   \clist_map_inline:nn
2673   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2674   { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2675   \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2676   { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2677   \usecounter { enumX#1 }
2678   \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2679   \str_if_eq:nnTF {#1} { v }
2680   {
2681     \__enumext_keyans_redefine_item:
2682     \__enumext_keyans_make_label:
2683     \__enumext_keyans_ref:
2684     \__enumext_keyans_fake_item:
2685     \bool_if:cT { \__enumext_show_length_#1_bool }
2686     {
2687       \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2688     }
2689   }
2690   {
2691     \__enumext_redefine_item:
2692     \__enumext_make_label:
2693     \__enumext_standar_ref:
2694     \__enumext_fake_item:
2695     \bool_if:cT { \__enumext_show_length_#1_bool }
2696     {
2697       \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_#1_int }
2698     }
2699   }
2700 }
2701 \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: For the horizontal environments enumext* and keyans* the implementation is similar, but, the value of
\__enumext_list_arg_two_viii: \partopsep is always 0pt. At this point we will modify the parsep key to make it take the value of the
                                itemsep key and later, in the environment definition, we will modify parindent to make it set the value
                                of listparindent and parsep to set the value of \parskip locally.

2702 \cs_set_protected:Npn \__enumext_tmp:n #1
2703 {
2704   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
2705   {
2706     \__enumext_calc_hspace:ccccc
2707     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2708     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2709     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2710     { \__enumext_leftmargin_tmp_#1_bool }
2711     \clist_map_inline:nn
2712       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2713       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2714     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2715       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2716     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2717     \skip_zero:N \partopsep
2718     \usecounter { enumX#1 }
2719     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2720     \__enumext_starred_ref:
2721     \str_if_eq:nnTF {#1} { vii }
2722       {
2723         \__enumext_fake_item_vii:
2724         \bool_if:cT { \__enumext_show_length_vii_bool }
2725           { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2726       }
2727       {
2728         \__enumext_fake_item_viii:
2729         \bool_if:cT { \__enumext_show_length_#1_bool }
2730           { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2731       }
2732   }
2733 }
2734 \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.

```

2735 \NewDocumentEnvironment{enumext}{0}{}
2736 {
2737   \__enumext_safe_exec:
2738   \__enumext_parse_keys:n {#1}
2739   \__enumext_before_list:
2740   \__enumext_start_store_level:
2741   \__enumext_start_list:nn
2742     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2743     {
2744       \use:c { \__enumext_list_arg_two_ \__enumext_level: : }
2745       \__enumext_before_keys_exec:
2746     }
2747   \__enumext_after_args_exec:
2748 }
2749 {
2750   \__enumext_stop_list:
2751   \__enumext_stop_store_level:
2752   \__enumext_after_list:
2753 }

```

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

`\__enumext_safe_exec:` The `\__enumext_safe_exec:` function first execute the function `\__enumext_is_not_nested:` which will set the variable `\g__enumext_standard_bool` to “true” if the environment is not nested in `enumext*`,

we increment the variable `\l__enumext_level_int` for the nesting levels and set the `\l__enumext_standard_bool` variable to “true”. Finally we set the variable `\l__enumext_standar_first_level_bool` to “true” only if the environment is not nested and we are at the “first level” of it using the function `\__enumext_is_on_first_level:`

```

2754 \cs_new_protected:Nn \__enumext_safe_exec:
2755 {
2756   \__enumext_is_not_nested:
2757   \int_incr:N \l__enumext_level_int
2758   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2759     { \msg_fatal:nn { enumext } { list-too-deep } }
2760   \bool_set_true:N \l__enumext_standar_bool
2761   \__enumext_is_on_first_level:
2762 }

```

(End of definition for `\__enumext_safe_exec:`)

`\__enumext_parse_keys:n`

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

```

2763 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2764 {
2765   \tl_if_novalue:nF {#1}
2766   {
2767     \str_clear:N \l__enumext_series_str
2768     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2769     {
2770       \keys_set:nn { enumext / level-1 } {#1}
2771       \__enumext_parse_series:n {#1}
2772     }
2773     {
2774       \exp_args:Ne \keys_set:nn
2775         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2776     }
2777     \bool_if:NT \l__enumext_store_active_bool
2778     {
2779       \__enumext_parse_store_keys:n {#1}
2780     }
2781   }
2782 }

```

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

`\__enumext_parse_store_keys:n`

The function `\__enumext_parse_store_keys:n` searches for the values of the `columns` and `columns-sep` keys in the optional arguments per-level in `enumext` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\l__enumext_store_opt_X_tl` which is used by the function `\__enumext_store_level_open:`.

```

2783 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2784 {
2785   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2786   {
2787     \regex_match:nnT { \b columns\b } {#1}
2788     {
2789       \int_set_eq:cc
2790         { \l__enumext_store_columns_ \__enumext_level: _int }
2791         { \l__enumext_columns_ \__enumext_level: _int }
2792       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2793         {
2794           columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2795         }
2796     }
2797   }
2798   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2799   {
2800     \regex_match:nnT { \b columns-sep\b } {#1}
2801     {
2802       \dim_set_eq:cc
2803         { \l__enumext_store_columns_sep_ \__enumext_level: _dim }

```

```

2804         { l__enumext_columns_sep_ \__enumext_level: _dim }
2805     \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2806     {
2807         columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2808     }
2809 }
2810 }
2811 }

```

(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.

```

2812 \cs_new_protected:Nn \__enumext_start_store_level:
2813 {
2814     \bool_lazy_all:nT
2815     {
2816         { \bool_if_p:N \__enumext_store_active_bool }
2817         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2818         { \bool_not_p:n { \g__enumext_starred_bool } }
2819     }
2820     {
2821         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2822         {
2823             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2824             \__enumext_store_level_open:
2825         }
2826     }
2827     \bool_lazy_all:nT
2828     {
2829         { \bool_if_p:N \__enumext_store_active_bool }
2830         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2831         { \bool_if_p:N \g__enumext_starred_bool }
2832     }
2833     {
2834         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2835         {
2836             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2837             \__enumext_store_level_open:
2838         }
2839     }
2840 }
2841 \cs_new_protected:Nn \__enumext_stop_store_level:
2842 {
2843     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2844     {
2845         \__enumext_store_level_close:
2846     }
2847 }

```

(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.

```

2848 \cs_new_protected:Nn \__enumext_before_list:
2849 {
2850     \__enumext_vspace_above:
2851     \__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.

```

2852     \__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.

```

2853     \dim_compare:nNnT

```



```

2854 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2855 {
2856   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2857   {
2858     \linewidth
2859     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2860     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2861   }

```

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

```

2862   \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2863   \int_gincr:N \g__enumext_minipage_stat_int
2864   \__enumext_mini_addvspace:
2865   \nointerlineskip\noindent
2866   \begin{\__enumext_mini_env*}
2867   { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2868   }
2869   \__enumext_multicols_start:
2870   }

```

(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=0pt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

2871 \cs_new_protected:Nn \__enumext_multicols_start:
2872 {
2873   \int_compare:nNt
2874   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2875   {
2876     \dim_compare:nNt
2877     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2878     {
2879       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2880       {
2881         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2882         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2883         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2884         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2885       }
2886     }
2887     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2888     \skip_zero:N \multicolsep
2889     \int_compare:nNt { \l__enumext_level_int } > { 1 }
2890     {
2891       \dim_zero:N \columnseprule
2892     }

```

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.

```

2893   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2894   {
2895     \__enumext_multi_addvspace:
2896   }
2897   \raggedcolumns
2898   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2899   }
2900   }

```

(End of definition for `\__enumext_multicols_start:`)

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

```

2901 \cs_new_protected:Nn \__enumext_multicols_stop:
2902 {
2903   \int_compare:nNt
2904     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2905   {
2906     \end{multicols}
2907     \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2908     {
2909       \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \__enumext_level: _skip } }
2910     }
2911   }
2912 }

```

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

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

```

2913 \cs_new_protected:Nn \__enumext_after_list:
2914 {
2915   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2916   {
2917     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2918     {
2919       \msg_warning:nn { enumext } { missing-miniright }
2920       \miniright
2921     }
2922     \int_gzero:N \g__enumext_minipage_stat_int
2923     \end{__enumext_mini_env*}
2924     \par\addvspace { \l__enumext_minipage_after_skip }
2925   }
2926   { \__enumext_multicols_stop: }

```

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

```

2927   %\bool_lazy_and:nnT
2928   % { \bool_if_p:N \l__enumext_check_ans_bool }
2929   % { \bool_if_p:N \g__enumext_standar_bool }
2930   % {
2931   %   \bool_gset_true:N \g__enumext_check_ans_show_bool
2932   %   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2933   % }
2934   \__enumext_check_ans_to_hook:

```

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

```

2935   \__enumext_after_stop_list:
2936   \__enumext_vspace_below:
2937   \bool_set_false:N \l__enumext_standar_bool
2938   \__enumext_resume_save_counter:
2939 }

```

(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`.

```

2940 \__enumext_after_env:nn {enumext}
2941 {
2942   \int_compare:nNt { \l__enumext_level_int } = { 0 }
2943   {
2944     \bool_if:NT \g__enumext_check_ans_show_bool
2945     {
2946       \__enumext_check_ans_show:
2947     }
2948     \bool_gset_false:N \g__enumext_standar_bool
2949     \bool_gset_false:N \g__enumext_check_ans_show_bool

```

```

2950         \tl_gclear:N \g__enumext_store_name_tl
2951     }
2952 }

```

### 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.

```

2953 \NewDocumentEnvironment{keyans}{0}{ }
2954 {
2955     \__enumext_keyans_safe_exec:
2956     \__enumext_keyans_parse_keys:n {#1}
2957     \__enumext_before_list_v:
2958     \__enumext_start_list:nn
2959     { \tl_use:N \l__enumext_label_v_tl }
2960     {
2961         \__enumext_list_arg_two_v:
2962         \__enumext_before_keys_exec_v:
2963     }
2964     \__enumext_after_args_exec_v:
2965 }
2966 {
2967     \__enumext_check_starred_cmd:n { item }
2968     \__enumext_stop_list:
2969     \__enumext_after_list_v:
2970 }

```

(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.

```

2971 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2972 {
2973     \bool_if:NF \l__enumext_store_active_bool
2974     {
2975         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
2976     }
2977     \int_incr:N \l__enumext_keyans_level_int
2978     \bool_set_true:N \l__enumext_keyans_env_bool
2979     \__enumext_keyans_save_start_line:
2980     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2981     \bool_set_false:N \l__enumext_store_active_bool
2982     \int_compare:nNtT { \l__enumext_keyans_level_int } > { 1 }
2983     {
2984         \msg_error:nn { enumext } { keyans-nested }
2985     }
2986     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2987     {
2988         \msg_error:nn { enumext } { keyans-wrong-level }
2989     }
2990 }

```

(End of definition for `\__enumext_keyans_safe_exec:`)

`\__enumext_keyans_parse_keys:n`

Parse [`key = val`] for `keyans` environment.

```

2991 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2992 {
2993     \keys_set:nn { enumext / keyans } {#1}
2994 }

```

(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.

```

2995 \cs_new_protected:Nn \__enumext_before_list_v:
2996 {
2997     \__enumext_vspace_above_v:
2998     \__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.

```

2999   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3000   {
3001     \dim_set:Nn \l__enumext_minipage_left_v_dim
3002     {
3003       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3004     }

```

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*.

```

3005     \bool_set_true:N \l__enumext_minipage_active_v_bool
3006     \int_gincr:N \g__enumext_minipage_stat_int
3007     \__enumext_keyans_mini_addvspace:
3008     \nointerlineskip\noindent
3009     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3010   }

```

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

```

3011   \__enumext_keyans_multicols_start:
3012 }

```

(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.

```

3013 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3014 {
3015   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3016   {

```

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

```

3017     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3018     {
3019       \dim_set:Nn \l__enumext_columns_sep_v_dim
3020       {
3021         (
3022           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3023         ) / \l__enumext_columns_v_int
3024         - \l__enumext_listoffset_v_dim
3025       }
3026     }
3027     \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).

```

3028     \skip_zero:N \multicolsep
3029     \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.

```

3030     \bool_if:NF \l__enumext_minipage_active_v_bool
3031     {
3032       \__enumext_keyans_multi_addvspace:
3033     }
3034     \raggedcolumns
3035     \begin{multicols}{ \l__enumext_columns_v_int }
3036   }
3037 }

```

(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.

```

3038 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3039 {
3040   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3041   {
3042     \end{multicols}
3043     \bool_if:NF \l__enumext_minipage_active_v_bool
3044     {
3045       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3046     }
3047   }
3048 }

```

(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.

```

3049 \cs_new_protected:Nn \__enumext_after_list_v:
3050 {
3051   \bool_if:NTF \l__enumext_minipage_active_v_bool
3052   {
3053     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3054     {
3055       \msg_warning:nn { enumext } { missing-miniright }
3056       \miniright
3057     }
3058     \int_gzero:N \g__enumext_minipage_stat_int
3059     \end{__enumext_mini_env*}
3060     \par\addvspace{ \l__enumext_minipage_after_skip }
3061   }
3062   { \__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.

```

3063   \bool_set_false:N \l__enumext_keyans_env_bool
3064   \__enumext_after_stop_list_v:
3065   \__enumext_vspace_below_v:
3066 }

```

(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.

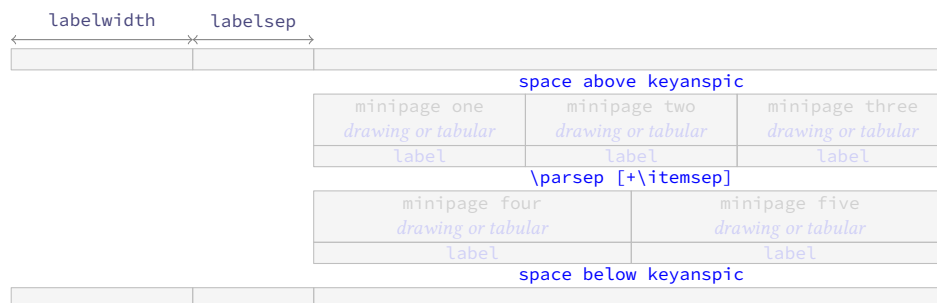


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

```
3067 \NewDocumentCommand \anspic { s o +m }
3068 {
```

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

```
3069 \bool_if:NF \__enumext_store_active_bool
3070 {
3071 \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3072 }
3073 \int_compare:nNnT { \__enumext_level_int } > { 1 }
3074 {
3075 \msg_error:nn { enumext } { keyanspic-wrong-level }
3076 }
3077 \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3078 {
3079 \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3080 }
```

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

```
3081 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3082 {
3083 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3084 }
3085 }
```

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

`\__enumext_keyans_anspic_code:nnn`

The function `\__enumext_keyans_anspic_code:nnn` will be in charge of handling the “counter” and  $\langle label \rangle$ , which will have the same configuration as the `keyans` environment.

```
3086 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3087 {
3088 \stepcounter { enumXvi }
3089 #3 \
3090 \bool_if:nT { #1 }
3091 {
3092 \__enumext_keyans_addto_prop:n { #2 }
3093 \__enumext_keyans_store_ref:
3094 \__enumext_keyans_addto_seq:n { #2 }
3095 \int_gincr:N \__enumext_check_starred_cmd_int
3096 \bool_lazy_or:nnT
3097 { \bool_if_p:N \__enumext_show_answer_bool }
3098 { \bool_if_p:N \__enumext_show_position_bool }
3099 {
3100 \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3101 \__enumext_keyans_show_left:n { #2 }
3102 \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3103 }
3104 }
3105 \tl_use:N \__enumext_label_font_style_v_tl
3106 \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3107 }
```

(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  $[\langle number above, number below \rangle]$  will determine the number of `minipage` environments that will be above and below separated by `\parsep+\itemsep` within it.

```
3108 \NewDocumentEnvironment{keyanspic}{ o }
3109 {
3110 \__enumext_keyans_pic_safe_exec:
3111 \__enumext_start_list:nn
3112 { }
3113 }
```

```

3114     \__enumext_keyans_pic_arg_two:
3115 }

```

We apply the “adjusted” vertical spacing above the environment

```

3116 \vspace { \l__enumext_keyans_pic_above_skip }
3117 }

```

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.

```

3118 {
3119   \tl_if_novalue:nTF { #1 }
3120   {
3121     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3122   }
3123   { \__enumext_keyans_pic_do:n { #1 } }
3124   \__enumext_stop_list:
3125   \__enumext_check_starred_cmd:n { anspic }
3126   \setcounter { enumXvi } { 0 }
3127   \vspace { \l__enumext_topsep_v_skip }
3128   %\bool_set_false:N \l__enumext_store_active_bool
3129 }

```

(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.

```

3130 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3131 {
3132   \int_incr:N \l__enumext_keyans_pic_level_int
3133   \int_compare:nNt { \l__enumext_keyans_pic_level_int } > { 1 }
3134   {
3135     \msg_error:nn { enumext } { keyanspic-nested }
3136   }
3137   \__enumext_keyans_save_start_line:
3138 }

```

(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`.

```

3139 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3140 {
3141   \dim_compare:nNt { #1 } < { 0pt }
3142   { \skip_set:Nn #1 { -#1 } }
3143 }

```

(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.

```

3144 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3145 {

```

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.

```

3146   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3147   \__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.

```

3148   \skip_add:Nn \parsep { \itemsep }
3149   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3150   \dim_zero:N \labelwidth
3151   \dim_zero:N \listparindent
3152   \dim_zero:N \labelsep
3153   \skip_zero:N \partopsep
3154   \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.

```

3155 \__enumext_keyans_pic_skip_abs:N \parsep
3156 \skip_set:Nn \l__enumext_keyans_pic_above_skip
3157 {
3158   \box_dp:N \strutbox
3159   + \l__enumext_topsep_v_skip
3160   - \parsep
3161 }
3162 \__enumext_item_std:w \scan_stop:
3163 }

```

(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`.

```

3164 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3165 {
3166   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3167 }
3168 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

`\__enumext_keyans_pic_row:n`

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

```

3169 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3170 {
3171   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3172   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3173   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3174   \int_step_inline:nnn
3175     { \l__enumext_keyans_pic_above_int + 1 }
3176     { \l__enumext_keyans_pic_below_int }
3177     {
3178       \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3179       \centering
3180       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3181       \__enumext_endminipage:
3182     }
3183   \par
3184 }

```

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

## 10.35 The environment `enumext*`

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

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

### 10.35.1 Functions for item box width

`\__enumext_starred_columns_set_vii:`

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3185 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3186 {
3187   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3188   {
3189     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3190     {
3191       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )

```

```

3192         / \l__enumext_columns_vii_int
3193     }
3194 }
3195 \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3196 \dim_set:Nn \l__enumext_item_width_vii_dim
3197 {
3198     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3199     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3200     - \l__enumext_labelsep_vii_dim
3201 }
3202 \dim_zero_new:N \itemwidth
3203 }

```

(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`.

```

3204 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3205 {
3206     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3207     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3208     {
3209         \msg_warning:nnee { enumext } { item-joined }
3210         { \int_use:N \l__enumext_joined_item_vii_int }
3211         { \int_use:N \l__enumext_columns_vii_int }
3212         \int_set:Nn \l__enumext_joined_item_vii_int
3213         {
3214             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3215         }
3216     }
3217     \int_compare:nNnT
3218     { \l__enumext_joined_item_vii_int }
3219     >
3220     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3221     {
3222         \msg_warning:nnee { enumext } { item-joined-columns }
3223         { \int_use:N \l__enumext_joined_item_vii_int }
3224         {
3225             \int_eval:n
3226             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3227         }
3228         \int_set:Nn \l__enumext_joined_item_vii_int
3229         {
3230             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3231         }
3232     }
3233 }

```

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

```

3233 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3234 {
3235     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3236     \int_decr:N \l__enumext_joined_item_aux_vii_int
3237     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3238     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3239     \dim_set:Nn \l__enumext_joined_width_vii_dim
3240     {
3241         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3242         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3243             + \l__enumext_columns_sep_vii_dim
3244             ) * \l__enumext_joined_item_aux_vii_int
3245     }
3246     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3247 }
3248 {
3249     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3250     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3251 }
3252 }

```

(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`.

```

3253 \cs_new_protected:Nn \__enumext_start_mini_vii:
3254 {
3255   \dim_compare:nNtT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3256   {
3257     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3258     {
3259       \linewidth
3260       - \l__enumext_minipage_right_vii_dim
3261       - \l__enumext_minipage_hsep_vii_dim
3262     }
3263     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3264     \dim_gset_eq:NN
3265       \g__enumext_minipage_right_vii_dim
3266       \l__enumext_minipage_right_vii_dim
3267     \__enumext_mini_addvspace_vii:
3268     \nointerlineskip\noindent
3269     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3270   }
3271 }

```

(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*”.

```

3272 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3273 {
3274   \bool_if:NT \l__enumext_minipage_active_vii_bool
3275   {
3276     \end{__enumext_mini_env*}
3277     \hfill
3278     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3279   }
3280 }

```

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*”.

```

3281 \__enumext_after_env:nn {enumext*}
3282 {
3283   \bool_if:NT \g__enumext_minipage_active_vii_bool
3284   {
3285     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3286     \par\addvspace { \g__enumext_minipage_right_skip }
3287     \bool_if:NF \g__enumext_minipage_center_vii_bool
3288     {
3289       \centering
3290     }
3291     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3292     \end{__enumext_mini_env*}
3293     \par\addvspace{ \g__enumext_minipage_after_skip }
3294   }
3295   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3296   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3297   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3298   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3299   \bool_gset_false:N \g__enumext_starred_bool
3300 }

```

(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.

```

3301 \NewDocumentEnvironment{enumext*}{ o }
3302 {

```

```

3303     \__enumext_safe_exec_vii:
3304     \__enumext_parse_keys_vii:n {#1}
3305     \__enumext_before_list_vii:
3306     \__enumext_start_store_level_vii:
3307     \__enumext_start_list:nn { }
3308     {
3309         \__enumext_list_arg_two_vii:
3310         \__enumext_before_keys_exec_vii:
3311     }
3312     \__enumext_starred_columns_set_vii:
3313     \item[] \scan_stop:
3314     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3315     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3316 }
3317 {
3318     \__enumext_stop_item_tmp_vii:
3319     \__enumext_remove_extra_parsep_vii:
3320     \__enumext_stop_list:
3321     \__enumext_stop_store_level_vii:
3322     \__enumext_after_list_vii:
3323 }

```

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

`\__enumext_safe_exec_vii:` First check the maximum nesting level for the `enumext*` environment then set the vars `\l__enumext_starred_bool` and `\g__enumext_starred_bool`.

```

3324 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3325 {
3326     \__enumext_is_not_nested:
3327     \int_incr:N \l__enumext_level_h_int
3328     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3329     {
3330         \msg_error:nn { enumext } { nested }
3331     }
3332     \bool_set_true:N \l__enumext_starred_bool
3333     \__enumext_is_on_first_level:
3334 }

```

(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.

```

3335 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3336 {
3337     \tl_if_novalue:nF {#1}
3338     {
3339         \str_clear:N \l__enumext_series_str
3340         \keys_set:nn { enumext / enumext* } {#1}
3341         \__enumext_parse_series:n {#1}
3342         \bool_if:NT \l__enumext_store_active_bool
3343         {
3344             \__enumext_parse_store_keys_vii:n {#1}
3345         }
3346     }
3347 }

```

(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:.`

```

3348 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3349 {
3350     \bool_if:NF \l__enumext_store_columns_vii_bool
3351     {
3352         \regex_match:nnT { \b columns\b } {#1}
3353         {

```

```

3354         \int_set_eq:NN
3355         \l__enumext_store_columns_vii_int
3356         \l__enumext_columns_vii_int
3357         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3358         {
3359             columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3360         }
3361     }
3362 }
3363 \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3364 {
3365     \regex_match:nnT { \b columns-sep \b} {#1}
3366     {
3367         \dim_set_eq:NN
3368         \l__enumext_store_columns_sep_vii_dim
3369         \l__enumext_columns_sep_vii_dim
3370         \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3371         {
3372             columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3373         }
3374     }
3375 }
3376 }

```

(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`.

```

3377 \cs_new_protected:Nn \__enumext_before_list_vii:
3378 {
3379     \__enumext_vspace_above_vii:
3380     \__enumext_check_ans_exec: % need by chek-ans
3381     \__enumext_before_args_exec_vii:
3382     \__enumext_start_mini_vii:
3383 }

```

(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.

```

3384 \cs_new_protected:Nn \__enumext_after_list_vii:
3385 {
3386     \__enumext_stop_mini_vii:
3387     \__enumext_after_stop_list_vii:
3388     \__enumext_check_ans_to_hook:
3389     \__enumext_vspace_below_vii:
3390     \bool_set_false:N \l__enumext_starred_bool
3391     \__enumext_resume_save_counter:
3392 }

```

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

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

`\__enumext_stop_store_level_vii:`

```

3393 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3394 {
3395     \bool_if:NT \l__enumext_store_active_bool
3396     {
3397         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3398         {
3399             \__enumext_store_level_open_vii:
3400         }
3401     }
3402 }

```

```

3403 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3404 {
3405     \bool_if:NT \l__enumext_store_active_bool
3406     {
3407         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3408         {
3409             \__enumext_store_level_close_vii:
3410         }
3411     }
3412 }

```

(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.

```

3413 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3414 {
3415     \__enumext_stop_item_tmp_vii:
3416     \int_incr:N \l__enumext_item_column_pos_vii_int
3417     \int_gincr:N \g__enumext_item_count_all_vii_int
3418     \__enumext_item_peek_args_vii:
3419 }

```

(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).

```

3420 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3421 {
3422     \peek_meaning:NTF (
3423     { \__enumext_joined_item_vii:w }
3424     { \__enumext_joined_item_vii:w (1) }
3425 }

```

(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.

```

3426 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3427 {
3428     \__enumext_starred_joined_item_vii:n {#1}
3429     \peek_meaning_remove:NTF *
3430     { \__enumext_starred_item_vii:w }
3431     { \__enumext_standard_item_vii:w }
3432 }

```

(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 ].

```

3433 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3434 {
3435     \bool_set_false:N \l__enumext_item_starred_vii_bool
3436     \peek_meaning:NTF [
3437     {
3438         \bool_set_eq:NN

```

```

3439         \l__enumext_wrap_label_vii_bool
3440         \l__enumext_wrap_label_opt_vii_bool
3441     \__enumext_start_item_vii:w
3442 }
3443 {
3444     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3445     \legacy_if_set_true:n { @noitemarg }
3446     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3447 }
3448 }

```

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

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

```

\__enumext_starred_item_vii:w
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii:w
3449 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3450 {
3451     \bool_set_true:N \l__enumext_item_starred_vii_bool
3452     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3453     \peek_meaning:NTF [
3454         { \__enumext_starred_item_vii_aux_i:w }
3455         { \__enumext_starred_item_vii_aux_ii:w }
3456     ]
3457     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3458     {
3459         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3460         \__enumext_starred_item_vii_aux_ii:w
3461     }
3462     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3463     {
3464         \peek_meaning:NTF [
3465             { \__enumext_starred_item_vii_aux_iii:w }
3466             {
3467                 \dim_set_eq:NN
3468                 \l__enumext_item_symbol_sep_vii_dim
3469                 \l__enumext_labelsep_vii_dim
3470                 \legacy_if_set_true:n { @noitemarg }
3471                 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3472             }
3473         ]
3474         \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3475         {
3476             \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3477             \legacy_if_set_true:n { @noitemarg }
3478             \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3479         }

```

(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.

```

3480 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3481 {
3482     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3483     \legacy_if:nT { @noitemarg }
3484     {
3485         \legacy_if_set_false:n { @noitemarg }
3486         \legacy_if:nT { @nmbrrlist }
3487         {
3488             \bool_if:NT \l__enumext_hyperref_bool
3489             {
3490                 \legacy_if_set_true:n { @hyper@item }
3491             }
3492             \refstepcounter{enumXvii}

```

```

3493         \bool_if:NT \l__enumext_check_ans_bool
3494         {
3495             \int_gincr:N \g__enumext_count_item_number_int
3496         }
3497     }
3498 }

```

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

```

3499 \group_begin:
3500 \lrbox{ \l__enumext_item_text_vii_box }
3501 \bool_if:NF \l__enumext_footnotes_key_bool
3502 {
3503     \__enumext_renew_footnote:
3504 }
3505 \bool_if:NT \l__enumext_item_starred_vii_bool
3506 {
3507     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3508     {
3509         \tl_gset_eq:NN
3510         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3511     }
3512     \mode_leave_vertical:
3513     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3514     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3515     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3516     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3517 }
3518 \group_begin:
3519 \tl_use:N \l__enumext_label_font_style_vii_tl
3520 \bool_if:NTF \l__enumext_wrap_label_vii_bool
3521 {
3522     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3523     { \__enumext_wrapper_label_vii:n {#1} }
3524 }
3525 {
3526     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3527 }
3528 \group_end:
3529 \skip_horizontal:N \l__enumext_labelsep_vii_dim
3530 \tl_use:N \l__enumext_after_list_args_vii_tl
3531 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3532 \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3533 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3534 \tl_use:N \l__enumext_fake_item_indent_vii_tl
3535 }

```

(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 `\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.

```

3536 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3537 {
3538     \__enumext_endminipage:
3539     \endlrbox
3540     \group_end:
3541     \box_set_wd:Nn \l__enumext_item_text_vii_box
3542     {
3543         \l__enumext_joined_width_vii_dim
3544         + \l__enumext_labelwidth_vii_dim
3545         + \l__enumext_labelsep_vii_dim
3546     }
3547     \int_set:Nn \hbadness { 10000 }
3548     \box_use:N \l__enumext_item_text_vii_box
3549     \bool_if:NF \l__enumext_footnotes_key_bool

```



```

3550     {
3551         \__enumext_print_footnote:
3552     }
3553     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3554     {
3555         \par\noindent
3556         \int_zero:N \l__enumext_item_column_pos_vii_int
3557     }
3558     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3559 }

```

(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.

```

3560 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3561 {
3562     \int_compare:nNnT
3563     {
3564         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3565     }
3566     =
3567     { \c_zero_int }
3568     {
3569         \par
3570         \vspace{ -\l__enumext_itemsep_vii_skip }
3571         \int_gzero:N \g__enumext_item_count_all_vii_int
3572     }
3573 }

```

(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`.

```

3574 \__enumext_after_env:nn {enumext*}
3575 {
3576     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3577     {
3578         \bool_if:NT \g__enumext_check_ans_show_h_bool
3579         {
3580             \__enumext_check_ans_show:
3581         }
3582         \bool_gset_false:N \g__enumext_starred_bool
3583         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3584         \tl_gclear:N \g__enumext_store_name_tl
3585     }
3586 }

```

## 10.36 The environment `keyans*`

### 10.36.1 Functions for item box width

\\_\_enumext\_starred\_columns\_set\_viii:

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3587 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3588 {
3589     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3590     {
3591         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3592         {
3593             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3594             / \l__enumext_columns_viii_int
3595         }
3596     }
3597     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3598     \dim_set:Nn \l__enumext_item_width_viii_dim
3599     {
3600         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3601         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3602         - \l__enumext_labelsep_viii_dim
3603     }
3604     \dim_zero_new:N \itemwidth
3605 }

```

(End of definition for `\__enumext_starred_columns_set_viii:`)

`\__enumext_starred_joined_item_viii:n`

The function `\__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the content passed to `\item<⟨number⟩⟩` will be stored together with the value of `\itemwidth`.

```

3606 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3607 {
3608   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3609   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3610   {
3611     \msg_warning:nnee { enumext } { item-joined }
3612     { \int_use:N \l__enumext_joined_item_viii_int }
3613     { \int_use:N \l__enumext_columns_viii_int }
3614     \int_set:Nn \l__enumext_joined_item_viii_int
3615     {
3616       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3617     }
3618   }
3619   \int_compare:nNnT
3620   { \l__enumext_joined_item_viii_int }
3621   >
3622   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3623   {
3624     \msg_warning:nnee { enumext } { item-joined-columns }
3625     { \int_use:N \l__enumext_joined_item_viii_int }
3626     {
3627       \int_eval:n
3628       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3629     }
3630     \int_set:Nn \l__enumext_joined_item_viii_int
3631     {
3632       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3633     }
3634   }
3635   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3636   {
3637     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3638     \int_decr:N \l__enumext_joined_item_aux_viii_int
3639     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3640     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3641     \dim_set:Nn \l__enumext_joined_width_viii_dim
3642     {
3643       \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3644       + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3645         + \l__enumext_columns_sep_viii_dim
3646       ) * \l__enumext_joined_item_aux_viii_int
3647     }
3648     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3649   }
3650   {
3651     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3652     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3653   }
3654 }

```

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

`\__enumext_start_mini_viii:`

The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.

`\__enumext_stop_mini_viii:`

```

3655 \cs_new_protected:Nn \__enumext_start_mini_viii:
3656 {
3657   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3658   {
3659     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3660     {
3661       \linewidth
3662       - \l__enumext_minipage_right_viii_dim
3663       - \l__enumext_minipage_hsep_viii_dim
3664     }
3665     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3666     \dim_gset_eq:NN
3667     \g__enumext_minipage_right_viii_dim
3668     \l__enumext_minipage_right_viii_dim

```

```

3669     \__enumext_mini_addvspace_viii:
3670     \nointerlineskip\noindent
3671     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3672   }
3673 }
3674 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3675 {
3676   \bool_if:NT \l__enumext_minipage_active_viii_bool
3677   {
3678     \end{__enumext_mini_env*}
3679     \hfill
3680     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3681   }
3682 }
3683 \__enumext_after_env:nn {keyans*}
3684 {
3685   \bool_if:NT \g__enumext_minipage_active_viii_bool
3686   {
3687     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3688     \par\addvspace { \g__enumext_minipage_right_skip }
3689     \bool_if:NF \g__enumext_minipage_center_viii_bool
3690     {
3691       \centering
3692     }
3693     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3694     \end{__enumext_mini_env*}
3695     \par\addvspace{ \g__enumext_minipage_after_skip }
3696   }
3697   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3698   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3699   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3700   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3701 }

```

(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.

```

3702 \NewDocumentEnvironment{keyans*}{ o }
3703 {
3704   \__enumext_safe_exec_viii:
3705   \__enumext_parse_keys_viii:n {#1}
3706   \__enumext_before_list_viii:
3707   \__enumext_start_list:nn { }
3708   {
3709     \__enumext_list_arg_two_viii:
3710     \__enumext_before_keys_exec_viii:
3711   }
3712   \__enumext_starred_columns_set_viii:
3713   \item[] \scan_stop:
3714   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3715   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3716 }
3717 {
3718   \__enumext_stop_item_tmp_viii:
3719   \__enumext_remove_extra_parsep_viii:
3720   \__enumext_check_starred_cmd:n { \item }
3721   \__enumext_stop_list:
3722   \__enumext_after_list_viii:
3723 }

```

(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.

```

3724 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3725 {
3726   \int_incr:N \l__enumext_keyans_level_h_int
3727   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3728   {
3729     \msg_error:nn { enumext } { nested }

```

```

3730     }
3731     \__enumext_keyans_save_start_line:
3732     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3733     \bool_set_false:N \__enumext_store_active_bool
3734     \int_compare:nNtT { \__enumext_level_int } > { 1 }
3735     {
3736         \msg_error:nn { enumext } { keyans-wrong-level }
3737     }
3738 }

```

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

\\_\_enumext\_parse\_keys\_viii:n Parse [*key = val*] for *keyans\**.

```

3739 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3740 {
3741     \tl_if_novalue:nF {#1}
3742     {
3743         \keys_set:nn { enumext / keyans* } {#1}
3744     }
3745 }

```

(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*.

```

3746 \cs_new_protected:Nn \__enumext_before_list_viii:
3747 {
3748     \__enumext_vspace_above_viii:
3749     \__enumext_before_args_exec_viii:
3750     \__enumext_start_mini_viii:
3751 }

```

(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.

```

3752 \cs_new_protected:Nn \__enumext_after_list_viii:
3753 {
3754     \__enumext_stop_mini_viii:
3755     \__enumext_after_stop_list_viii:
3756     \__enumext_vspace_below_viii:
3757 }

```

(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 \\_\_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*.

```

3758 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3759 {
3760     \__enumext_stop_item_tmp_viii:
3761     \int_incr:N \__enumext_item_column_pos_viii_int
3762     \int_gincr:N \g__enumext_item_count_all_viii_int
3763     \__enumext_item_peek_args_viii:
3764 }

```

(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).

```

3765 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3766 {
3767   \peek_meaning:NTF (
3768     { \__enumext_joined_item_viii:w }
3769     { \__enumext_joined_item_viii:w (1) }
3770   }

```

(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`.

```

3771 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3772 {
3773   \__enumext_starred_joined_item_viii:n {#1}
3774   \peek_meaning_remove:NTF *
3775   { \__enumext_starred_item_viii:w }
3776   { \__enumext_standard_item_viii:w }
3777 }

```

(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 ]`.

```

3778 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3779 {
3780   \bool_set_false:N \l__enumext_item_starred_viii_bool
3781   \peek_meaning:NTF [
3782     {
3783       \bool_set_eq:NN
3784         \l__enumext_wrap_label_viii_bool
3785         \l__enumext_wrap_label_opt_viii_bool
3786       \__enumext_start_item_viii:w
3787     }
3788     {
3789       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3790       \legacy_if_set_true:n { @noitemarg }
3791       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3792     }
3793   }

```

(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
3794 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3795 {
3796   \bool_set_true:N \l__enumext_item_starred_viii_bool
3797   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3798   \peek_meaning:NTF [
3799     { \__enumext_starred_item_viii_aux_i:w }
3800     { \__enumext_starred_item_viii_aux_ii:w }
3801   }

```

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

```

3802 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3803 {

```

```

3804 \tl_clear:N \l__enumext_store_keyans_label_tl
3805 \tl_if_novalue:nF { #1 }
3806 {
3807     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3808     {
3809         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3810         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3811     }
3812     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3813 }
3814 \__enumext_starred_item_viii_aux_ii:w
3815 }
3816 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3817 {
3818     \legacy_if_set_true:n { @noitemarg }
3819     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3820 }

```

(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.

```

3821 \cs_new_protected:Nn \__enumext_starred_item_exec:
3822 {
3823     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3824     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3825     \__enumext_keyans_store_ref:
3826     \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3827     \__enumext_keyans_addto_seq_link:
3828     \int_gincr:N \g__enumext_check_starred_cmd_int
3829     \bool_if:NT \l__enumext_show_answer_bool
3830     {
3831         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3832     }
3833     \bool_if:NT \l__enumext_show_position_bool
3834     {
3835         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3836         {
3837             \group_begin:
3838             \exp_not:N \normalfont
3839             \exp_not:N \footnotesize [ \int_eval:n
3840                 {
3841                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3842                 }
3843             ]
3844             \group_end:
3845         }
3846         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3847     }
3848 }

```

(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.

```

3849 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3850 {
3851     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3852     \legacy_if:nT { @noitemarg }
3853     {
3854         \legacy_if_set_false:n { @noitemarg }
3855         \legacy_if:nT { @nmbrrlist }
3856         {
3857             \bool_if:NT \l__enumext_hyperref_bool
3858             {
3859                 \legacy_if_set_true:n { @hyper@item }
3860             }
3861             \refstepcounter{enumXviii}

```

```

3862     }
3863 }

```

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

```

3864 \group_begin:
3865   \lrbox{ \l__enumext_item_text_viii_box }
3866   \bool_if:NF \l__enumext_footnotes_key_bool
3867   {
3868     \__enumext_renew_footnote:
3869   }
3870   \bool_if:NT \l__enumext_item_starred_viii_bool
3871   {
3872     \__enumext_starred_item_exec:
3873   }
3874   \group_begin:
3875     \tl_use:N \l__enumext_label_font_style_viii_tl
3876     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3877     {
3878       \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3879         { \__enumext_wrapper_label_viii:n {#1} }
3880     }
3881     {
3882       \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3883     }
3884   \group_end:
3885   \skip_horizontal:N \l__enumext_labelsep_viii_dim
3886   \tl_use:N \l__enumext_after_list_args_viii_tl
3887   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3888     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3889     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3890     \bool_if:NT \l__enumext_item_starred_viii_bool
3891     {
3892       \tl_use:N \l__enumext_fake_item_indent_viii_tl
3893       \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3894     }
3895     {
3896       \tl_use:N \l__enumext_fake_item_indent_viii_tl
3897     }
3898   }

```

(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.

```

3899 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3900 {
3901   \__enumext_endminipage:
3902   \endlrbox
3903   \group_end:
3904   \box_set_wd:Nn \l__enumext_item_text_viii_box
3905   {
3906     \l__enumext_joined_width_viii_dim
3907     + \l__enumext_labelwidth_viii_dim
3908     + \l__enumext_labelsep_viii_dim
3909   }
3910   \int_set:Nn \hbadness { 10000 }
3911   \box_use:N \l__enumext_item_text_viii_box
3912   \bool_if:NF \l__enumext_footnotes_key_bool
3913   {
3914     \__enumext_print_footnote:
3915   }
3916   \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3917   {
3918     \par\noindent
3919     \int_zero:N \l__enumext_item_column_pos_viii_int
3920   }
3921   { \hspace{ \l__enumext_columns_sep_viii_dim } }
3922 }

```

(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.

```

3923 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3924 {
3925   \int_compare:nNtT
3926   {
3927     \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3928   }
3929   =
3930   { \c_zero_int }
3931   {
3932     \par
3933     \vspace{ -\l__enumext_itemsep_viii_skip }
3934     \int_gzero:N \g__enumext_item_count_all_viii_int
3935   }
3936 }

```

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

### 10.37 The command `\getkeyans`

`\getkeyans`

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

```

3937 \NewDocumentCommand \getkeyans { m }
3938 {
3939   \exp_args:Ne \__enumext_getkeyans_aux:n
3940   { \tl_to_str:e { \text_expand:n {#1} } }
3941 }

```

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

`\__enumext_getkeyans_aux:n`

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

```

3942 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3943 {
3944   \str_if_in:nnTF {#1} { : }
3945   {
3946     \use:e
3947     {
3948       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3949       { {##1} {##2} }
3950     }
3951     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3952   }
3953   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3954 }

```

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

`\__enumext_getkeyans:nn`

The internal function `\__enumext_getkeyans:nn` will check for the existence of the `⟨prop list⟩`, if it does not exist it will return an error message, then it will fetch the content specified by the second `⟨argument⟩` from `⟨prop list⟩`.

```

3955 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3956 {
3957   \prop_if_exist:cF { g__enumext_#1_prop }
3958   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3959   \group_begin:
3960     \prop_item:cn { g__enumext_#1_prop }{#2}
3961   \group_end:
3962 }

```

(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`.

```

3963 \keys_define:nn { keyanskey / print }
3964 {
3965   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3966                 {
3967                   \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3968                 },
3969   level-1 .initial:n = { label=\arabic*, nosep, columns=2, first=\small, font=\small },
3970   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3971                 {
3972                   \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3973                 },
3974   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3975   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3976                 {
3977                   \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3978                 },
3979   level-3 .initial:n = { nosep, label=\roman*, first=\small, font=\small },
3980   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3981                 {
3982                   \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3983                 },
3984   level-4 .initial:n = { nosep, label=\Alph*, first=\small, font=\small },
3985   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3986                 {
3987                   \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3988                 },
3989   level-* .initial:n = { label=\arabic*, nosep, columns=2, first=\small, font=\small },
3990 }

```

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

```

3991 \NewDocumentCommand \printkeyans { s O{} m }
3992 {
3993   \group_begin:
3994     \tl_use:N \l__enumext_print_keyans_i_tl
3995     \tl_use:N \l__enumext_print_keyans_ii_tl
3996     \tl_use:N \l__enumext_print_keyans_iii_tl
3997     \tl_use:N \l__enumext_print_keyans_iv_tl
3998     \tl_use:N \l__enumext_print_keyans_vii_tl
3999     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4000   \group_end:
4001 }

```

(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

```

4002 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4003 {
4004   \seq_if_exist:cTF { g__enumext_#3_seq }
4005   {
4006     \seq_if_empty:cF { g__enumext_#3_seq }
4007     {
4008       %%\seq_show:c { g__enumext_#3_seq }
4009       \bool_if:nTF {#1}
4010       {
4011         \begin{enumext*}[#2]
4012         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4013         \end{enumext*}
4014       }
4015     }

```

```

4016         \begin{enumext}[#2]
4017         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4018         \end{enumext}
4019     }
4020 }
4021 }
4022 {
4023     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4024 }
4025 }

```

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

### 10.39 The command `\setenumext`

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

```

4026 \keys_define:nn { enumext / meta-families }
4027 {
4028     level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4029     level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4030     level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4031     level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4032     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4033     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4034     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4035     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
4036     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
4037     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
4038     print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
4039     print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
4040     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4041 }

```

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

```

4042 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4043 {
4044     level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
4045     keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
4046 }

```

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

```

4047 \NewDocumentCommand \setenumext { o +m }
4048 {
4049     \tl_if_novalue:nTF {#1}
4050     {
4051         \seq_map_inline:Nn \c__enumext_all_families_seq
4052     }
4053     {
4054         \seq_clear:N \l__enumext_setkey_tmpa_seq
4055         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4056         \int_set:Nn \l__enumext_setkey_tmpa_int
4057         {
4058             \seq_count:N \l__enumext_setkey_tmpb_seq
4059         }
4060         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4061         {
4062             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4063             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4064             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4065             {
4066                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4067             }
4068         }
4069         {
4070             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4071         }
4072         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4073         { \seq_map_inline:Nn \c__enumext_all_families_seq }
4074         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4075     }
4076     {
4077         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }

```

```

4078     }
4079 }

```

(End of definition for `\setenumext`. This function is documented on page 5.)

```

\__enumext_set_parse:n
\__enumext_set_error:nn

```

Internal functions used by the `\setenumext` command.

```

4080 \cs_new_protected:Npn \__enumext_set_parse:n #1
4081 {
4082   \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4083   \int_step_inline:nnn { 0 } { 4 } {%< max level
4084     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4085     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4086     {
4087       \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
4088         { \tl_trim_spaces:n {#1} }
4089     }
4090     { \__enumext_set_error:nn {#1} { } }
4091   }
4092   \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4093   { \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.

```

4094 \msg_new:nnn { enumext } { package-load }
4095 {
4096   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4097 }
4098 \msg_new:nnn { enumext } { package-not-load }
4099 {
4100   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4101 }
4102 \msg_new:nnn { enumext } { package-load-foot }
4103 {
4104   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4105 }

```

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

```

4106 \msg_new:nnn { enumext } { counters }
4107 {
4108   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
4109   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4110 }

```

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

```

4111 \msg_new:nnn { enumext } { invalid-key }
4112 {
4113   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4114 }
4115 \msg_new:nnn { enumext } { unknown-key-family }
4116 {
4117   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4118 }

```

Messages used in length calculation.

```

4119 \msg_new:nnn { enumext } { width-negative }
4120 {
4121   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4122   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4123 }
4124 \msg_new:nnn { enumext } { width-zero }
4125 {
4126   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4127   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4128 }

```

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

```

4129 \msg_new:nnn { enumext } { list-lengths }
4130 {
4131     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4132     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4133     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4134     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4135     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4136     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4137     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4138     \__enumext_show_length:nnn { skip } { topsep } {#1}
4139     \__enumext_show_length:nnn { skip } { parsep } {#1}
4140     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4141     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4142     *****
4143 }

```

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

```

4144 \msg_new:nnn { enumext } { list-lengths-not-nested }
4145 {
4146     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4147     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4148     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4149     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4150     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4151     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4152     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4153     \__enumext_show_length:nnn { skip } { topsep } {#1}
4154     \__enumext_show_length:nnn { skip } { parsep } {#1}
4155     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4156     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4157     *****
4158 }

```

Messages used by `ref` key.

```

4159 \msg_new:nnn { enumext } { key-ref-empty }
4160 {
4161     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4162 }

```

Messages used by `save-ans` key.

```

4163 \msg_new:nnn { enumext } { save-ans-empty }
4164 {
4165     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4166 }
4167 \msg_new:nnn { enumext } { save-ans-ok }
4168 {
4169     Set ~ 'save-ans=#2' ~ in ~ '#1' ~ \msg_line_context:.
4170 }

```

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

```

4171 \msg_new:nnn { enumext } { need-save-ans }
4172 {
4173     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4174 }
4175 \msg_new:nnn { enumext } { items-same-answer }
4176 {
4177     *****~Checking~answers~on~'#1'~OK~*****\\
4178     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4179     *****
4180     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4181 }
4182 \msg_new:nnn { enumext } { item-different-answer }
4183 {
4184     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4185     answer ~ stored ~ in ~ '#1'~ #2.
4186 }
4187 %\msg_new:nnn { enumext } { item-different-answer }
4188 % {
4189 %     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
4190 %     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4191 % }

```

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

```
4192 \msg_new:nnn { enumext } { missing-starred }
4193 {
4194     Missing ~ '\c_backslash_str #1*' ~ #2.
4195 }
4196 \msg_new:nnn { enumext } { many-starred }
4197 {
4198     Many ~ '\c_backslash_str #1*' ~ #2.
4199 }
```

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

```
4200 \msg_new:nnn { enumext } { list-too-deep }
4201 {
4202     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4203     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4204 }
```

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

```
4205 \msg_new:nnn { enumext } { anskey-wrong-place }
4206 {
4207     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4208     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4209 }
4210 \msg_new:nnn { enumext } { anspic-wrong-place }
4211 {
4212     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4213     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4214 }
4215 \msg_new:nnn { enumext } { command-wrong-place }
4216 {
4217     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4218     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4219 }
```

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

```
4220 \msg_new:nnn { enumext } { keyans-nested }
4221 {
4222     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4223 }
4224 \msg_new:nnn { enumext } { keyans-wrong-level }
4225 {
4226     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4227     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4228 }
4229 \msg_new:nnn { enumext } { wrong-place }
4230 {
4231     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4232     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4233 }
4234 \msg_new:nnn { enumext } { keyanspic-nested }
4235 {
4236     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4237 }
4238 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4239 {
4240     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4241     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4242 }
```

Messages used by `\getkeyans` command.

```
4243 \msg_new:nnn { enumext } { undefined-storage-anskey }
4244 {
4245     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4246 }
```

Messages used by `\miniright` command.

```
4247 \msg_new:nnn { enumext } { missing-miniright }
4248 {
4249     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4250     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4251 }
4252 \msg_new:nnn { enumext } { wrong-miniright-place }
4253 {
```

```

4254     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4255     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4256   }
4257   \msg_new:nnn { enumext } { wrong-miniright-use }
4258   {
4259     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4260     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4261   }

```

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

```

4262   \msg_new:nnn { enumext } { nested }
4263   {
4264     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4265   }
4266   \msg_new:nnn { enumext } { item-joined }
4267   {
4268     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4269   }
4270   \msg_new:nnn { enumext } { item-joined-columns }
4271   {
4272     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
4273   }

```

## 10.41 Finish package

Finish package implementation.

```

4274   \file_input_stop:
4275   </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> .....	203
<code>\+</code> .....	195
<code>\-</code> .....	195
<code>\\</code> .....	211, 3089, 4108, 4121, 4126, 4131, 4146, 4177, 4178, 4202, 4207, 4212, 4217, 4226, 4231, 4240, 4249, 4254, 4259
A	
<code>above</code> .....	<u>1339</u>
<code>above*</code> .....	<u>1339</u>
<code>\addvspace</code> ..	986, 1014, 1137, 1216, 1279, 1285, 1313, 1330, 2909, 2924, 3045, 3060, 3286, 3293, 3688, 3695
<code>after</code> .....	<u>824</u>
<code>align</code> .....	<u>441</u>
<code>\Alph</code> .....	32, 36, 37
<code>\Alph</code> .....	393, 502, 547, 615, 3984
<code>\alph</code> .....	32, 36, 37
<code>\alph</code> .....	394, 500, 3974
<code>\anskey</code> .....	11, 65, <u>2050</u>
<code>\anspic</code> .....	13, 86, 87, <u>3067</u>
<code>\anspic*</code> .....	61
<code>\arabic</code> .....	27, 32
<code>\arabic</code> .....	392, 499, 546, 3969, 3989
B	
<code>\b</code> .....	2787, 2800, 3352, 3365
<code>\baselineskip</code> .....	44
<code>\baselineskip</code> .....	2010, 2018
<code>before</code> .....	<u>824</u>
<code>before*</code> .....	<u>824</u>
<code>below</code> .....	<u>1339</u>
<code>below*</code> .....	<u>1339</u>
bool commands:	
<code>\bool_gset_false:N</code> ..	2948, 2949, 3295, 3299, 3582, 3583, 3697
<code>\bool_gset_true:N</code> ..	231, 241, 928, 1833, 1840, 2931, 3278, 3296, 3680, 3698
<code>\bool_if:NTF</code> ..	333, 345, 362, 1361, 1375, 1388, 1399, 1410, 1421, 1432, 1443, 1496, 1513, 1518, 1526, 1553, 1591, 1596, 1603, 1607, 1629, 1634, 1642, 1649, 1680, 1688, 1711, 1715, 1721, 1727, 1799, 1809, 1813, 1817, 1972, 1996, 2003, 2031, 2062, 2075, 2077, 2088, 2108, 2233, 2244, 2248, 2287, 2302, 2374, 2385, 2389, 2502, 2532, 2606, 2622, 2684, 2694, 2724, 2729, 2777, 2785, 2798, 2843, 2893, 2907, 2915, 2944, 2973, 3030, 3043, 3051, 3069, 3274, 3283, 3287, 3342, 3350, 3363, 3395, 3405, 3488, 3493, 3501, 3505, 3520, 3549, 3578, 3676, 3685, 3689, 3829, 3833, 3857, 3866, 3870, 3876, 3890, 3912
<code>\bool_if:nTF</code> ..	1314, 1331, 2116, 2543, 2578, 2642, 3090, 4009
<code>\bool_if_p:N</code> ..	251, 264, 1660, 1661, 1669, 1670, 1778, 1830, 1831, 1837, 1838, 2099, 2142, 2143, 2167, 2176, 2177, 2189, 2205, 2361, 2362, 2399, 2400, 2816, 2829, 2831, 2928, 2929, 3097, 3098
<code>\bool_lazy_all:nTF</code> ..	249, 262, 1776, 2165, 2174, 2187, 2203, 2814, 2827
<code>\bool_lazy_and:nnTF</code> ..	227, 237, 1659, 1668, 1829, 1836, 2098, 2141, 2360, 2927
<code>\bool_lazy_or:nnTF</code> ..	2398, 3096
<code>\bool_new:N</code> ..	25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88, 93, 94, 97, 117, 119, 121, 124, 125, 134, 135, 136, 137, 143, 144, 158, 169, 171
<code>\bool_not_p:n</code> ..	228, 238, 2100, 2192, 2207, 2817, 2818, 2830
<code>\bool_set_eq:NN</code> ..	2510, 2558, 3438, 3783
<code>\bool_set_false:N</code> ..	342, 1760, 1761, 2937, 2981, 3063, 3128, 3146, 3390, 3435, 3733, 3780
<code>\bool_set_true:N</code> ..	256, 269, 324, 328, 434, 752, 1345, 1350, 1616, 1737, 1738, 1934, 1941, 2506, 2536, 2554, 2566, 2760, 2823, 2836, 2862, 2978, 3005, 3263, 3332, 3444, 3451, 3452, 3665, 3789, 3796, 3797
box commands:	
<code>\box_dp:N</code> ..	1033, 1037, 1041, 1052, 1056, 1067, 1076, 1082, 1092, 1105, 1111, 1117, 1148, 1149, 1150, 1153, 1163, 1167, 1176, 1183, 1188, 1196, 1225, 1226, 1229, 1236, 1249, 1257, 1263, 1271, 3158
<code>\box_new:N</code> ..	58, 164
<code>\box_set_wd:Nn</code> ..	3541, 3904
<code>\box_use:N</code> ..	3548, 3911
<code>\box_wd:N</code> ..	400
C	
<code>\c</code> .....	203, 204, 652, 654, 666, 668
<code>\cB</code> .....	204
<code>\cE</code> .....	204
<code>\centering</code> .....	1316, 1333, 3179, 3289, 3691
<code>check-ans</code> .....	<u>1752</u>
Document class:	
<code>article</code> .....	38
clist commands:	
<code>\clist_const:Nn</code> .....	176
<code>\clist_map_function:nN</code> .....	3166
<code>\clist_map_inline:Nn</code> ..	440, 694, 757, 823, 838, 919, 1355
<code>\clist_map_inline:nn</code> ..	36, 47, 66, 72, 84, 96, 123, 152, 175, 219, 465, 482, 762, 934, 1461, 1705, 1766, 1911, 1929, 1950, 2162, 2296, 2460, 2671, 2674, 2701, 2711, 2714, 2734
<code>\columnbreak</code> .....	66
<code>\columnbreak</code> .....	2102
<code>columns</code> .....	<u>903</u>
<code>columns*</code> .....	<u>1930</u>
<code>columns-sep</code> .....	<u>903</u>
<code>columns-sep*</code> .....	<u>1930</u>
<code>\columnsep</code> .....	82, 85
<code>\columnsep</code> .....	2887, 3027
<code>\columnseprule</code> .....	82, 85
<code>\columnseprule</code> .....	2891, 3029
Commands provide by <b>enumext</b> :	
<code>\anskey</code> ..	24, 25, 58, 59, 63, 65, 67–69, 71, 81, 93, 104, 105, 109
<code>\anspic*</code> ..	24, 25, 61, 69, 70, 87–89, 104, 105
<code>\anspic</code> ..	63, 86–88, 109
<code>\getkeyans</code> ..	63, 104, 109
<code>\item*</code> ..	24, 25, 61, 63, 69, 70, 74, 75, 95, 101, 102, 104, 105
<code>\itemwidth</code> ..	89, 90, 97, 98
<code>\item</code> ..	73, 75, 90, 94, 95, 98, 100, 101
<code>\miniright</code> ..	24, 42, 49, 50, 82, 83, 85, 86, 109
<code>\printkeyans</code> ..	25, 63, 105

\setenumext . . . . . 25, 106, 107

Counters defined by **enumext**:

enumXiii . . . . . 23, 31

enumXii . . . . . 23, 31

enumXiv . . . . . 23, 31

enumXi . . . . . 23, 31

enumXviii . . . . . 23, 31

enumXvii . . . . . 23, 31, 95

enumXvi . . . . . 23, 31

enumXv . . . . . 23, 31

cs commands:

\cs\_generate\_variant:Nn 402, 418, 658, 674, 1955,  
1964, 1969, 2049, 2661, 3168

\cs\_if\_exist:NTF . . . . . 372

\cs\_new:Nn . . . . . 189

\cs\_new:Npn . . . . . 207, 1462, 1471, 1480

\cs\_new\_eq:NN 308, 309, 310, 314, 315, 347, 348, 351,  
352

\cs\_new\_protected:Nn . 199, 221, 247, 276, 319, 523,  
586, 638, 839, 843, 847, 851, 855, 859, 863, 867, 871,  
875, 879, 883, 887, 891, 895, 899, 935, 947, 971, 988,  
999, 1023, 1098, 1122, 1139, 1201, 1218, 1240, 1275,  
1281, 1356, 1370, 1384, 1395, 1406, 1417, 1428, 1439,  
1524, 1627, 1640, 1657, 1678, 1735, 1771, 1807, 1827,  
1844, 1970, 1994, 2001, 2029, 2036, 2153, 2285, 2300,  
2328, 2358, 2394, 2406, 2413, 2465, 2469, 2488, 2539,  
2574, 2590, 2600, 2616, 2754, 2812, 2841, 2848, 2871,  
2901, 2913, 2971, 2995, 3013, 3038, 3049, 3086, 3130,  
3144, 3164, 3169, 3185, 3253, 3272, 3324, 3377, 3384,  
3393, 3403, 3420, 3560, 3587, 3655, 3674, 3724, 3746,  
3752, 3765, 3821, 3923

\cs\_new\_protected:Npn 181, 185, 355, 370, 387, 397,  
403, 503, 548, 620, 645, 659, 1303, 1322, 1492, 1511,  
1581, 1614, 1706, 1859, 1956, 1965, 2085, 2230, 2242,  
2264, 2338, 2379, 2498, 2516, 2550, 2562, 2630, 2664,  
2704, 2763, 2783, 2991, 3139, 3204, 3335, 3348, 3426,  
3433, 3449, 3457, 3462, 3474, 3606, 3739, 3771, 3778,  
3794, 3802, 3816, 3942, 3955, 4002, 4080, 4092

\cs\_new\_protected\_nopar:Nn . . . 3413, 3536, 3758,  
3899

\cs\_new\_protected\_nopar:Npn . . . . . 3480, 3849

\cs\_set:Nn . . . . . 2235

\cs\_set:Npn . . . . . 2163, 2201, 3948

\cs\_set\_eq:NN . . . 3314, 3315, 3482, 3714, 3715, 3851

\cs\_set\_protected:Nn . . . . 213, 763, 779, 791, 803

\cs\_set\_protected:Npn . . 32, 41, 59, 67, 79, 85, 113,  
148, 156, 215, 419, 441, 470, 483, 530, 675, 695, 739,  
758, 815, 824, 903, 920, 1339, 1450, 1697, 1752, 1876,  
1912, 1930, 2155, 2289, 2449, 2662, 2702

\cs\_to\_str:N . . . . . 389, 412

## D

\d . . . . . 195

\DeclareDocumentEnvironment . . . . . 1016

dim commands:

\dim\_abs:n . . . . . 2635, 2640

\dim\_add:Nn . . . . . 3149

\dim\_compare:nNnTF . 765, 781, 793, 805, 1305, 1324,  
2632, 2637, 2643, 2649, 2651, 2653, 2853, 2876, 2999,  
3017, 3141, 3187, 3255, 3589, 3657

\dim\_compare:nTF . . . . . 2126

\dim\_gset\_eq:NN . . . . . 3264, 3666

\dim\_gzero:N . . . . . 3298, 3700

\dim\_new:N . 55, 62, 63, 64, 81, 107, 120, 130, 165, 166,  
172

\dim\_set:Nn . . . 400, 753, 1942, 2530, 2635, 2640, 2642,  
2645, 2646, 2650, 2652, 2655, 2656, 2658, 2856, 2879,  
3001, 3019, 3171, 3189, 3196, 3239, 3257, 3476, 3591,  
3598, 3641, 3659

\dim\_set\_eq:NN 490, 537, 608, 612, 2525, 2673, 2713,  
2802, 2887, 3027, 3246, 3249, 3250, 3367, 3467, 3648,  
3651, 3652

\dim\_use:N 766, 774, 1306, 1312, 2039, 2042, 2047, 2595,  
2597, 2854, 2859, 2860, 2867, 2877, 2881, 2882, 2884

\dim\_zero:N . . . . . 2891, 3029, 3150, 3151, 3152

\dim\_zero\_new:N . . . . . 3202, 3604

\c\_zero\_dim 768, 782, 794, 806, 1306, 1324, 2128, 2632,  
2637, 2643, 2650, 2854, 2877, 2999, 3017, 3187, 3255,  
3589, 3657

## E

\end . . . 1309, 1327, 1998, 2033, 2906, 2923, 3042, 3059, 3276,  
3292, 3678, 3694, 4013, 4018

\endlist . . . . . 29

\endlist . . . . . 309

\endlrbox . . . . . 3539, 3902

\endminipage . . . . . 29

\endminipage . . . . . 315

enumext . . . . . 5, 2735

enumext internal commands:

\l\_\_enumext\_u\_check\_start\_line\_env\_tl . . . 28

\l\_\_enumext\_u\_ref\_the\_count\_tl . . . . . 34

\g\_\_enumext\_ t\_\_enumext\_store\_name\_tl  
\_prop . . . . . 71

\l\_\_enumext\_\_resume\_name\_tl . . . . . 54, 55

\\_\_enumext\_add\_pre\_parsep: . . . 43, 945, 947, 947

\\_\_enumext\_after\_args\_exec: . . 40, 839, 851, 2747

\\_\_enumext\_after\_args\_exec\_v: 41, 855, 867, 2964

\\_\_enumext\_after\_args\_exec\_vii: . . . 871, 895

\\_\_enumext\_after\_args\_exec\_viii: . . . . . 899

\\_\_enumext\_after\_env:nn . . 83, 97, 185, 185, 2940,  
3281, 3574, 3683

\\_\_enumext\_after\_hyperref: . . . 30, 317, 319, 319

\\_\_enumext\_after\_list: . . 83, 93, 100, 2752, 2913,  
2913

\l\_\_enumext\_after\_list\_args\_v\_tl . . . . . 869

\l\_\_enumext\_after\_list\_args\_vii\_tl 897, 3530

\l\_\_enumext\_after\_list\_args\_viii\_tl 901, 3886

\\_\_enumext\_after\_list\_v: . . 86, 2969, 3049, 3049

\\_\_enumext\_after\_list\_vii: . . . 3322, 3384, 3384

\\_\_enumext\_after\_list\_viii: . . 3722, 3752, 3752

\\_\_enumext\_after\_star\_env:nn . . . . . 91

\\_\_enumext\_after\_stop\_list: . . 40, 41, 839, 847,  
2935

\\_\_enumext\_after\_stop\_list\_v: 41, 855, 863, 3064

\l\_\_enumext\_after\_stop\_list\_v\_tl . . . . . 865

\\_\_enumext\_after\_stop\_list\_vii: 871, 887, 3387

\l\_\_enumext\_after\_stop\_list\_vii\_tl . . . 889

\\_\_enumext\_after\_stop\_list\_viii: . 891, 3755

\l\_\_enumext\_after\_stop\_list\_viii\_tl . . . 893

\l\_\_enumext\_align\_label\_vii\_str . . 3522, 3526

\l\_\_enumext\_align\_label\_viii\_str . 3878, 3882

\l\_\_enumext\_align\_label\_X\_str . . . . . 156

\c\_\_enumext\_all\_envs\_clist . . 176, 440, 694, 757,  
823, 838, 919, 1355

\c\_\_enumext\_all\_families\_seq . . 106, 4042, 4051,  
4073

\\_\_enumext\_anskey\_wrapper:n . . . . . 1880, 2240

\\_\_enumext\_at\_begin\_document:n . . 29, 181, 181,  
306, 312



`\__enumext_before_args_exec:` [40](#), [839](#), [839](#), [2851](#)  
`\__enumext_before_args_exec_v:` [40](#), [41](#), [855](#), [855](#), [2998](#)  
`\__enumext_before_args_exec_vii:` [871](#), [871](#), [3381](#)  
`\__enumext_before_args_exec_viii:` [875](#), [3749](#)  
`\__enumext_before_keys_exec:` [40](#), [839](#), [843](#), [2745](#)  
`\__enumext_before_keys_exec_v:` [40](#), [855](#), [859](#), [2962](#)  
`\__enumext_before_keys_exec_vii:` [871](#)  
`\__enumext_before_keys_exec_vii:` [41](#), [879](#), [3310](#)  
`\__enumext_before_keys_exec_viii:` [41](#), [883](#), [3710](#)  
`\__enumext_before_list:` [81](#), [2739](#), [2848](#), [2848](#)  
`\__enumext_before_list_v:` [84](#), [2957](#), [2995](#), [2995](#)  
`\__enumext_before_list_vii:` [93](#), [3305](#), [3377](#), [3377](#)  
`\__enumext_before_list_viii:` [100](#), [3706](#), [3746](#), [3746](#)  
`\l__enumext_before_no_starred_key_v_tl` [861](#)  
`\l__enumext_before_no_starred_key_vii_tl` [881](#)  
`\l__enumext_before_no_starred_key_viii_tl` [885](#)  
`\l__enumext_before_starred_key_v_tl` [857](#)  
`\l__enumext_before_starred_key_vii_tl` [873](#)  
`\l__enumext_before_starred_key_viii_tl` [877](#)  
`\__enumext_calc_hspace:NNNNNN` [77](#), [2630](#), [2630](#), [2661](#), [2666](#), [2706](#)  
`\l__enumext_check_ans_bool` [73](#), [74](#), [134](#), [1756](#), [1761](#), [1809](#), [1830](#), [1837](#), [2077](#), [2374](#), [2502](#), [2532](#), [2928](#), [3493](#)  
`\__enumext_check_ans_exec:` [60](#), [81](#), [1807](#), [1807](#), [2852](#), [3380](#)  
`\__enumext_check_ans_gdecr:` [60](#), [1771](#), [1771](#), [1823](#)  
`\g__enumext_check_ans_item_tl` [71](#)  
`\__enumext_check_ans_show:` [61](#), [1844](#), [1844](#), [2946](#), [3580](#)  
`\g__enumext_check_ans_show_bool` [83](#), [134](#), [1833](#), [2931](#), [2944](#), [2949](#)  
`\g__enumext_check_ans_show_h_bool` [134](#), [1840](#), [3578](#), [3583](#)  
`\__enumext_check_ans_to_hook:` [61](#), [1827](#), [1827](#), [2934](#), [3388](#)  
`\__enumext_check_starred_cmd:n` [28](#), [61](#), [71](#), [1859](#), [1859](#), [2967](#), [3125](#), [3720](#)  
`\g__enumext_check_starred_cmd_int` [134](#), [1862](#), [1868](#), [1873](#), [2572](#), [3095](#), [3828](#)  
`\g__enumext_check_start_line_env_tl` [28](#), [134](#), [257](#), [270](#), [1853](#), [1857](#)  
`\l__enumext_check_start_line_env_tl` [134](#), [282](#), [290](#), [298](#), [1865](#), [1871](#), [1874](#)  
`\l__enumext_columns_sep_v_dim` [3017](#), [3019](#), [3027](#)  
`\l__enumext_columns_sep_vii_dim` [3187](#), [3189](#), [3198](#), [3243](#), [3369](#), [3558](#)  
`\l__enumext_columns_sep_viii_dim` [3589](#), [3591](#), [3600](#), [3645](#), [3921](#)  
`\l__enumext_columns_v_int` [1144](#), [3015](#), [3023](#), [3035](#), [3040](#)  
`\l__enumext_columns_vii_int` [3192](#), [3195](#), [3199](#), [3207](#), [3211](#), [3214](#), [3220](#), [3226](#), [3230](#), [3356](#), [3553](#), [3564](#)  
`\l__enumext_columns_viii_int` [3594](#), [3597](#), [3601](#), [3609](#), [3613](#), [3616](#), [3622](#), [3628](#), [3632](#), [3916](#), [3927](#)  
`\g__enumext_count_item_anskey_int` [71](#), [134](#), [1847](#), [1856](#), [2079](#), [2376](#)  
`\g__enumext_count_item_number_int` [134](#), [1782](#), [1787](#), [1790](#), [1793](#), [1801](#), [1847](#), [1855](#), [2504](#), [2534](#), [3495](#)  
`\g__enumext_count_item_with_ans_int` [65](#)  
`\l__enumext_counter_i_tl` [32](#), [379](#)  
`\l__enumext_counter_ii_tl` [32](#), [380](#)  
`\l__enumext_counter_iii_tl` [32](#), [381](#)  
`\l__enumext_counter_iv_tl` [32](#), [382](#)  
`\c__enumext_counter_style_tl` [27](#), [37](#), [201](#)  
`\g__enumext_counter_styles_tl` [23](#), [32](#), [55](#), [390](#), [408](#)  
`\l__enumext_counter_v_tl` [32](#), [383](#), [628](#)  
`\l__enumext_counter_vi_tl` [32](#), [384](#)  
`\l__enumext_counter_vii_tl` [32](#), [385](#), [558](#)  
`\l__enumext_counter_viii_tl` [32](#), [386](#), [575](#)  
`\l__enumext_current_widest_dim` [23](#), [55](#), [414](#), [491](#), [538](#), [609](#), [613](#)  
`\__enumext_default_item:n` [2498](#), [2498](#), [2547](#)  
`\__enumext_define_counters:Nn` [23](#), [370](#), [370](#), [379](#), [380](#), [381](#), [382](#), [383](#), [384](#), [385](#), [386](#)  
`\__enumext_endminipage:` [29](#), [312](#), [315](#), [1022](#), [3181](#), [3538](#), [3901](#)  
`\__enumext_fake_item:` [763](#), [763](#), [2693](#)  
`\l__enumext_fake_item_indent_v_dim` [782](#), [787](#)  
`\l__enumext_fake_item_indent_v_tl` [784](#), [2555](#), [2559](#), [2567](#)  
`\l__enumext_fake_item_indent_vii_dim` [794](#), [799](#)  
`\l__enumext_fake_item_indent_vii_tl` [796](#), [3534](#)  
`\l__enumext_fake_item_indent_viii_dim` [806](#), [811](#), [3893](#)  
`\l__enumext_fake_item_indent_viii_tl` [808](#), [3892](#), [3896](#)  
`\l__enumext_fake_item_indent_X_tl` [85](#)  
`\__enumext_fake_item_vii:` [763](#), [791](#), [2723](#)  
`\__enumext_fake_item_viii:` [763](#), [803](#), [2728](#)  
`\__enumext_filter_series:n` [53](#), [1462](#), [1462](#), [1504](#), [1516](#), [1521](#)  
`\__enumext_filter_series_key:n` [53](#), [1462](#), [1467](#), [1471](#)  
`\__enumext_filter_series_pair:nn` [53](#), [1462](#), [1468](#), [1480](#)  
`\g__enumext_footnote_arg_seq` [153](#), [2471](#), [2484](#), [2494](#)  
`\g__enumext_footnote_int` [153](#), [2478](#), [2481](#), [2483](#), [2485](#)  
`\g__enumext_footnote_int_seq` [153](#), [2472](#), [2485](#), [2490](#), [2493](#)  
`\__enumext_footnotes_key_bool` [30](#)  
`\l__enumext_footnotes_key_bool` [25](#), [30](#), [96](#), [143](#), [328](#), [333](#), [342](#), [3501](#), [3549](#), [3866](#), [3912](#)  
`\__enumext_footnotetext:nn` [2465](#), [2465](#), [2495](#)  
`\__enumext_getkeyans:nn` [104](#), [3951](#), [3955](#), [3955](#)  
`\__enumext_getkeyans_aux:n` [104](#), [3939](#), [3942](#), [3942](#)  
`\l__enumext_hyperref_bool` [25](#), [30](#), [143](#), [324](#), [345](#), [362](#), [2143](#), [2362](#), [3488](#), [3857](#)  
`\__enumext_hypertarget:nn` [30](#), [319](#), [347](#), [351](#), [367](#)  
`\__enumext_if_is_int:n` [193](#)  
`\__enumext_if_is_int:nTF` [193](#), [647](#), [661](#)  
`\__enumext_is_not_nested:` [28](#), [79](#), [221](#), [221](#), [2756](#), [3326](#)  
`\__enumext_is_on_first_level:` [28](#), [80](#), [221](#), [247](#), [2761](#), [3333](#)  
`\l__enumext_item_column_pos_vii_int` [94](#), [3214](#), [3220](#), [3226](#), [3230](#), [3237](#), [3416](#), [3553](#), [3556](#)

```

\l__enumext_item_column_pos_viii_int .. 100,
    3616, 3622, 3628, 3632, 3639, 3761, 3916, 3919
l__enumext_item_column_pos_X_int ..... 156
\g__enumext_item_count_all_vii_int 94, 3238,
    3417, 3564, 3571
\g__enumext_item_count_all_viii_int 100, 3640,
    3762, 3927, 3934
\g__enumext_item_count_all_X_int ..... 156
\__enumext_item_peek_args_vii: 94, 3418, 3420,
    3420
\__enumext_item_peek_args_viii: 100, 101, 3763,
    3765, 3765
\__enumext_item_starred: .. 76, 2590, 2590, 2608
\l__enumext_item_starred_vii_bool 3435, 3451,
    3505
\l__enumext_item_starred_viii_bool 3780, 3796,
    3870, 3890
\l__enumext_item_starred_X_bool ..... 156
\__enumext_item_std:w 29, 73-75, 89, 306, 310, 2507,
    2513, 2537, 2555, 2559, 2567, 3162
\g__enumext_item_symbol_aux_vii_tl 3459, 3507,
    3510, 3514, 3516
\g__enumext_item_symbol_aux_X_tl ..... 156
\l__enumext_item_symbol_sep_vii_dim .. 3468,
    3476, 3513, 3515
\g__enumext_item_symbol_tl 23, 74, 48, 2522, 2596,
    2613
\l__enumext_item_symbol_vii_tl ..... 3510
\l__enumext_item_text_vii_box 3500, 3541, 3548
\l__enumext_item_text_viii_box 3865, 3904, 3911
\l__enumext_item_text_X_box ..... 156
\l__enumext_item_width_vii_dim ... 3196, 3241,
    3249, 3250
\l__enumext_item_width_viii_dim .. 3598, 3643,
    3651, 3652
\l__enumext_item_width_X_dim ..... 156
\l__enumext_itemindent_X_dim ..... 59
\l__enumext_itemsep_vii_skip ..... 3570
\l__enumext_itemsep_viii_skip ..... 3933
\l__enumext_joined_item_aux_vii_int .. 3235,
    3236, 3237, 3238, 3244
\l__enumext_joined_item_aux_viii_int . 3637,
    3638, 3639, 3640, 3646
\l__enumext_joined_item_aux_X_int .... 156
\__enumext_joined_item_vii:w .. 94, 3423, 3424,
    3426, 3426
\l__enumext_joined_item_vii_int .. 3206, 3207,
    3210, 3212, 3218, 3223, 3228, 3233, 3235, 3241
\__enumext_joined_item_viii:w . 101, 3768, 3769,
    3771, 3771
\l__enumext_joined_item_viii_int . 3608, 3609,
    3612, 3614, 3620, 3625, 3630, 3635, 3637, 3643
\l__enumext_joined_item_X_int ..... 156
\l__enumext_joined_width_vii_dim . 3239, 3246,
    3249, 3531, 3543
\l__enumext_joined_width_viii_dim 3641, 3648,
    3651, 3887, 3906
\l__enumext_joined_width_X_dim ..... 156
\__enumext_keyans_addto_prop:n 69, 2264, 2264,
    2569, 3092
\__enumext_keyans_addto_seq:n . 70, 2338, 2338,
    2571, 3094
\__enumext_keyans_addto_seq_link: 2338, 2356,
    2358, 3827
\__enumext_keyans_anspic_code:nnn . 87, 3083,
    3086, 3086
\__enumext_keyans_default_item:n .. 75, 2550,
    2550, 2586
\l__enumext_keyans_env_bool 20, 2817, 2830, 2978,
    3063
\__enumext_keyans_fake_item: .. 763, 779, 2683
\l__enumext_keyans_item_opt_tl 97, 2383, 2396,
    2402, 3812
\l__enumext_keyans_level_h_int .. 20, 568, 595,
    2316, 3726, 3727
\l__enumext_keyans_level_int .. 20, 1297, 2066,
    2311, 2977, 2982, 3077
\__enumext_keyans_make_label: 32, 76, 2616, 2616,
    2681
\__enumext_keyans_mini_addvspace: 48, 85, 1201,
    1201, 3007
\__enumext_keyans_mini_right_cmd:n 50, 1299,
    1322, 1322
\__enumext_keyans_mini_set_vskip: . 47, 1139,
    1139, 1203
\__enumext_keyans_multi_addvspace: . 85, 988,
    999, 3032
\__enumext_keyans_multi_set_vskip: . 44, 988,
    988, 1001
\__enumext_keyans_multicols_start: 85, 3011,
    3013, 3013
\__enumext_keyans_multicols_stop: . 86, 1326,
    3038, 3038, 3062
\__enumext_keyans_parse_keys:n 2956, 2991, 2991
\l__enumext_keyans_pic_above_int . 129, 3172,
    3173, 3175
\l__enumext_keyans_pic_above_skip .. 89, 129,
    3116, 3156
\__enumext_keyans_pic_arg_two: 88, 3114, 3144,
    3144
\l__enumext_keyans_pic_below_int . 129, 3172,
    3173, 3176
\l__enumext_keyans_pic_body_seq .. 87-89, 129,
    3081, 3121, 3180
\__enumext_keyans_pic_do:n 89, 3121, 3123, 3164,
    3164, 3168
\l__enumext_keyans_pic_level_int .. 20, 1289,
    2070, 2267, 2306, 2341, 2415, 3132, 3133
\__enumext_keyans_pic_row:n 89, 3166, 3169, 3169
\__enumext_keyans_pic_safe_exec: .. 88, 3110,
    3130, 3130
\__enumext_keyans_pic_skip_abs:N .. 88, 3139,
    3139, 3155
\l__enumext_keyans_pic_width_dim . 129, 3171,
    3178
\__enumext_keyans_redefine_item: .. 75, 2574,
    2574, 2680
\__enumext_keyans_ref: ..... 36, 620, 638, 2682
\__enumext_keyans_ref:n ..... 36, 617, 620, 620
\__enumext_keyans_safe_exec: . 2955, 2971, 2971
\__enumext_keyans_save_start_line: . 28, 276,
    276, 2979, 3137, 3731
\__enumext_keyans_show_ans: .. 2379, 2387, 2406
\__enumext_keyans_show_item_opt: . 2379, 2394,
    2567, 3106, 3893
\__enumext_keyans_show_left:n . 75, 2379, 2379,
    2565, 3101
\__enumext_keyans_show_pos: .. 2379, 2391, 2413

```

`\__enumext_keyans_starred_item:n` .. 75, [2562](#), [2562](#), [2582](#)  
`\__enumext_keyans_store_ref:` .. 70, [2285](#), [2285](#), [2570](#), [3093](#), [3825](#)  
`\__enumext_keyans_store_ref_aux_i:` 70, [2285](#), [2297](#), [2300](#)  
`\__enumext_keyans_store_ref_aux_ii:` 70, [2285](#), [2326](#), [2328](#)  
`\l__enumext_keyans_tmpa_dim` ..... 97  
`\l__enumext_keyans_tmpa_tl` 24, [101](#), [97](#), [2564](#), [2568](#)  
`\l__enumext_keyans_tmpb_tl` ..... 101, [97](#)  
`\__enumext_keyans_wrapper_opt:n` .. 1883, [2402](#)  
`\l__enumext_label_copy_i_tl` .. 2197, [2304](#), [2309](#), [2314](#), [2319](#)  
`\l__enumext_label_copy_v_tl` ..... 2314  
`\l__enumext_label_copy_vi_tl` ..... 2309  
`\l__enumext_label_copy_vii_tl` 2172, [2183](#), [2214](#), [2304](#)  
`\l__enumext_label_copy_viii_tl` ..... 2319  
`\l__enumext_label_copy_X_tl` ..... 145  
`\l__enumext_label_fill_left_v_tl` ..... 2620  
`\l__enumext_label_fill_left_X_tl` ..... 85  
`\l__enumext_label_fill_right_v_tl` .... 2627  
`\l__enumext_label_fill_right_X_tl` ..... 85  
`\l__enumext_label_font_style_v_tl` 2621, [3105](#)  
`\l__enumext_label_font_style_vii_tl` ... 3519  
`\l__enumext_label_font_style_viii_tl` .. 3875  
`\l__enumext_label_i_tl` ..... 483  
`\l__enumext_label_ii_tl` ..... 483  
`\l__enumext_label_iii_tl` ..... 483  
`\l__enumext_label_iv_tl` ..... 483  
`\__enumext_label_style:Nnn` 23, 32, [403](#), [403](#), [418](#), [488](#), [535](#), [606](#), [610](#)  
`\l__enumext_label_v_tl` .. 69, 70, [603](#), [2272](#), [2346](#), [2408](#), [2442](#), [2564](#), [2568](#), [2959](#), [3100](#), [3102](#)  
`\l__enumext_label_vi_tl` . 69, 70, [603](#), [2269](#), [2343](#), [3100](#), [3102](#), [3106](#)  
`\l__enumext_label_vii_tl` . 530, [3446](#), [3471](#), [3478](#)  
`\l__enumext_label_viii_tl` 530, [3791](#), [3819](#), [3823](#)  
`\l__enumext_label_width_by_box` .. 55, [399](#), [400](#)  
`\__enumext_label_width_by_box:Nn` 32, [397](#), [397](#), [402](#), [414](#), [671](#)  
`\l__enumext_labelsep_i_dim` ... 2410, [2446](#), [3831](#), [3846](#)  
`\l__enumext_labelsep_v_dim` ..... 3022  
`\l__enumext_labelsep_vii_dim` . 3191, [3200](#), [3242](#), [3469](#), [3529](#), [3545](#)  
`\l__enumext_labelsep_viii_dim` 3593, [3602](#), [3644](#), [3885](#), [3908](#)  
`\l__enumext_labelwidth_i_dim` . 2410, [2445](#), [3831](#), [3846](#)  
`\l__enumext_labelwidth_v_dim` ..... 3022  
`\l__enumext_labelwidth_vii_dim` ... 3191, [3199](#), [3242](#), [3522](#), [3526](#), [3544](#)  
`\l__enumext_labelwidth_viii_dim` .. 3593, [3601](#), [3644](#), [3878](#), [3882](#), [3907](#)  
`\l__enumext_leftmargin_tmp_v_bool` . 88, [3146](#)  
`\l__enumext_leftmargin_tmp_X_bool` ..... 59  
`\l__enumext_leftmargin_tmp_X_dim` ..... 59  
`\l__enumext_leftmargin_X_dim` ..... 59  
`\__enumext_level:` 189, [189](#), [512](#), [515](#), [516](#), [525](#), [527](#), [766](#), [770](#), [774](#), [841](#), [845](#), [849](#), [853](#), [937](#), [939](#), [941](#), [943](#), [976](#), [978](#), [980](#), [982](#), [986](#), [1026](#), [1029](#), [1048](#), [1057](#), [1063](#), [1068](#), [1072](#), [1083](#), [1087](#), [1088](#), [1093](#), [1129](#), [1133](#), [1306](#), [1312](#), [1359](#), [1361](#), [1363](#), [1366](#), [1373](#), [1375](#), [1377](#), [1380](#), [1974](#), [1982](#), [1986](#), [1990](#), [2235](#), [2238](#), [2239](#), [2506](#), [2507](#), [2511](#), [2512](#), [2513](#), [2520](#), [2522](#), [2526](#), [2527](#), [2530](#), [2536](#), [2537](#), [2592](#), [2595](#), [2597](#), [2604](#), [2605](#), [2606](#), [2609](#), [2612](#), [2742](#), [2744](#), [2785](#), [2790](#), [2791](#), [2792](#), [2794](#), [2798](#), [2803](#), [2804](#), [2805](#), [2807](#), [2823](#), [2836](#), [2843](#), [2854](#), [2856](#), [2859](#), [2860](#), [2862](#), [2867](#), [2874](#), [2877](#), [2879](#), [2881](#), [2882](#), [2883](#), [2884](#), [2887](#), [2893](#), [2898](#), [2904](#), [2907](#), [2909](#), [2915](#)  
`\l__enumext_level_h_int` .. 20, [229](#), [253](#), [265](#), [551](#), [588](#), [1779](#), [1796](#), [2191](#), [2208](#), [3327](#), [3328](#)  
`\l__enumext_level_int` . 80, 20, [191](#), [239](#), [252](#), [266](#), [949](#), [1100](#), [1293](#), [1773](#), [2168](#), [2178](#), [2184](#), [2190](#), [2198](#), [2206](#), [2213](#), [2696](#), [2757](#), [2758](#), [2768](#), [2775](#), [2821](#), [2834](#), [2889](#), [2942](#), [2986](#), [3073](#), [3397](#), [3407](#), [3576](#), [3734](#)  
`\__enumext_list_arg_two_i:` ..... 2662  
`\__enumext_list_arg_two_ii:` ..... 2662  
`\__enumext_list_arg_two_iii:` ..... 2662  
`\__enumext_list_arg_two_iv:` ..... 2662  
`\__enumext_list_arg_two_v:` . 75, [2662](#), [2961](#), [3147](#)  
`\__enumext_list_arg_two_vii:` .... 2702, [3309](#)  
`\__enumext_list_arg_two_viii:` .... 2702, [3709](#)  
`\l__enumext_listoffset_v_dim` ..... 3024  
`\l__enumext_listparindent_vii_dim` .... 3532  
`\l__enumext_listparindent_viii_dim` ... 3888  
`\__enumext_make_label:` 32, 73, 74, 76, [2600](#), [2600](#), [2691](#)  
`\l__enumext_mark_answer_sym_tl` . 65, [124](#), [1889](#), [2044](#), [2250](#), [2417](#), [2430](#), [3835](#)  
`\l__enumext_mark_position_str` 124, [1893](#), [1894](#), [1917](#), [1918](#), [2042](#)  
`\l__enumext_mark_ref_sym_tl` .. 124, [1903](#), [2148](#), [2370](#)  
`\__enumext_mini_addvspace:` .. 46, 82, [1122](#), [1122](#), [2864](#)  
`\__enumext_mini_addvspace_vii:` 49, [1275](#), [1275](#), [3267](#)  
`\__enumext_mini_addvspace_viii:` 49, [1275](#), [1281](#), [3669](#)  
`\__enumext_mini_env*` ..... 1016  
`\__enumext_mini_right_cmd:n` 50, [1301](#), [1303](#), [1303](#)  
`\__enumext_mini_set_vskip:` . 45, [1023](#), [1023](#), [1124](#)  
`\__enumext_mini_set_vskip_vii:` 48, [1218](#), [1218](#), [1277](#)  
`\__enumext_mini_set_vskip_viii:` 48, [1218](#), [1240](#), [1283](#)  
`\__enumext_minipage:w` .. 29, [312](#), [314](#), [1018](#), [3178](#), [3531](#), [3887](#)  
`\l__enumext_minipage_active_v_bool` ... 85, 86, [3005](#), [3030](#), [3043](#), [3051](#)  
`\g__enumext_minipage_active_vii_bool` ... 91, [3278](#), [3283](#), [3295](#)  
`\l__enumext_minipage_active_vii_bool` . 3263, [3274](#)  
`\g__enumext_minipage_active_viii_bool` 3680, [3685](#), [3697](#)  
`\l__enumext_minipage_active_viii_bool` 3665, [3676](#)  
`\g__enumext_minipage_active_X_bool` ... 156  
`\l__enumext_minipage_active_X_bool` .... 73  
`\g__enumext_minipage_after_skip` 73, [1222](#), [1234](#), [3293](#), [3695](#)  
`\l__enumext_minipage_after_skip` 45, 46, 83, 86, [73](#), [1039](#), [1054](#), [1074](#), [1090](#), [1105](#), [1111](#), [1117](#), [1131](#), [1141](#), [1150](#), [1153](#), [1165](#), [1183](#), [1194](#), [1210](#), [1242](#), [1255](#), [1269](#), [2924](#), [3060](#)

`\g__enumext_minipage_center_vii_bool` . 3287, 3296  
`\g__enumext_minipage_center_viii_bool` 3689, 3698  
`\g__enumext_minipage_center_X_bool` ... 156  
`\l__enumext_minipage_hsep_v_dim` ... 85, 3003  
`\l__enumext_minipage_hsep_vii_dim` ... 3261  
`\l__enumext_minipage_hsep_viii_dim` ... 3663  
`\l__enumext_minipage_left_skip` 45, 85, 73, 1031, 1046, 1065, 1080, 1127, 1137, 1142, 1148, 1157, 1174, 1186, 1206, 1216, 1220, 1225, 1229, 1243, 1247, 1261, 1279, 1285  
`\l__enumext_minipage_left_v_dim` 85, 3001, 3009  
`\l__enumext_minipage_left_vii_dim` 3257, 3269  
`\l__enumext_minipage_left_viii_dim` 3659, 3671  
`\l__enumext_minipage_left_X_dim` ... 73  
`\g__enumext_minipage_right_skip` 73, 1221, 1226, 1230, 3286, 3688  
`\l__enumext_minipage_right_skip` . 45, 73, 1035, 1050, 1070, 1085, 1143, 1149, 1161, 1179, 1190, 1244, 1251, 1265, 1313, 1330  
`\l__enumext_minipage_right_v_dim` .. 85, 1324, 1329, 2999, 3003  
`\g__enumext_minipage_right_vii_dim` 91, 3265, 3285, 3298  
`\l__enumext_minipage_right_vii_dim` 91, 3255, 3260, 3266  
`\g__enumext_minipage_right_viii_dim` .. 3667, 3687, 3700  
`\l__enumext_minipage_right_viii_dim` .. 3657, 3662, 3668  
`\g__enumext_minipage_right_X_dim` ... 156  
`\g__enumext_minipage_right_X_skip` ... 156  
`\g__enumext_minipage_stat_int` . 82, 85, 73, 1318, 1335, 2863, 2917, 2922, 3006, 3053, 3058  
`\g__enumext_miniright_code_vii_tl` . 91, 3291, 3297  
`\g__enumext_miniright_code_viii_tl` 3693, 3699  
`\g__enumext_miniright_code_X_tl` ... 156  
`\__enumext_multi_addvspace`: ... 43, 82, 971, 971, 2895  
`\__enumext_multi_set_vskip`: .. 43, 935, 935, 973  
`\l__enumext_multicols_above_ii_skip` ... 954  
`\l__enumext_multicols_above_iii_skip` .. 960  
`\l__enumext_multicols_above_iv_skip` ... 966  
`\l__enumext_multicols_above_v_skip` 990, 1004, 1014  
`\l__enumext_multicols_above_X_skip` .... 67  
`\l__enumext_multicols_below_v_skip` 994, 1008, 3045  
`\l__enumext_multicols_below_X_skip` .... 67  
`\__enumext_multicols_start`: 82, 2869, 2871, 2871  
`\__enumext_multicols_stop`: 83, 1308, 2901, 2901, 2926  
`\__enumext_newlabel:nn` 26, 30, 68, 355, 355, 2224, 2332  
`\l__enumext_newlabel_arg_one_tl` 26, 30, 68, 70, 145, 2147, 2217, 2225, 2321, 2333, 2368  
`\l__enumext_newlabel_arg_two_tl` 26, 30, 67, 145, 2171, 2181, 2195, 2211, 2226, 2308, 2313, 2318, 2334  
`\__enumext_parse_keys:n` ... 54, 2738, 2763, 2763  
`\__enumext_parse_keys_vii:n` 54, 3304, 3335, 3335  
`\__enumext_parse_keys_viii:n` . 3705, 3739, 3739  
`\__enumext_parse_series:n` .. 54, 80, 1492, 1492, 2771, 3341  
`\__enumext_parse_store_keys:n` . 80, 2779, 2783, 2783  
`\__enumext_parse_store_keys_vii:n` . 92, 3344, 3348, 3348  
`\l__enumext_parsep_i_skip` 952, 954, 1103, 1151  
`\l__enumext_parsep_ii_skip` .... 958, 960, 1109  
`\l__enumext_parsep_iii_skip` ... 964, 966, 1115  
`\l__enumext_parsep_vii_skip` ..... 3533  
`\l__enumext_parsep_viii_skip` ..... 3889  
`\l__enumext_partopsep_v_skip` . 1006, 1010, 1177, 1181, 1188, 1192, 1208, 1212  
`\l__enumext_partopsep_viii_skip` ..... 1253  
`\__enumext_phantomsection`: 30, 319, 348, 352, 368  
`\__enumext_print_footnote`: ... 2465, 2488, 3551, 3914  
`\__enumext_print_keyans_box:NN` 65, 2036, 2036, 2049, 2237, 2410, 2444, 3831, 3846  
`\l__enumext_print_keyans_i_tl` .... 3965, 3994  
`\l__enumext_print_keyans_ii_tl` ... 3970, 3995  
`\l__enumext_print_keyans_iii_tl` .. 3975, 3996  
`\l__enumext_print_keyans_iv_tl` ... 3980, 3997  
`\l__enumext_print_keyans_vii_tl` .. 3985, 3998  
`\l__enumext_print_keyans_X_tl` ..... 113  
`\__enumext_printkeyans:nnn` 105, 3999, 4002, 4002  
`\__enumext_redefine_item`: . 74, 2539, 2539, 2690  
`\l__enumext_ref_key_arg_tl` 34, 37, 204, 505, 506, 519, 550, 553, 564, 570, 581, 622, 623, 634  
`\l__enumext_ref_the_count_tl` . 34, 37, 512, 515, 518, 558, 560, 563, 575, 577, 580, 628, 630, 633  
`\__enumext_regex_counter_style`: .. 27, 34, 199, 199, 513, 559, 576, 629  
`\__enumext_register_counter_style:Nn` .. 387, 387, 392, 393, 394, 395, 396  
`\__enumext_remove_extra_parsep_vii`: .. 3319, 3560, 3560  
`\__enumext_remove_extra_parsep_viii`: . 3719, 3923, 3923  
`\__enumext_renew_footnote`: ... 2465, 2469, 3503, 3868  
`\l__enumext_renew_the_count_v_tl` 631, 640, 642  
`\l__enumext_renew_the_count_vii_tl` 561, 590, 592  
`\l__enumext_renew_the_count_viii_tl` 578, 597, 599  
`\l__enumext_renew_the_count_X_tl` ..... 37  
`\l__enumext_resume_active_bool` 54, 56, 48, 1496, 1616  
`\l__enumext_resume_bool` ..... 23  
`\__enumext_resume_counter`: .. 55, 56, 1614, 1620, 1627  
`\__enumext_resume_counter:n` . 54, 56, 1585, 1590, 1614, 1614, 1684, 1692  
`\__enumext_resume_counter_save_ans`: .. 56, 57, 1614, 1625, 1657  
`\__enumext_resume_counter_series`: 56, 57, 1614, 1623, 1640  
`\g__enumext_resume_int` . 23, 48, 1537, 1631, 1632  
`\__enumext_resume_last:n` .. 54, 1492, 1498, 1511  
`\l__enumext_resume_name_tl` 48, 1533, 1541, 1544, 1560, 1568, 1571, 1617, 1618, 1646, 1653  
`\__enumext_resume_save_counter`: 54, 1524, 1524, 2938, 3391  
`\__enumext_resume_series:n` . 55, 1456, 1581, 1581



```

\__enumext_resume_starred: . 57, 1457, 1678, 1678
\g__enumext_resume_vii_int . . 93, 48, 1564, 1636,
    1637
\__enumext_safe_exec: . . . . . 79, 2737, 2754, 2754
\__enumext_safe_exec_vii: . . . 3303, 3324, 3324
\__enumext_safe_exec_viii: . . . 3704, 3724, 3724
\l__enumext_series_name_tl . . . . . 56
\l__enumext_series_str . . 54, 80, 1454, 1494, 1502,
    1503, 1505, 1507, 1528, 1531, 1535, 1555, 1558, 1562,
    2767, 3339
\__enumext_set_error:nn . . . . . 4080, 4090, 4092
\__enumext_set_parse:n . . . . . 4063, 4080, 4080
\l__enumext_setkey_tmpa_int . . . 108, 4056, 4060
\l__enumext_setkey_tmpa_seq . . 108, 4054, 4064,
    4070, 4072, 4074, 4087
\l__enumext_setkey_tmpa_tl . . . 108, 4062, 4066
\l__enumext_setkey_tmpb_seq . . 108, 4055, 4058,
    4062, 4063
\l__enumext_setkey_tmpb_tl 108, 4082, 4084, 4085
\l__enumext_show_answer_bool . 124, 1897, 1921,
    2244, 2385, 2399, 3097, 3829
\__enumext_show_length:nnn . . 39, 207, 207, 4132,
    4133, 4134, 4135, 4136, 4137, 4138, 4139, 4140, 4141,
    4147, 4148, 4149, 4150, 4151, 4152, 4153, 4154, 4155,
    4156
\l__enumext_show_position_bool 124, 1900, 1924,
    2248, 2389, 2400, 3098, 3833
\g__enumext_standar_bool . . 28, 20, 228, 231, 251,
    1526, 1591, 1603, 1629, 1642, 1680, 1799, 1831, 2929,
    2948
\l__enumext_standar_bool 83, 20, 2176, 2189, 2205,
    2760, 2937
\l__enumext_standar_first_level_bool . 80, 20,
    256, 1513, 1660, 1711, 1721, 1813
\__enumext_standar_ref: . . . . 34, 503, 523, 2692
\__enumext_standar_ref:n . . . . 34, 495, 503, 503
\g__enumext_standar_series_tl . 48, 1515, 1516,
    1682, 1685
\g__enumext_standard_bool . . . . . 79
\l__enumext_standard_bool . . . . . 80
\l__enumext_standard_first_level_bool . . 28
\__enumext_standard_item_vii:w 94, 3431, 3433,
    3433
\__enumext_standard_item_viii:w . . 101, 3776,
    3778, 3778
\g__enumext_starred_bool 28, 92, 93, 20, 238, 241,
    264, 1553, 1596, 1607, 1634, 1649, 1688, 1778, 1838,
    2167, 2177, 2207, 2302, 2818, 2831, 3299, 3582
\l__enumext_starred_bool . . 92, 93, 20, 2100, 2108,
    2192, 2233, 3332, 3390
\__enumext_starred_columns_set_vii: . . 3185,
    3185, 3312
\__enumext_starred_columns_set_viii: . 3587,
    3587, 3712
\l__enumext_starred_first_level_bool 20, 269,
    1518, 1669, 1715, 1727, 1817
\__enumext_starred_item:nn . . 2516, 2516, 2545
\__enumext_starred_item_exec: . 102, 3821, 3821,
    3872
\__enumext_starred_item_vii:w 94, 95, 3430, 3449,
    3449
\__enumext_starred_item_vii_aux_i:w . . 3449,
    3454, 3457
\__enumext_starred_item_vii_aux_ii:w . 3449,
    3455, 3460, 3462
\__enumext_starred_item_vii_aux_iii:w 3449,
    3465, 3474
\__enumext_starred_item_viii:w 101, 3775, 3794,
    3794
\__enumext_starred_item_viii_aux_i:w . 3794,
    3799, 3802
\__enumext_starred_item_viii_aux_ii:w 3794,
    3800, 3814, 3816
\__enumext_starred_joined_item_vii:n . 90, 94,
    3204, 3204, 3428
\__enumext_starred_joined_item_viii:n . . 98,
    101, 3606, 3606, 3773
\__enumext_starred_ref: . . . . 35, 548, 586, 2720
\__enumext_starred_ref:n . . . . 34, 542, 548, 548
\g__enumext_starred_series_tl . 48, 1520, 1521,
    1690, 1693
\__enumext_start_from:NNn 36, 645, 645, 658, 680
\l__enumext_start_i_int . . . . 1632, 1644, 1663
\__enumext_start_item_tmp_vii: 91, 3315, 3413,
    3413
\__enumext_start_item_tmp_viii: 99, 3715, 3758,
    3758
\__enumext_start_item_vii:w . 94, 95, 3441, 3446,
    3471, 3478, 3480, 3480
\__enumext_start_item_viii:w . . 101, 3786, 3791,
    3819, 3849, 3849
\__enumext_start_list:nn 29, 77, 88, 306, 308, 2741,
    2958, 3111, 3307, 3707
\__enumext_start_mini_vii: . 93, 3253, 3253, 3382
\__enumext_start_mini_viii: . . 100, 3655, 3655,
    3750
\__enumext_start_store_level: . 81, 2740, 2812,
    2812
\__enumext_start_store_level_vii: . 93, 3306,
    3393, 3393
\l__enumext_start_vii_int . . . 1637, 1651, 1672
\l__enumext_start_X_int . . . . . 85, 675
\__enumext_stop_item_tmp_vii: . 91, 94, 95, 3314,
    3318, 3415, 3482
\__enumext_stop_item_tmp_viii: . 99, 100, 3714,
    3718, 3760, 3851
\__enumext_stop_item_vii: 95, 96, 3482, 3536, 3536
\__enumext_stop_item_viii: 103, 3851, 3899, 3899
\__enumext_stop_list: . . 29, 306, 309, 2750, 2968,
    3124, 3320, 3721
\__enumext_stop_mini_vii: 91, 93, 3272, 3272, 3386
\__enumext_stop_mini_viii: 100, 3655, 3674, 3754
\__enumext_stop_store_level: . . 81, 2751, 2812,
    2841
\__enumext_stop_store_level_vii: . . 93, 3321,
    3393, 3403
\l__enumext_store_active_bool 24, 58, 80, 92, 97,
    1661, 1670, 1737, 2062, 2777, 2816, 2829, 2973, 2981,
    3069, 3128, 3342, 3395, 3405, 3733
\__enumext_store_addto_prop:n 63, 69, 1955, 1956,
    1964, 2087, 2283, 3824
\__enumext_store_addto_seq:n 63, 71, 1965, 1965,
    1969, 1976, 1990, 1998, 2007, 2025, 2033, 2151, 2373
\l__enumext_store_ans_bool . 58, 134, 1738, 1760,
    1972, 1996, 2003, 2031, 2075
\l__enumext_store_anskey_arg_tl . . 24, 66, 97,
    2093, 2102, 2104, 2110, 2118, 2121, 2131, 2136, 2139,
    2145, 2151
\__enumext_store_anskey_code:nnnn 65, 66, 2081,

```

[2085](#), [2085](#)  
`\__enumext_store_anskey_show_left:n` [69](#), [2092](#),  
[2242](#), [2242](#)  
`\__enumext_store_anskey_show_wrap:n` [68](#), [2230](#),  
[2230](#), [2246](#), [2261](#)  
`\l__enumext_store_columns_break_bool` . [2056](#),  
[2099](#)  
`\l__enumext_store_columns_join_int` [97](#), [2107](#),  
[2112](#)  
`\l__enumext_store_columns_sep_vii_bool` [3363](#)  
`\l__enumext_store_columns_sep_vii_dim` [3368](#),  
[3372](#)  
`\l__enumext_store_columns_sep_X_bool` . [113](#)  
`\l__enumext_store_columns_sep_X_dim` . . . [113](#)  
`\l__enumext_store_columns_vii_bool` . . . [3350](#)  
`\l__enumext_store_columns_vii_int` [3355](#), [3359](#)  
`\l__enumext_store_columns_X_bool` . . . . . [113](#)  
`\l__enumext_store_columns_X_int` . . . . . [113](#)  
`\__enumext_store_internal_ref:` . . [66](#), [67](#), [2090](#),  
[2153](#), [2153](#)  
`\l__enumext_store_item_symbol_sep_dim` [2054](#),  
[2128](#), [2133](#)  
`\l__enumext_store_item_symbol_tl` . [2052](#), [2119](#),  
[2123](#)  
`\l__enumext_store_keyans_item_opt_sep_-`  
`tl` . . . . [1886](#), [2277](#), [2279](#), [2350](#), [2352](#), [3807](#), [3809](#)  
`\l__enumext_store_keyans_item_opt_tl` . . . [97](#)  
`\l__enumext_store_keyans_label_tl` [24](#), [69](#), [71](#),  
[97](#), [2266](#), [2269](#), [2272](#), [2279](#), [2281](#), [2283](#), [2340](#), [2343](#),  
[2346](#), [2352](#), [2354](#), [2364](#), [2373](#), [3804](#), [3809](#), [3810](#), [3823](#),  
[3824](#), [3826](#)  
`\__enumext_store_level_close:` . [63](#), [1970](#), [1994](#),  
[2845](#)  
`\__enumext_store_level_close_vii:` [2001](#), [2029](#),  
[3409](#)  
`\__enumext_store_level_open:` . . [62](#), [63](#), [80](#), [1970](#),  
[1970](#), [2824](#), [2837](#)  
`\__enumext_store_level_open_vii:` . . [92](#), [2001](#),  
[2001](#), [3399](#)  
`\g__enumext_store_name_tl` [24](#), [83](#), [97](#), [1834](#), [1841](#),  
[1849](#), [1853](#), [2932](#), [2950](#), [3584](#)  
`\l__enumext_store_name_tl` [24](#), [58](#), [97](#), [1547](#), [1550](#),  
[1574](#), [1577](#), [1665](#), [1674](#), [1708](#), [1709](#), [1724](#), [1730](#), [1739](#),  
[1741](#), [1743](#), [1745](#), [1747](#), [1749](#), [1811](#), [1834](#), [1841](#), [1958](#),  
[1960](#), [1967](#), [2219](#), [2220](#), [2256](#), [2323](#), [2324](#), [2423](#), [2436](#),  
[2932](#), [3841](#)  
`\l__enumext_store_opt_vii_tl` . [2005](#), [2015](#), [2021](#),  
[2025](#), [3357](#), [3370](#)  
`\l__enumext_store_opt_X_tl` . . . . . [113](#)  
`\l__enumext_store_ref_key_bool` [66](#), [1906](#), [2088](#),  
[2142](#), [2287](#), [2361](#)  
`\l__enumext_store_upper_level_X_bool` . . [113](#)  
`\l__enumext_store_write_aux_file_tl` [26](#), [68](#), [70](#),  
[145](#), [2222](#), [2228](#), [2330](#), [2336](#)  
`\__enumext_storing_exec:` . [58](#), [1706](#), [1725](#), [1731](#),  
[1735](#)  
`\__enumext_storing_set:n` . . [58](#), [1701](#), [1706](#), [1706](#)  
`\l__enumext_the_counter_v_tl` . . . . . [630](#)  
`\l__enumext_the_counter_vii_tl` . . . . . [560](#)  
`\l__enumext_the_counter_viii_tl` . . . . . [577](#)  
`\l__enumext_the_counter_X_tl` . . . . . [37](#)  
`\__enumext_tmp:n` [32](#), [36](#), [41](#), [47](#), [59](#), [66](#), [67](#), [72](#), [79](#), [84](#),  
[85](#), [96](#), [113](#), [123](#), [148](#), [152](#), [156](#), [175](#), [215](#), [219](#), [758](#), [762](#),  
[1450](#), [1461](#), [1697](#), [1705](#), [1752](#), [1770](#), [1876](#), [1911](#), [1912](#),  
[1929](#), [2155](#), [2162](#), [2163](#), [2184](#), [2198](#), [2201](#), [2213](#), [2289](#),  
[2296](#), [2662](#), [2701](#), [2702](#), [2734](#)  
`\__enumext_tmp:nn` [419](#), [440](#), [441](#), [469](#), [470](#), [482](#), [675](#),  
[694](#), [739](#), [757](#), [815](#), [823](#), [824](#), [838](#), [903](#), [919](#), [920](#), [934](#),  
[1339](#), [1355](#), [1930](#), [1954](#), [2449](#), [2464](#)  
`\__enumext_tmp:nnn` [483](#), [499](#), [500](#), [501](#), [502](#), [530](#), [546](#),  
[547](#)  
`\__enumext_tmp:nnnnnn` [695](#), [720](#), [723](#), [726](#), [728](#), [730](#),  
[733](#), [736](#)  
`\__enumext_tmp:w` . . . . . [3948](#), [3951](#)  
`\l__enumext_tmpa_vii_int` . . . . . [3195](#), [3198](#)  
`\l__enumext_tmpa_viii_int` . . . . . [3597](#), [3600](#)  
`\l__enumext_tmpa_X_int` . . . . . [156](#)  
`\l__enumext_topsep_v_skip` [992](#), [996](#), [1146](#), [1159](#),  
[1167](#), [1172](#), [1192](#), [1196](#), [3127](#), [3159](#)  
`\l__enumext_topsep_vii_skip` . . [1223](#), [1232](#), [1236](#)  
`\l__enumext_topsep_viii_skip` . [1245](#), [1267](#), [1271](#)  
`\l__enumext_vspace_a_star_v_bool` . . . . . [1388](#)  
`\l__enumext_vspace_a_star_vii_bool` . . . [1410](#)  
`\l__enumext_vspace_a_star_viii_bool` . . [1421](#)  
`\l__enumext_vspace_a_star_X_bool` . . . . . [85](#)  
`\__enumext_vspace_above:` . . [51](#), [1356](#), [1356](#), [2850](#)  
`\__enumext_vspace_above_v:` . [52](#), [1384](#), [1384](#), [2997](#)  
`\l__enumext_vspace_above_v_skip` . . [1386](#), [1390](#),  
[1392](#)  
`\__enumext_vspace_above_vii:` . . [52](#), [1406](#), [1406](#),  
[3379](#)  
`\l__enumext_vspace_above_vii_skip` [1408](#), [1412](#),  
[1414](#)  
`\__enumext_vspace_above_viii:` . [52](#), [1406](#), [1417](#),  
[3748](#)  
`\l__enumext_vspace_above_viii_skip` [1419](#), [1423](#),  
[1425](#)  
`\l__enumext_vspace_b_star_v_bool` . . . . . [1399](#)  
`\l__enumext_vspace_b_star_vii_bool` . . . [1432](#)  
`\l__enumext_vspace_b_star_viii_bool` . . [1443](#)  
`\l__enumext_vspace_b_star_X_bool` . . . . . [85](#)  
`\__enumext_vspace_below:` . . [51](#), [1370](#), [1370](#), [2936](#)  
`\__enumext_vspace_below_v:` . [52](#), [1395](#), [1395](#), [3065](#)  
`\l__enumext_vspace_below_v_skip` . . [1397](#), [1401](#),  
[1403](#)  
`\__enumext_vspace_below_vii:` . . [52](#), [1428](#), [1428](#),  
[3389](#)  
`\l__enumext_vspace_below_vii_skip` [1430](#), [1434](#),  
[1436](#)  
`\__enumext_vspace_below_viii:` . [52](#), [1428](#), [1439](#),  
[3756](#)  
`\l__enumext_vspace_below_viii_skip` [1441](#), [1445](#),  
[1447](#)  
`\__enumext_widest_from:nnNn` . . [36](#), [659](#), [659](#), [674](#),  
[686](#)  
`\g__enumext_widest_label_tl` [23](#), [32](#), [55](#), [407](#), [411](#),  
[415](#)  
`\l__enumext_wrap_label_opt_v_bool` . . . . [2558](#)  
`\l__enumext_wrap_label_opt_vii_bool` [94](#), [3440](#)  
`\l__enumext_wrap_label_opt_viii_bool` . . [101](#),  
[3785](#)  
`\l__enumext_wrap_label_opt_X_bool` . . . . . [85](#)  
`\l__enumext_wrap_label_v_bool` [2554](#), [2558](#), [2566](#),  
[2622](#)  
`\l__enumext_wrap_label_vii_bool` [94](#), [3439](#), [3444](#),  
[3452](#), [3520](#)  
`\l__enumext_wrap_label_viii_bool` . [101](#), [3784](#),  
[3789](#), [3797](#), [3876](#)  
`\l__enumext_wrap_label_X_bool` . . . . . [85](#)

<code>\__enumext_wrapper_label_v:n</code> . . . . .	2624, 3106
<code>\__enumext_wrapper_label_vii:n</code> . . . . .	3523
<code>\__enumext_wrapper_label_viii:n</code> . . . . .	3879
<code>\__enumext_zero_count_level:</code> . . . . .	213, 213
<code>\__enumext_zero_parsep:</code> . . . . .	46, 1043, 1098, 1098
<code>enumext*</code> . . . . .	5, 3301
<code>enumXi</code> . . . . .	379
<code>enumXii</code> . . . . .	379
<code>enumXiii</code> . . . . .	379
<code>enumXiv</code> . . . . .	379
<code>enumXv</code> . . . . .	379
<code>enumXvi</code> . . . . .	379
<code>enumXvii</code> . . . . .	379
<code>enumXviii</code> . . . . .	379

#### Environments provide by `enumext`:

<code>enumext*</code> . . . . .	22, 23, 25–28, 31, 34, 35, 38, 39, 41, 42, 48, 49, 52–55, 57–59, 62–68, 70, 72, 73, 79, 81, 92, 93, 95, 97, 98, 100, 102, 105, 108, 110
<code>enumext</code> . . . . .	22, 23, 25, 27, 28, 31–34, 36–40, 42–51, 53–55, 57–68, 70, 72–77, 79–81, 83, 84, 88, 89, 91, 93, 105, 108, 109
<code>keyans*</code> . . . . .	22–24, 26–28, 31, 34–36, 38, 39, 41, 42, 48, 49, 52, 58, 59, 61–63, 69, 73, 79, 99, 100, 108, 110
<code>keyanspic</code> . . . . .	22–25, 28, 31, 32, 35, 49, 58, 59, 61, 63, 69–71, 86–89, 109
<code>keyans</code> . . . . .	22–25, 27, 28, 31, 32, 35, 37–42, 44, 47–52, 58, 59, 61–63, 69–71, 75–77, 84, 86–88, 91, 100, 108, 109

#### Environments:

<code>list</code> . . . . .	26, 29, 77, 79
<code>lrbox</code> . . . . .	89, 96, 103
<code>minipage</code> . . . . .	26, 29, 42, 44, 86–89, 96, 103
<code>multicols</code> . . . . .	42–45, 50, 82, 83, 85, 86

#### exp commands:

<code>\exp_after:wN</code> . . . . .	3951
<code>\exp_args:Ne</code> . . . . .	2774, 3939
<code>\exp_not:N</code> . . . . .	45, 410, 518, 563, 580, 633, 772, 786, 787, 798, 799, 810, 811, 2147, 2253, 2254, 2366, 2420, 2421, 2433, 2434, 3838, 3839, 3948
<code>\exp_not:n</code> . . . . .	259, 272, 284, 292, 300, 518, 519, 563, 564, 580, 581, 633, 634, 773, 1478, 1490, 1938, 1945, 2112, 2123, 2133, 2147, 2148, 2225, 2333, 2368, 2370, 2794, 2807, 3359, 3372

## F

<code>\fbox</code> . . . . .	1881
file commands:	
<code>\file_input_stop:</code> . . . . .	4274
<code>first</code> . . . . .	824
<code>font</code> . . . . .	419
<code>\footnote</code> . . . . .	73
<code>\footnote</code> . . . . .	73, 2473
<code>\footnotemark</code> . . . . .	2483
<code>\footnotesize</code> . . . . .	2254, 2421, 2434, 3839
<code>\footnotetext</code> . . . . .	2467

## G

<code>\getkeyans</code> . . . . .	14, 104, 3937
group commands:	
<code>\group_begin:</code> . . . . .	2074, 2252, 2419, 2432, 3499, 3518, 3837, 3864, 3874, 3959, 3993
<code>\group_end:</code> . . . . .	2083, 2259, 2426, 2439, 3528, 3540, 3844, 3884, 3903, 3961, 4000

## H

<code>\hbadness</code> . . . . .	3547, 3910
----------------------------------	------------

#### hbox commands:

<code>\hbox_set:Nn</code> . . . . .	399
<code>\hfill</code> . . . . .	449, 453, 458, 459, 1310, 1328, 2147, 2366, 3277, 3679
hook commands:	
<code>\hook_gput_code:nnn</code> . . . . .	9, 183, 187, 317
<code>\hook_gset_rule:nnnn</code> . . . . .	318
<code>\hspace</code> . . . . .	3558, 3921
<code>\hyperlink</code> . . . . .	67, 71
<code>\hyperlink</code> . . . . .	2147, 2366
<code>\hypertarget</code> . . . . .	30
<code>\hypertarget</code> . . . . .	347

## I

<code>\IfHyperBoolean</code> . . . . .	325
<code>\IfPackageLoadedTF</code> . . . . .	11, 321, 335
<code>\ignorespaces</code> . . . . .	775
<code>\inputlineno</code> . . . . .	259, 272, 284, 292, 300
int commands:	
<code>\int_add:Nn</code> . . . . .	3237, 3639
<code>\int_case:nn</code> . . . . .	949, 1100, 1773, 1796
<code>\int_compare:nNnTF</code> . . . . .	551, 568, 588, 595, 1025, 1144, 1289, 1293, 1297, 1846, 1861, 1867, 2066, 2070, 2267, 2306, 2311, 2316, 2341, 2415, 2758, 2768, 2821, 2834, 2873, 2889, 2903, 2917, 2942, 2982, 2986, 3015, 3040, 3053, 3073, 3077, 3133, 3207, 3217, 3233, 3328, 3397, 3407, 3553, 3562, 3576, 3609, 3619, 3635, 3727, 3734, 3916, 3925, 4060
<code>\int_compare_p:nNn</code> . . . . .	229, 239, 252, 253, 265, 266, 1779, 2168, 2178, 2190, 2191, 2206, 2208
<code>\int_decr:N</code> . . . . .	3236, 3638
<code>\int_eval:n</code> . . . . .	1960, 2220, 2254, 2324, 2421, 2434, 2677, 2719, 3225, 3627, 3839
<code>\int_from_alph:n</code> . . . . .	653, 667
<code>\int_from_roman:n</code> . . . . .	655, 669
<code>\int_gadd:Nn</code> . . . . .	3238, 3640
<code>\int_gdecr:N</code> . . . . .	1782, 1787, 1790, 1793, 1801
<code>\int_gincr:N</code> . . . . .	1631, 1636, 2079, 2376, 2504, 2534, 2572, 2863, 3006, 3095, 3417, 3495, 3762, 3828
<code>\int_gset:Nn</code> . . . . .	2481
<code>\int_gset_eq:NN</code> . . . . .	1530, 1537, 1543, 1549, 1557, 1564, 1570, 1576, 2478
<code>\int_gzero:N</code> . . . . .	217, 1318, 1335, 1855, 1856, 1873, 2922, 3058, 3571, 3934
<code>\int_if_exist:NTF</code> . . . . .	1505, 1541, 1547, 1568, 1574, 1747
<code>\int_incr:N</code> . . . . .	2757, 2977, 3132, 3327, 3416, 3726, 3761
<code>\int_mod:nn</code> . . . . .	3564, 3927
<code>\int_new:N</code> . . . . .	20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 110, 118, 131, 132, 140, 141, 142, 153, 159, 160, 161, 162, 163, 1507, 1749
<code>\int_set:Nn</code> . . . . .	649, 653, 655, 1644, 1651, 1663, 1672, 1935, 2107, 3172, 3173, 3195, 3206, 3212, 3228, 3547, 3597, 3608, 3614, 3630, 3910, 4056
<code>\int_set_eq:NN</code> . . . . .	1632, 1637, 2789, 3235, 3354, 3637
<code>\int_step_function:nnN</code> . . . . .	2184, 2198, 2213
<code>\int_step_inline:nnn</code> . . . . .	3174, 4083
<code>\int_to_roman:n</code> . . . . .	191, 2164, 2202
<code>\int_use:N</code> . . . . .	1026, 1646, 1653, 1665, 1674, 2677, 2696, 2719, 2775, 2874, 2883, 2898, 2904, 3210, 3211, 3223, 3612, 3613, 3625
<code>\int_zero:N</code> . . . . .	3556, 3919
<code>\c_one_int</code> . . . . .	3195, 3214, 3220, 3226, 3230, 3233, 3597, 3616, 3622, 3628, 3632, 3635
<code>\c_zero_int</code> . . . . .	2168, 2178, 2190, 2191, 2206, 2208, 3397, 3407, 3567, 3930
<code>\item</code> . . . . .	29, 40, 41, 64, 73, 86, 87, 89, 91, 99

`\item` 73, 75, 94, 95, 100, 102, 310, 1978, 1984, 2009, 2017, 2104, 2343, 2346, 2541, 2576, 3313, 3315, 3713, 3715, 3826

`\item*` ..... 6, 12, 61, 2574

`item-pos*` ..... 2449

`item-sym*` ..... 2449

`\itemindent` ..... 23, 78

`\itemindent` ..... 77

`itemindent` ..... 739

`\itemsep` ..... 87, 88

`\itemsep` ..... 3148, 3154

`\itemwidth` ..... 3202, 3246, 3250, 3604, 3648, 3652

K

`keyans` ..... 12, 2953

`keyans*` ..... 12, 3702

`keyanspic` ..... 13, 3108

Keys for environments provide by `enumext`:

`above*` ..... 24, 51, 52

`above` ..... 24, 51, 52, 81, 84, 93, 100

`after` ..... 40, 41, 83, 86, 93, 100

`align` ..... 24, 32, 76, 96

`before*` ..... 40, 41, 81, 93, 100

`before` ..... 40, 41, 84

`below*` ..... 24, 51, 52

`below` ..... 24, 51, 52, 83, 86, 93, 100

`check-ans` 24, 25, 27, 28, 58, 59, 61, 65, 71, 73, 74, 81, 83, 97, 108

`columns-sep*` ..... 25, 62, 80, 92

`columns-sep` ..... 42, 63, 80, 82, 85, 92

`columns*` ..... 25, 62, 80, 92

`columns` ..... 24, 42, 45, 51, 63, 80, 82, 85, 92

`first` ..... 40, 41, 96

`font` ..... 32, 76, 96

`item-pos*` ..... 65, 66, 72

`item-sym*` ..... 23, 65, 66, 72, 74

`item*-sep` ..... 74

`itemindent` ..... 24, 38, 75, 96

`itemsep` ..... 37, 79

`labelsep` ..... 32, 74, 78, 96

`labelwidth` ..... 31-35, 37, 78

`label` ..... 23, 31-33, 36, 37, 89

`lisparindent` ..... 79

`list-indent` ..... 23, 38, 88

`list-offset` ..... 38

`listparindent` ..... 38, 96

`mark-ans` ..... 25, 62, 69

`mark-pos` ..... 62

`mark-ref` ..... 25, 62, 67

`mini-env` .. 24, 42, 44, 50, 51, 73, 81, 85, 91, 93, 98, 100

`mini-sep` ..... 24, 42, 81, 85

`miniright*` ..... 24, 42

`miniright` ..... 24, 42, 49, 91

`minirigth*` ..... 27

`minirigth` ..... 27

`no-store` ..... 25, 58-60

`noitemsep` ..... 37, 46

`nosep` ..... 37, 46

`parindent` ..... 79

`parsep` ..... 37, 79, 96

`partopsep` ..... 37

`ref` ..... 23, 27, 33-35, 108

`resume*` ..... 53, 54, 57, 58, 83

`resume` ..... 23, 53-58, 83, 93

`rightmargin` ..... 38

`save-ans` 24, 53-58, 63, 65, 69, 71, 75, 80, 84, 87, 93, 100, 102, 104, 105, 108

`save-key` ..... 25, 53, 80

`save-ref` ..... 26, 30, 62, 66, 67, 70, 71, 75, 102

`save-sep` ..... 62

`series` ..... 53-57, 80, 83

`show-ans` ..... 25, 62, 65, 66, 69, 75, 101, 102

`show-length` ..... 27, 39, 108

`show-pos` ..... 25, 62, 65, 66, 69, 71, 75, 101, 102

`start` ..... 24, 27, 36, 37, 53

`store-brk` ..... 65, 66

`topsep` ..... 37

`widest` ..... 23, 27, 36, 37

`wrap-ans` ..... 62, 65, 68

`wrap-label*` ..... 32, 73, 76, 94, 96, 101

`wrap-label` ..... 32, 76, 94, 96, 101

`wrap-opt` ..... 62

keys commands:

`\keys_define:nn` 421, 443, 472, 485, 532, 603, 677, 697, 741, 760, 817, 826, 905, 922, 1341, 1452, 1699, 1754, 1878, 1914, 1932, 2050, 2451, 3963, 4026

`\l_keys_key_str` ..... 4117

`\keys_set:nn` . 435, 929, 1346, 1351, 1593, 1598, 1685, 1693, 2096, 2770, 2774, 2993, 3340, 3743, 4028, 4029, 4030, 4031, 4032, 4033, 4034, 4035, 4036, 4037, 4038, 4039, 4077

keyval commands:

`\keyval_parse:NNn` ..... 1466

L

`label` ..... 483, 530, 603

Labels provide by `enumext`:

`\Alph*` ..... 31, 32

`\Roman*` ..... 31, 32

`\alph*` ..... 31, 32

`\arabic*` ..... 27, 31, 32

`\roman*` ..... 31, 32

`\labelsep` ..... 88

`\labelsep` ..... 3149, 3152

`labelsep` ..... 419

`\labelwidth` ..... 32, 88

`\labelwidth` ..... 3149, 3150

`labelwidth` ..... 419

`\leftmargin` ..... 23, 78

`\leftmargin` ..... 77, 3149

legacy commands:

`\legacy_if:nTF` ..... 3483, 3486, 3852, 3855

`\legacy_if_gset_false:n` ..... 1019

`\legacy_if_set_false:n` ..... 3485, 3854

`\legacy_if_set_true:n` 3445, 3470, 3477, 3490, 3790, 3818, 3859

`\linewidth` ..... 81, 85

`\linewidth` .... 2858, 3003, 3171, 3198, 3259, 3600, 3661

`\list` ..... 29

`\list` ..... 308

`list-indent` ..... 739

`list-offset` ..... 739

`\listparindent` ..... 3151

`listparindent` ..... 739

`\lrbox` ..... 3500, 3865

M

`\makebox` ..... 89

`\makebox` .. 2040, 2042, 2596, 3514, 3522, 3526, 3878, 3882

`\makelabel` ..... 73, 76, 89



<code>\makelabel</code> .....	76, 2602, 2618
<code>\makesavenoteenv</code> .....	341
<code>mark-ans</code> .....	1876
<code>mark-pos</code> .....	1876, 1912
<code>mark-ref</code> .....	1876
<code>mini-env</code> .....	903
<code>mini-sep</code> .....	903
<code>\minipage</code> .....	29
<code>\minipage</code> .....	314
<code>\miniright</code> .....	10, 49, 1287, 2920, 3056
<code>\miniright*</code> .....	10
mode commands:	
<code>\mode_if_vertical:TF</code> .....	974, 1002, 1125, 1204
<code>\mode_leave_vertical:</code> ..	772, 786, 798, 810, 2009, 2017, 2038, 2594, 3512
msg commands:	
<code>\msg_error:nn</code> ..	2984, 2988, 3075, 3135, 3330, 3729, 3736, 4040
<code>\msg_error:nnn</code> ..	508, 555, 572, 625, 1291, 1295, 1320, 1337, 1605, 1609, 1713, 1717, 3953, 3958, 4023, 4093
<code>\msg_error:nnnn</code> ..	1815, 1819, 2064, 2068, 2072, 2975, 3071, 3079
<code>\msg_fatal:nn</code> .....	2759
<code>\msg_fatal:nnn</code> .....	373
<code>\msg_info:nnn</code> .....	13, 16, 323, 337
<code>\msg_line_context:</code> ..	4121, 4126, 4131, 4146, 4161, 4165, 4169, 4173, 4190, 4202, 4207, 4212, 4217, 4222, 4226, 4231, 4236, 4240, 4245, 4249, 4254, 4259, 4264, 4268, 4272
<code>\msg_new:nnn</code> ..	4094, 4098, 4102, 4106, 4111, 4115, 4119, 4124, 4129, 4144, 4159, 4163, 4167, 4171, 4175, 4182, 4187, 4192, 4196, 4200, 4205, 4210, 4215, 4220, 4224, 4229, 4234, 4238, 4243, 4247, 4252, 4257, 4262, 4266, 4270
<code>\msg_note:nnnn</code> .....	1723, 1729
<code>\msg_term:nnn</code> .....	1849
<code>\msg_term:nnnn</code> .....	2686, 2696, 2725, 2730
<code>\msg_warning:nn</code> .....	2919, 3055
<code>\msg_warning:nnnn</code> ..	1852, 1864, 1870, 2634, 2639, 3209, 3222, 3611, 3624
<code>\multicolsep</code> .....	82, 85
<code>\multicolsep</code> .....	2888, 3028
<b>N</b>	
<code>\NeedsTeXFormat</code> .....	3
<code>\newcounter</code> .....	376
<code>\NewDocumentCommand</code> ..	1287, 2060, 3067, 3937, 3991, 4047
<code>\NewDocumentEnvironment</code> ..	2735, 2953, 3108, 3301, 3702
<code>\newlabel</code> .....	30
<code>\newlabel</code> .....	359
<code>no-store</code> .....	1752
<code>\noindent</code> .....	91, 99
<code>\noindent</code> ..	2865, 3008, 3268, 3314, 3555, 3670, 3714, 3918
<code>\nointerlineskip</code> .....	2865, 3008, 3268, 3670
<code>noitemsep</code> .....	695
<code>\nopagebreak</code> .....	985, 1013, 1136, 1215, 1278, 1284
<code>\normalfont</code> .....	2253, 2420, 2433, 3838
<code>nosep</code> .....	695
<b>P</b>	
Packages:	
enumext .....	22, 33, 58, 77, 86, 107
enumitem .....	31
expl3 .....	89
footnotehyper .....	30
hyperref .....	25, 26, 30, 67, 71, 95, 107
lua-visual-debug .....	44
multicol .....	22, 107
shortlst .....	89
<code>\par</code> ..	985, 1013, 1136, 1215, 1278, 1284, 1313, 1330, 2232, 2909, 2924, 3045, 3060, 3183, 3286, 3293, 3555, 3569, 3688, 3695, 3918, 3932
<code>\parindent</code> .....	3532, 3888
<code>\parsep</code> .....	43, 46, 87, 88
<code>\parsep</code> .....	2010, 2018, 2716, 3148, 3155, 3160
<code>parsep</code> .....	695
<code>\parskip</code> .....	3533, 3889
<code>\partopsep</code> .....	88
<code>\partopsep</code> .....	2717, 3153
<code>partopsep</code> .....	695
peek commands:	
<code>\peek_meaning:N</code> ..	3422, 3436, 3453, 3464, 3767, 3781, 3798
<code>\peek_meaning_remove:N</code> ..	3429, 3774
<code>\peek_remove_spaces:n</code> ..	2580
<code>\phantomsection</code> .....	30
<code>\phantomsection</code> .....	348
prg commands:	
<code>\prg_do_nothing:</code> .....	352
<code>\prg_new_protected_conditional:Npnn</code> ..	193
<code>\prg_replicate:nn</code> .....	210, 4180
<code>\prg_return_false:</code> .....	197
<code>\prg_return_true:</code> .....	196
<code>\printkeyans</code> .....	14, 105, 3991
prop commands:	
<code>\prop_count:N</code> ..	1960, 2220, 2256, 2324, 2423, 2436, 3841
<code>\prop_gput_if_not_in:Nnn</code> .....	1955, 1958
<code>\prop_if_exist:N</code> ..	1739, 3957
<code>\prop_item:Nn</code> .....	3960
<code>\prop_new:N</code> .....	1741
<code>\ProvidesExplPackage</code> .....	4
<b>R</b>	
<code>\raggedcolumns</code> .....	2897, 3034
<code>\ref</code> .....	67, 69
<code>ref</code> .....	483, 530, 603
<code>\refstepcounter</code> .....	3492, 3861
regex commands:	
<code>\regex_match:nnTF</code> ..	195, 652, 654, 666, 668, 2787, 2800, 3352, 3365
<code>\regex_replace_once:nnN</code> .....	203
<code>\renewcommand</code> .....	518, 563, 580, 633
<code>\RenewDocumentCommand</code> ..	2473, 2541, 2576, 2602, 2618
<code>\RequirePackage</code> .....	17
<code>resume</code> .....	1450
<code>resume*</code> .....	1450
<code>rightmargin</code> .....	739
<code>\Roman</code> .....	32, 36, 37
<code>\Roman</code> .....	395
<code>\roman</code> .....	32, 36, 37
<code>\roman</code> .....	396, 501, 3979
<b>S</b>	
<code>save-ans</code> .....	1697
<code>save-ref</code> .....	1876
<code>save-sep</code> .....	1876
scan commands:	
<code>\scan_stop:</code> .....	89, 3162, 3313, 3713, 3948, 3951
seq commands:	
<code>\seq_clear:N</code> .....	4054

<code>\seq_const_from_clist:Nn</code> . . . . .	4042	<code>\str_if_empty:NTF</code> . . . . .	1494, 1535, 1562
<code>\seq_count:N</code> . . . . .	3121, 4058	<code>\str_if_eq:nnTF</code> . . . . .	2678, 2721
<code>\seq_gclear:N</code> . . . . .	2471, 2472	<code>\str_if_in:nnTF</code> . . . . .	3944
<code>\seq_gput_right:Nn</code> . . . . .	1967, 2484, 2485	<code>\str_new:N</code> . . . . .	128, 168
<code>\seq_if_empty:NTF</code> . . . . .	2490, 4006, 4072	<code>\str_set:Nn</code> . . . . .	475, 476, 477, 1893, 1894, 1917, 1918
<code>\seq_if_exist:NTF</code> . . . . .	1743, 4004	<code>\string</code> . . . . .	341
<code>\seq_item:Nn</code> . . . . .	3180	<code>\strutbox</code> . . . . .	1033, 1037, 1041, 1052, 1056, 1067, 1076, 1082, 1092, 1105, 1111, 1117, 1148, 1149, 1150, 1153, 1163, 1167, 1176, 1183, 1188, 1196, 1225, 1226, 1229, 1236, 1249, 1257, 1263, 1271, 3158
<code>\seq_map_function:NN</code> . . . . .	4063	<b>T</b>	
<code>\seq_map_inline:Nn</code> . . . . .	4012, 4017, 4051, 4073, 4074		
<code>\seq_map_pairwise_function:NNN</code> . . . . .	2492	TeX and $\LaTeX$ 2 <sub>ε</sub> commands:	
<code>\seq_new:N</code> . . . . .	111, 112, 129, 154, 155, 1745	<code>\@auxout</code> . . . . .	357
<code>\seq_pop_left:NN</code> . . . . .	4062	<code>\@currentenv</code> . . . . .	223, 278
<code>\seq_put_right:Nn</code> . . . . .	3081, 4070, 4087	<code>\protected@write</code> . . . . .	357
<code>\seq_set_from_clist:Nn</code> . . . . .	4055	text commands:	
<code>\seq_set_map_e:NNn</code> . . . . .	4064	<code>\text_expand:n</code> . . . . .	3940
<code>\seq_show:N</code> . . . . .	4008	<code>\textasteriskcentered</code> . . . . .	1890, 1904
<code>series</code> . . . . .	1450	<code>\thepage</code> . . . . .	363
<code>\setcounter</code> . . . . .	663, 667, 669, 2677, 2719, 3126	tl commands:	
<code>\setenumext</code> . . . . .	6–9, 106, 3967, 3972, 3977, 3982, 3987, 4047	<code>\c_space_tl</code> . . . . .	2402, 4131, 4146
<code>\setlength</code> . . . . .	2011, 2019	<code>\tl_clear:N</code> . . . . .	448, 454, 1874, 2093, 2266, 2340, 3804
<code>show-ans</code> . . . . .	1876, 1912	<code>\tl_clear_new:N</code> . . . . .	405
<code>show-length</code> . . . . .	815	<code>\tl_const:Nn</code> . . . . .	37, 389
<code>show-pos</code> . . . . .	1912	<code>\tl_gclear:N</code> . . . . .	1515, 1520, 1857, 2613, 2950, 3297, 3516, 3584, 3699
skip commands:		<code>\tl_gclear_new:N</code> . . . . .	1502
<code>\skip_add:Nn</code> . . . . .	954, 960, 966, 976, 980, 1004, 1008, 1105, 1111, 1117, 1127, 1131, 1153, 1206, 1210, 3148	<code>\tl_gput_right:Nn</code> . . . . .	390
<code>\skip_eval:n</code> . . . . .	2010, 2018	<code>\tl_greplace_all:Nnn</code> . . . . .	411
<code>\skip_gset:Nn</code> . . . . .	1226, 1230, 1234	<code>\tl_gset:Nn</code> . . . . .	257, 270, 1503, 1516, 1521, 1834, 1841, 2932, 3459
<code>\skip_gzero_new:N</code> . . . . .	1221, 1222	<code>\tl_gset_eq:NN</code> . . . . .	407, 2522, 3509
<code>\skip_horizontal:N</code> . . . . .	787, 799, 811, 3515, 3529, 3885	<code>\tl_if_blank:NTF</code> . . . . .	3507
<code>\skip_horizontal:n</code> . . . . .	773, 2039, 2047, 2595, 2597, 3513, 3893	<code>\tl_if_empty:NTF</code> . . . . .	506, 525, 553, 570, 590, 597, 623, 640, 1528, 1533, 1555, 1560, 1618, 1682, 1690, 1709, 1811, 1974, 2005, 2119, 2277, 2350, 2396, 2592, 3807, 4085
<code>\skip_if_eq:nnTF</code> . . . . .	952, 958, 964, 1028, 1062, 1103, 1109, 1115, 1146, 1151, 1172, 1223, 1245, 1358, 1372, 1386, 1397, 1408, 1419, 1430, 1441	<code>\tl_if_empty:nTF</code> . . . . .	1583
<code>\skip_new:N</code> . . . . .	69, 70, 74, 75, 76, 77, 78, 133, 173	<code>\tl_if_exist:NTF</code> . . . . .	1588
<code>\skip_set:Nn</code> . . . . .	937, 941, 990, 994, 1031, 1035, 1039, 1046, 1050, 1054, 1065, 1070, 1074, 1080, 1085, 1090, 1148, 1149, 1150, 1157, 1161, 1165, 1174, 1179, 1183, 1186, 1190, 1194, 1225, 1229, 1247, 1251, 1255, 1261, 1265, 1269, 3142, 3156	<code>\tl_if_novalue:nTF</code> . . . . .	2094, 2105, 2274, 2348, 2381, 2475, 2500, 2518, 2523, 2552, 2765, 3119, 3337, 3741, 3805, 4049
<code>\skip_set_eq:NN</code> . . . . .	2675, 2715, 2716, 3532, 3533, 3888, 3889	<code>\tl_map_inline:Nn</code> . . . . .	201, 408
<code>\skip_use:N</code> . . . . .	939, 943, 978, 982, 986, 1006, 1010, 1029, 1048, 1057, 1063, 1068, 1072, 1083, 1087, 1088, 1093, 1129, 1133, 1159, 1359, 1363, 1366, 1373, 1377, 1380, 2909	<code>\tl_new:N</code> . . . . .	34, 39, 40, 43, 44, 50, 52, 53, 54, 56, 57, 90, 91, 92, 98, 99, 100, 102, 103, 104, 105, 106, 108, 109, 115, 116, 126, 127, 138, 139, 145, 146, 147, 150, 167, 170
<code>\skip_zero:N</code> . . . . .	2717, 2888, 3028, 3153, 3154	<code>\tl_put_left:Nn</code> . . . . .	1982, 2015, 2102, 2408, 2442, 3823, 3826
<code>\skip_zero_new:N</code> . . . . .	1141, 1142, 1143, 1220, 1242, 1243, 1244	<code>\tl_put_right:Nn</code> . . . . .	406, 516, 561, 578, 631, 1936, 1943, 1986, 2021, 2104, 2110, 2118, 2121, 2131, 2136, 2139, 2145, 2171, 2181, 2195, 2211, 2217, 2222, 2269, 2272, 2279, 2281, 2308, 2313, 2318, 2321, 2330, 2343, 2346, 2352, 2354, 2364, 2792, 2805, 3357, 3370, 3809, 3810, 3965, 3970, 3975, 3980, 3985
<code>\c_zero_skip</code> . . . . .	952, 958, 964, 1029, 1063, 1103, 1109, 1115, 1146, 1151, 1172, 1223, 1245, 1359, 1373, 1386, 1397, 1408, 1419, 1430, 1441	<code>\tl_remove_all:Nn</code> . . . . .	4084
<code>\small</code> . . . . .	3969, 3974, 3979, 3984, 3989	<code>\tl_remove_once:Nn</code> . . . . .	2159, 2293
<code>\star</code> . . . . .	2455	<code>\tl_replace_all:Nnn</code> . . . . .	410
<code>start</code> . . . . .	675	<code>\tl_reverse:N</code> . . . . .	2158, 2160, 2292, 2294
<code>\stepcounter</code> . . . . .	2477, 3088	<code>\tl_set:Nn</code> . . . . .	45, 282, 290, 298, 375, 449, 453, 458, 459, 505, 550, 622, 770, 784, 796, 808, 1617, 1708, 2250, 2383, 2417, 2430, 2520, 3812, 3835, 4082
str commands:		<code>\tl_set_eq:NN</code> . . . . .	416, 511, 514, 558, 560, 575, 577, 628, 630, 2157, 2291, 2304, 2564, 2568, 3100, 3102
<code>\c_backslash_str</code> . . . . .	4194, 4198, 4207, 4208, 4212, 4213, 4217, 4218, 4249, 4250, 4254, 4259, 4260	<code>\tl_to_str:n</code> . . . . .	1588, 1594, 1599, 3940
<code>\c_colon_str</code> . . . . .	2219, 2323, 3948	<code>\tl_trim_spaces:n</code> . . . . .	406, 4070, 4082, 4088
<code>\str_case:nn</code> . . . . .	223, 278		
<code>\str_case:nnTF</code> . . . . .	1473, 1482		
<code>\str_clear:N</code> . . . . .	2767, 3339		
<code>\str_count:n</code> . . . . .	210, 4180		

<code>\tl_use:N</code> .	412, 415, 527, 592, 599, 642, 841, 845, 849, 853, 857, 861, 865, 869, 873, 877, 881, 885, 889, 893, 897, 901, 2044, 2164, 2172, 2183, 2197, 2202, 2214, 2507, 2513, 2537, 2555, 2559, 2567, 2604, 2605, 2612, 2620, 2621, 2627, 2742, 2959, 3105, 3291, 3519, 3530, 3534, 3693, 3875, 3886, 3892, 3896, 3994, 3995, 3996, 3997, 3998, 4066	<code>\use:n</code> . . . . .	1464, 3946
		<code>\use_none:nn</code> . . . . .	351
		<code>\usecounter</code> . . . . .	2676, 2718
<b>V</b>			
	<code>\value</code> . . . .	1531, 1537, 1544, 1550, 1558, 1564, 1571, 1577	
	<code>\vspace</code>	1020, 1363, 1366, 1377, 1380, 1390, 1392, 1401, 1403, 1412, 1414, 1423, 1425, 1434, 1436, 1445, 1447, 2010, 2018, 3116, 3127, 3570, 3933	
<b>W</b>			
	<code>widest</code> . . . . .		<u>675</u>
	<code>wrap-ans</code> . . . . .		<u>1876</u>
	<code>wrap-label</code> . . . . .		<u>419</u>
	<code>wrap-label*</code> . . . . .		<u>419</u>
	<code>wrap-opt</code> . . . . .		<u>1876</u>
<b>U</b>			
<code>\u</code> . . . . .		204	
use commands:			
	<code>\use:N</code> . . . . .	211, 2609, 2744	
<code>\tl_use:N</code> .	412, 415, 527, 592, 599, 642, 841, 845, 849, 853, 857, 861, 865, 869, 873, 877, 881, 885, 889, 893, 897, 901, 2044, 2164, 2172, 2183, 2197, 2202, 2214, 2507, 2513, 2537, 2555, 2559, 2567, 2604, 2605, 2612, 2620, 2621, 2627, 2742, 2959, 3105, 3291, 3519, 3530, 3534, 3693, 3875, 3886, 3892, 3896, 3994, 3995, 3996, 3997, 3998, 4066		
token commands:			
	<code>\token_to_str:N</code> . . . . .	359	
	<code>\topsep</code> . . . . .	2011, 2019	
	<code>topsep</code> . . . . .	<u>695</u>	
	<code>\typeout</code> .	232, 242, 286, 294, 302, 327, 330, 340, 341, 1783, 1802	