

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-04-22<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 Introduction</b> . . . . .	<b>2</b>	<b>4 The storage system</b> . . . . .	<b>9</b>
1.1 Description and usage . . . . .	3	4.1 Keys for storage . . . . .	10
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 check answers . . . . .	10
1.3.1 Internal counters . . . . .	3	4.4 The command <code>\anskey</code> . . . . .	10
1.3.2 Support for multicol . . . . .	4	4.5 The environment keyans . . . . .	11
1.3.3 Support for minipage . . . . .	4	4.5.1 The <code>\item*</code> in keyans . . . . .	11
1.3.4 The <code>\label</code> and <code>\ref</code> system . . . . .	4	4.6 The environment keyanspic . . . . .	12
1.3.5 Support for <code>\footnote</code> . . . . .	4	4.6.1 The command <code>\anspic</code> . . . . .	12
<b>2 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 The command <code>\setenumext</code></b> . . . . .	<b>5</b>	<b>5 Full examples</b> . . . . .	<b>14</b>
3.1 Keys for label and ref . . . . .	6	<b>6 The way of non-enumerated lists</b> . . . . .	<b>16</b>
3.2 Keys for spaces . . . . .	6	<b>7 References</b> . . . . .	<b>18</b>
3.2.1 Vertical spaces . . . . .	7	<b>8 Change history</b> . . . . .	<b>18</b>
3.2.2 Horizontal spaces . . . . .	8	<b>9 Index of Documentation</b> . . . . .	<b>19</b>
3.3 Keys for add code . . . . .	8	<b>10 Implementation</b> . . . . .	<b>21</b>
3.4 Keys for start and resume . . . . .	9	<b>11 Index of Implementation</b> . . . . .	<b>101</b>
3.5 Keys for multicol . . . . .	9		
3.6 Keys for minipage . . . . .	9		
3.6.1 The command <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  $\text{\TeX}$  team for their great work and to the different members of the **TeX-SX** community who have provided great answers and ideas. Here a note of the main ones:

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

## License and Requirements

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

The **enumext** package loads and requires **multicol**[3] package, need to have a modern  $\text{\TeX}$  distribution such as  $\text{\TeX}$  Live or Mi $\text{\TeX}$ . It has been tested with the standard classes provided by  $\text{\TeX}$ : **book**, **report**, **article** and **letter** on 10pt, 11pt and 12pt.

<sup>\*</sup>This file describes a documentation for v1.0, last revised 2024-04-22.

<sup>†</sup>E-mail: [pablgonz@educarchile.cl](mailto:pablgonz@educarchile.cl).

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. (a)  $(x - 1)^2$ 

\* ii. Yes, dnf

\*

(b)  $3(x + y + z)$ 

\* iii. A. doesn’t exist

(c) i. False

\* for now :(

ii. Very True!

\* B. very good

(d) i. Yes

\* C. obsolete
- Another very common thing to use in my work is “multiple choice questions”, for example:
1. First type of questions

4. Question with image and label below:
- (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
- 
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. (a) (B)  $x = 5$ 

\* (e) (C) some note

\*

(b)

(f) (B)

(c) (D)

(g) (D) “other note”

(d)
- These “simple worksheets” or “multiple choice questions” appear to be easy to obtain using a combination of the `enumerate`, `minipage` and `multicols` environments, but like many things, what “looks simple” is not so simple.
- The `enumext` package was created and designed to meet these small requirements in the creation of “simple worksheets” and “multiple choice questions”.
- ©2024 by Pablo González L
- 2 / 112

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

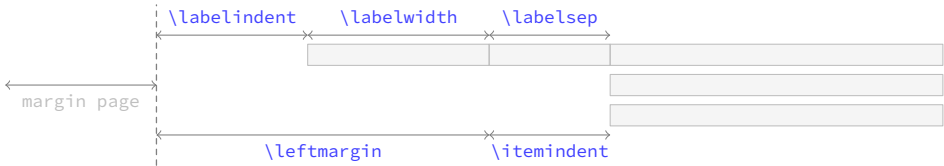


Figure 1: Representation of horizontal lengths in `enumitem`.

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

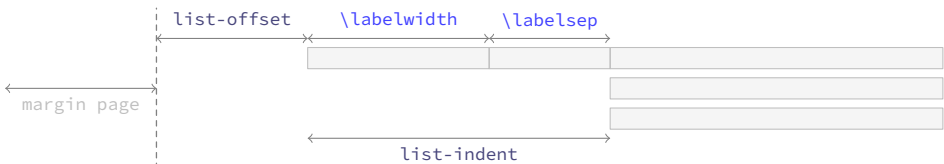


Figure 2: Representation of horizontal lengths concept in `enumext`.

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

### 1.3.2 Support for multicols

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

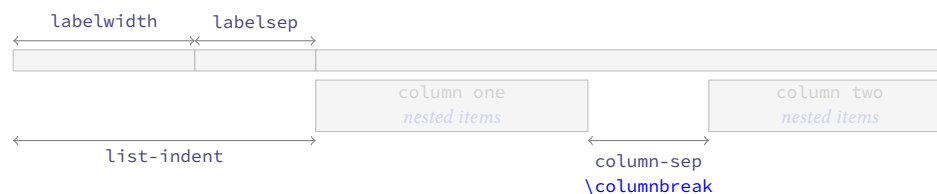


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

### 1.3.3 Support for minipage

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



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

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

### 1.3.4 The \label and \ref system

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

- The implementation of `\label` and `\ref` together with the `store-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  $\langle key \rangle$  that sets a *rubber or rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and *keyans* and *keyanspic* environments. All  $\langle keys \rangle$  related to vertical or horizontal spacing accept a “*skip*” or “*dim*” expression if passed between braces, i.e. you do not need to use `\dimexpr` or `\dimeval` to perform calculations.

### 3.1 Keys for label and ref

`label = { $\langle \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \rangle$ }` default: *by levels*

Sets the  $\langle label \rangle$  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  $\langle label \rangle$  will be displayed, and the and the form in which it is used by standard “*label and ref*” and the “*internal reference*” system with the `store-ref` key. You cannot use commands with  $\langle label \rangle$  as an argument, for example `\emph{\langle \backslash alph* \rangle}` will return an error. For full customization of how  $\langle label \rangle$  is displayed use the `font` or `wrap-label` keys.

`ref = { $\langle code \{ \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \} more code \rangle$ }` 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{\langle \backslash alph* \rangle}` 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 = { $\langle rigid length \rangle$ }` default: `0.3333em`

Sets the *horizontal space* between the box containing the current  $\langle label \rangle$  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 = { $\langle rigid length \rangle$ }` default: *by label*

Sets the *width* of the box containing the current  $\langle label \rangle$  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 = { $\langle integer | string \rangle$ }` default: *empty*

Sets the `labelwidth` key pass the  $\langle integer \rangle$  or converting the  $\langle string \rangle$  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 = { $\langle font commands \rangle$ }` default: *empty*

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

`align = { $\langle left | right | center \rangle$ }` default: *left*

Sets the *aligned* of  $\langle label \rangle$  defined by `label` key on the current level in the label box.

`wrap-label = { $\langle code \{ \#1 \} more code \rangle$ }` default: *empty*

Wraps the current  $\langle label \rangle$  defined by `label` key referenced by  $\{ \#1 \}$ . The  $\{ \langle code \rangle \}$  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* = { $\langle code \{ \#1 \} more code \rangle$ }` default: *empty*

The same as the `wrap-label` key but also applies on `\item[ $\langle custom \rangle$ ]`.

### 3.2 Keys for spaces

`show-length = { $\langle true | false \rangle$ }` 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  $\text{\LaTeX}$  does *not discard* this space at page break.

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

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

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

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

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

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

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

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

### 3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

`list-indent` = {*<rigid length>*} default: `labelwidth + labelsep`

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

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

## 3.3 Keys for add code

- The following *<keys>* should be used with “*caution*”, they are intended to inject *{<code>}* into different parts of the defined environments. We must keep in mind that the defined environments are based on the `list` base environment provided by  $\text{\LaTeX}$  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>}* 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>}`. The *{<code>}* must be passed between braces.

`before*` = {*<code>*} default: *not used*



Execute `{\code}` “before” the environment starts. The `{\code}` 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}`. The `{\code}` must be passed between braces.

`first = {\code}` default: *not used*  
 Executes `{\code}` when “starting” the environment. The `{\code}` must be passed between braces, is executed right “after” all *list parameters* are done, after the second argument of `list`, just before the first occurrence of `\item`: `\list{\arg one}{\arg two}{\code}\item`.

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

`after = {\code}` default: *not used*  
 Execute `{\code}` “after” finishing the environment. The `{\code}` must be passed between braces.

### 3.4 Keys for start and resume

`start = {\integer | string}` default: `1`  
 Sets the *start value* of the numbering on the current level. Internally `\string` is passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

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

### 3.5 Keys for multicol

`columns = {\integer}` default: `1`  
 Set the *number of columns* to be used by the `multicol` environment within the environment. The value must be a positive integer less than or equal to `10`.

`columns-sep = {\rigid length}` default: *by level*  
 Set the *space between columns* used by the `multicol` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

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

### 3.6 Keys for minipage

`mini-env = {\rigid length}` default: *not used*  
 Sets the *width* of the `minipage` environment on the “right side”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “left side”, taking `\linewidth` as the maximum reference value.

`mini-sep = {\rigid length}` default: `0.3333em`  
 Sets the *space between* the `minipage` environment on the “left side” and the `minipage` environment on the “right side”. This separation is applied together with `\hfill`.

#### 3.6.1 The command `\miniright`

---

`\miniright`    The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage`  
`\miniright*` 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.

## 4 The storage system

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

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

4.1 Keys for storage

- save-ans = { <store name> }

default: not set

Sets the “name” of the <sequence> and <prop list> in which the contents will be “stored” by \anskey in enumext environment, \item\* in keyans environment and \anspic\* in keyanspic environment. If the <sequence> or <prop list> does not exist, it will be created globally.
- wrap-ans = { <code {#1} more code> }

default: \fbox

Wraps the current <argument> passed \anskey command to referenced by {#1}. The {<code>} must be passed between braces. This <key> 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}’.
- 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.
- 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 store-ref key is active.

4.2 Keys for internal label and ref

- store-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 store-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 check answers

- 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 store-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

- ★ 1. Text containing our instructions or questions.

\* 

first answer

2. Text containing our instructions or questions.

(a) Question.

\* 

second answer
3. Text containing our instructions or questions.

\* 

third answer

4. Text containing our instructions or questions.

\* 

fourth answer

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

## 4.5 The environment keyans

---

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

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

---

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

---

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

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

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

### Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans]
  \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*[\langle note \rangle] 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.

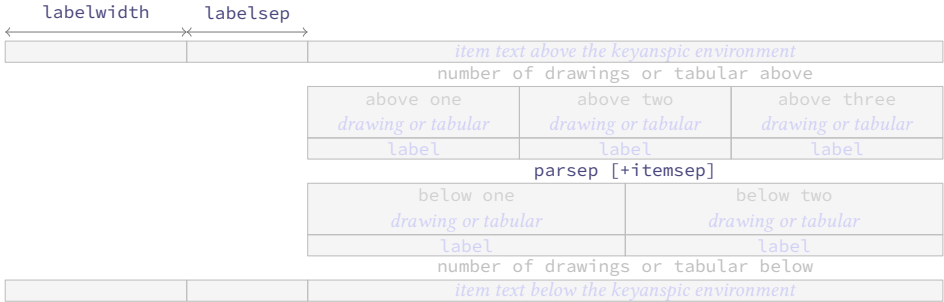


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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]

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 `store-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,nosep,store-ref]
  \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. (a)  $3(x + y + z)$

(b) i. Very True!

(c) i. Yes

ii. A. very good

B. 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. (a) B

(b) A
- (c) B

(d) 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. (a) B

(b) A

(c) B

(d) 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

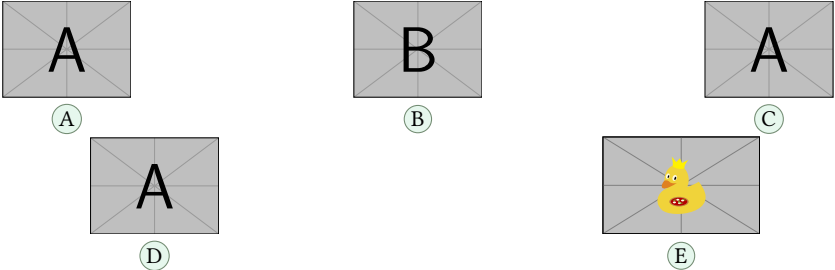
C I and II only
3. Third type of questions
- (1)  $2\alpha + 2\delta = 90^\circ$

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

A value

B value

C value
4. Question with image and label below:



- D I and III only
- E I, II, and III

- D value
- E value

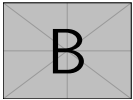
5. Question with image on left side:
- A value

B value

C value

D correct


E value



Test keys

1. (a) B  $x = 5$
- (b)
- (c) D
- (d)
- (e) C some note
- (f) B
- (g) D other note

Example 4

A “simple worksheet” using ducks :) .

- ca. 1.

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

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

True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?
- ca. 4.

Related to Linux
- (a)

You use linux?
- (b)

Usually uses the package manager?
- (c)

Rate the following package and class
- i.

`xsim-exam`
- ii.

`xsim`
- iii.

`exsheets`

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

2

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

3

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

4

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

1. (a) A)            (c) B)  
      (b) C)        (d) 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  $\TeX$  for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` 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 `nosep` 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,

**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 2024-04-22** – First public release.

9 Index of Documentation

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

C

Document class:

article . . . . . 1

book . . . . . 1

exam . . . . . 2

letter . . . . . 1

report . . . . . 1

\columnbreak . . . . . 4

\columnsep . . . . . 9

Commands provide by enumext:

\anskey . . . . . 3, 9–11

\anspic\* . . . . . 3, 10, 12

\anspic . . . . . 12

\getkeyans . . . . . 3, 10, 13

\item\* . . . . . 3–6, 10, 11

\item . . . . . 5, 6, 9–11

\miniright . . . . . 3, 4, 9

\printkeyans . . . . . 3, 5, 10, 13

\setenumext . . . . . 3, 5, 6, 10, 11, 13

Counters defined by enumext:

enumXiii . . . . . 3

enumXii . . . . . 3

enumXiv . . . . . 3

enumXi . . . . . 3

enumXviii . . . . . 3

enumXvii . . . . . 3

enumXvi . . . . . 3

enumXv . . . . . 3

E

Environments provide by enumext:

enumext\* . . . . . 3, 4

enumext . . . . . 3–5, 9–11, 13, 16

keyans\* . . . . . 3, 4

keyanspic . . . . . 3, 6, 9, 10, 12, 16

keyans . . . . . 3–7, 9–12, 16

Environments:

enumerate . . . . . 1–3, 5, 18

list . . . . . 3, 8, 18

minipage . . . . . 2–4, 9, 18

multicols . . . . . 2, 4, 9

I

\item . . . . . 3, 4

\itemsep . . . . . 8

K

Keys for environments provide by enumext:

above\* . . . . . 8

above . . . . . 8

after . . . . . 9

align . . . . . 6, 17

before\* . . . . . 8

before . . . . . 8

below\* . . . . . 8

below . . . . . 8

check-ans . . . . . 10

columns-sep . . . . . 4, 9

columns . . . . . 4, 8, 9

first . . . . . 9

font . . . . . 6

item-pos\* . . . . . 5

item-sym\* . . . . . 5

itemindent . . . . . 8

itemsep . . . . . 8, 12

labelsep . . . . . 3, 5, 6, 8–10, 17

labelwidth . . . . . 3, 5, 6, 8–10, 17

label . . . . . 6, 9, 11, 13, 16, 17

list-indent . . . . . 3, 8

list-offset . . . . . 3, 8, 17

listparindent . . . . . 8

mark-ans . . . . . 10

mark-pos . . . . . 10

mark-ref . . . . . 10

mini-env . . . . . 4, 8, 9

mini-sep . . . . . 4, 9

no-store . . . . . 10

noitemsep . . . . . 8

nosep . . . . . 8, 16

parsep . . . . . 7, 8, 12

partopsep . . . . . 7

ref . . . . . 4, 6

resume . . . . . 9

rightmargin . . . . . 8

save-ans . . . . . 4, 9–13

show-ans . . . . . 10

show-length . . . . . 6

show-pos . . . . . 10, 13

start . . . . . 9

store-ref . . . . . 4, 6, 10, 13

topsep . . . . . 7, 8

widest . . . . . 6

wrap-ans . . . . . 10

wrap-label\* . . . . . 6, 17

wrap-label . . . . . 6

L

\label . . . . . 4

Labels provide by enumext:

\Alph\* . . . . . 6, 11

\Roman\* . . . . . 6

\alph\* . . . . . 6

\arabic\* . . . . . 6

\roman\* . . . . . 6

\labelsep . . . . . 3, 6

\labelwidth . . . . . 3, 6

\linewidth . . . . . 9

\listparindent . . . . . 8

P

Packages:

enumerate . . . . . 17

enumext . . . . . 1–3, 12, 17, 18

enumitem . . . . . 3, 4, 8, 17, 18

footnotehyper . . . . . 4

hyperref . . . . . 4, 10, 18

l3prop . . . . . 1, 18

l3seq . . . . . 1, 18

multicol . . . . . 1, 4, 18

xsim . . . . . 2

\parsep . . . . . 7

\partopsep . . . . . 7

©2024 by Pablo González L

19 / 112

<b>R</b>		
\raggedcolumns . . . . .	4	\rightmargin . . . . . 8
		<b>T</b>
\ref . . . . .	4	\topsep . . . . . 7



## 10 Implementation

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

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

### 10.1 General conventions

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

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

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

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

### 10.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

Identify the internal prefix (L<sup>A</sup>T<sub>E</sub>X3 DocStrip convention) for l3doc class.

```
2 (@@=enumext)
```

### 10.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L<sup>A</sup>T<sub>E</sub>X to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-04-22}
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 \l__enumext_level_int
21 \int_new:N \l__enumext_level_h_int
22 \int_new:N \l__enumext_keyans_level_int
23 \int_new:N \l__enumext_keyans_level_h_int
24 \int_new:N \l__enumext_keyans_pic_level_int
25 \bool_new:N \l__enumext_starred_bool
26 \bool_new:N \g__enumext_starred_bool
```

```

27 \bool_new:N \l__enumext_standar_bool
28 \bool_new:N \g__enumext_standar_bool
29 \bool_new:N \l__enumext_keyans_env_bool

```

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

```

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

```

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

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

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

```

30 \cs_set_protected:Npn \__enumext_tmp:n #1
31 {
32   \tl_new:c { l__enumext_counter_#1_tl }
33 }
34 \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.)

```

\l__enumext_resume_bool
\g__enumext_resume_int
\l__enumext_resume_vii_bool
\g__enumext_resume_vii_int
\g__enumext_item_symbol_tl

```

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

```

35 \bool_new:N \l__enumext_resume_bool
36 \int_new:N \g__enumext_resume_int
37 \bool_new:N \l__enumext_resume_vii_bool
38 \int_new:N \g__enumext_resume_vii_int
39 \tl_new:N \g__enumext_item_symbol_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

40 \dim_new:N \l__enumext_current_widest_dim
41 \tl_new:N \g__enumext_counter_styles_tl
42 \tl_new:N \g__enumext_widest_label_tl
43 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

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

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

```

44 \cs_set_protected:Npn \__enumext_tmp:n #1
45 {
46   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
47   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
48   \dim_new:c { l__enumext_leftmargin_#1_dim }
49   \dim_new:c { l__enumext_itemindent_#1_dim }
50 }
51 \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).

```

52 \cs_set_protected:Npn \__enumext_tmp:n #1
53 {
54   \skip_new:c { l__enumext_multicols_above_#1_skip }
55   \skip_new:c { l__enumext_multicols_below_#1_skip }
56 }
57 \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).

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

```
70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \bool_new:c { \l__enumext_wrap_label_#1_bool }
73   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
74   \int_new:c { \l__enumext_start_#1_int }
75   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
76   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
77   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
78   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
79   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
80 }
81 \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_keyans_tmpa_tl
```

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

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

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.28.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§10.34.1) for the `keyanspic` environment.

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

```
82 \bool_new:N \l__enumext_store_active_bool
83 \tl_new:N \l__enumext_store_name_tl
84 \tl_new:N \g__enumext_store_name_tl
85 \tl_new:N \l__enumext_store_anskey_arg_tl
86 \int_new:N \l__enumext_store_columns_join_int
87 \tl_new:N \l__enumext_store_keyans_label_tl
88 \tl_new:N \l__enumext_keyans_tmpa_tl
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpp_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpp_seq
```

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

```
89 \tl_new:N \l__enumext_setkey_tmpa_tl
90 \tl_new:N \l__enumext_setkey_tmpp_tl
91 \int_new:N \l__enumext_setkey_tmpa_int
92 \seq_new:N \l__enumext_setkey_tmpa_seq
93 \seq_new:N \l__enumext_setkey_tmpp_seq
```

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

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

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

```
94 \cs_set_protected:Npn \l__enumext_tmp:n #1
95 {
96   \tl_new:c { \l__enumext_store_opt_#1_tl }
97   \tl_new:c { \l__enumext_print_keyans_#1_tl }
98   \bool_new:c { \l__enumext_store_columns_#1_bool }
99   \int_new:c { \l__enumext_store_columns_#1_int }
100   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
101   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
102   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
103 }
104 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

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

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

Internal variables for “storage system” mechanism used by `\anskey` (§10.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.23).

```
105 \bool_new:N \l__enumext_show_answer_bool
106 \bool_new:N \l__enumext_show_position_bool
107 \tl_new:N \l__enumext_mark_ref_sym_tl
108 \tl_new:N \l__enumext_mark_answer_sym_tl
109 \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).

```
110 \seq_new:N \l__enumext_keyans_pic_body_seq
111 \dim_new:N \l__enumext_keyans_pic_width_dim
112 \int_new:N \l__enumext_keyans_pic_above_int
113 \int_new:N \l__enumext_keyans_pic_below_int
114 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

```
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\l__enumext_compare_items_ans_int
\g__enumext_count_item_with_ans_int
\g__enumext_count_item_all_int
\g__enumext_count_level_X_int
\g__enumext_count_item_X_int
```

Internal variables used by “check answer” mechanism (§10.22.1) controlled by the `check-ans` and `no-store` keys.

```
115 \bool_new:N \l__enumext_store_ans_bool
116 \bool_new:N \l__enumext_check_ans_bool
117 \bool_new:N \g__enumext_check_ans_show_bool
118 \bool_new:N \g__enumext_check_ans_show_h_bool
119 \tl_new:N \g__enumext_check_ans_item_tl
120 \int_new:N \l__enumext_compare_items_ans_int
121 \int_new:N \g__enumext_count_item_with_ans_int
122 \int_new:N \g__enumext_count_item_all_int
123 \cs_set_protected:Npn \l__enumext_tmp:n #1
124 {
125   \int_new:c { \g__enumext_count_level_#1_int }
126   \int_new:c { \g__enumext_count_item_#1_int }
127 }
128 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_check_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`.

```
129 \bool_new:N \l__enumext_hyperref_bool
130 \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 `store-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.

```

131 \tl_new:N \l__enumext_newlabel_arg_one_tl
132 \tl_new:N \l__enumext_newlabel_arg_two_tl
133 \tl_new:N \l__enumext_store_write_aux_file_tl
134 \cs_set_protected:Npn \__enumext_tmp:n #1
135 {
136   \tl_new:c { l__enumext_label_copy_#1_tl }
137 }
138 \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`.

```

139 \int_new:N \g__enumext_footnote_int
140 \seq_new:N \g__enumext_footnote_arg_seq
141 \seq_new:N \g__enumext_footnote_int_seq

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_aux_tl
\l__enumext_the_counter_X_tl
\l__enumext_counter_style_for_ref_X_tl

```

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

```

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

```

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

```

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

```

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

```

153 \cs_set_protected:Npn \__enumext_tmp:n #1
154 {
155   \bool_new:c { l__enumext_item_starred_#1_bool }
156   \int_new:c { l__enumext_item_column_pos_#1_int }
157   \int_new:c { g__enumext_item_count_all_#1_int }
158   \int_new:c { l__enumext_joined_item_#1_int }
159   \int_new:c { l__enumext_joined_item_aux_#1_int }
160   \int_new:c { l__enumext_tmpa_#1_int }
161   \box_new:c { l__enumext_item_text_#1_box }
162   \dim_new:c { l__enumext_joined_width_#1_dim }
163   \dim_new:c { l__enumext_item_width_#1_dim }
164   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
165   \str_new:c { l__enumext_align_label_#1_str }
166   \bool_new:c { g__enumext_minipage_active_#1_bool }
167   \tl_new:c { g__enumext_miniright_code_#1_tl }
168   \bool_new:c { g__enumext_minipage_center_#1_bool }
169   \dim_new:c { g__enumext_minipage_right_#1_dim }
170   \skip_new:c { g__enumext_minipage_right_#1_skip }
171 }
172 \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`.

```

173 \clist_const:Nn \c__enumext_all_envs_clist
174 {
175   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
176   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
177 }

```

(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.

```
178 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
179 {
180   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
181 }
```

(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.

```
182 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
183 {
184   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
185 }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
186 \cs_new:Nn \__enumext_level:
187 {
188   \int_to_roman:n { \__enumext_level_int }
189 }
```

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

`\__enumext_level_set:n`

Function for set level in `enumext*`, `keyans*` and `keyans`.

`\__enumext_level_end:n`

```
190 \cs_new:Npn \__enumext_level_set:n #1
191 {
192   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
193   \cs_set:Nn \__enumext_level: { #1 }
194 }
195 \cs_new:Npn \__enumext_level_end:n #1
196 {
197   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
198 }
```

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

`\__enumext_if_is_int:nT`

`\__enumext_if_is_int:nF`

`\__enumext_if_is_int:nTF`

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

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

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

`\__enumext_show_length:nnn`

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

```
205 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
206 {
207   * ~ #2
208   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
209   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
210 }
```

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



## 10.6 Copying list and minipage environments

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

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

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

```
\__enumext_start_list:nn
  \__enumext_stop_list:
  \__enumext_item_std:w
```

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

```
211 \__enumext_at_begin_document:n
212 {
213   \cs_new_eq:NN \__enumext_start_list:nn \list
214   \cs_new_eq:NN \__enumext_stop_list: \endlist
215   \cs_new_eq:NN \__enumext_item_std:w \item
216 }
```

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

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

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

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

```
\__enumext_minipage:w
\__enumext_endminipage:
```

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

```
217 \__enumext_at_begin_document:n
218 {
219   \cs_new_eq:NN \__enumext_minipage:w \minipage
220   \cs_new_eq:NN \__enumext_endminipage: \endminipage
221 }
```

(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.

```
222 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
223 \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”.

```
224 \cs_new_protected:Nn \__enumext_after_hyperref:
225 {
226   \IfPackageLoadedTF { hyperref }
227   {
228     \msg_info:nnn { enumext } { package-load } { hyperref }
229     \bool_set_true:N \l__enumext_hyperref_bool
230     \IfHyperBoolean{hyperfootnotes}
231     {
232       \typeout{hyperfootnotes=true}
233       \bool_set_true:N \l__enumext_footnotes_key_bool
234     }
235     { \typeout{hyperfootnotes=false} }
236   }
237   { }
```

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`.

```
238   \bool_if:NT \l__enumext_footnotes_key_bool
239   {
240     \IfPackageLoadedTF { footnotehyper }
241     {
```

```

242         \msg_info:nnn { enumext } { package-load } { footnotehyper }
243     }
244     {
245         \typeout{No ~ footnotehyper ~ load}
246         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
247         \bool_set_false:N \l__enumext_footnotes_key_bool
248     }
249 }

```

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.

```

250     \bool_if:NTF \l__enumext_hyperref_bool
251     {
252         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
253         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
254     }
255     {
256         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
257         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
258     }
259 }

```

(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 `store-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.

```

260 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
261 {
262     \protected@write \@auxout { }
263     {
264         \token_to_str:N \newlabel {#1}
265         {
266             {#2}
267             \bool_if:NT \l__enumext_hyperref_bool
268             { { \thepage } {#2} {#1} }
269             { }
270         }
271     }
272     \__enumext_hypertarget:nn {#1} { }
273     \__enumext_phantomsection:
274 }

```

(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.

```

275 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
276 {
277     \cs_if_exist:cTF { c@ #2 }
278     { \msg_fatal:nnn { enumext } { counters } { #2 } }
279     {
280         \tl_set:Nn #1 { #2 }
281         \newcounter { #2 }
282     }
283 }

```

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

```

enumXi    The counters created here are enumXi, enumXii, enumXiii and enumXiv for enumext environment,
enumXii   enumXv for keyans environment, enumXvi for keyanspic environment, enumXvii for enumext* and
enumXiii  enumXviii for the keyans* environments.
enumXiv   284 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv    285 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi   286 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii  287 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 288 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
          289 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
          290 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
          291 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

(End of definition for enumXi and others.)

## 10.9 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```

292 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
293 {
294   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
295   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
296 }
297 \__enumext_register_counter_style:Nn \arabic { 0 }
298 \__enumext_register_counter_style:Nn \Alph { M }
299 \__enumext_register_counter_style:Nn \alph { m }
300 \__enumext_register_counter_style:Nn \Roman { VIII }
301 \__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.

```

302 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
303 {
304   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
305   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
306 }
307 \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.

```

308 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
309 {
310   \tl_clear_new:N #1
311   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
312   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
313   \tl_map_inline:Nn \g__enumext_counter_styles_tl
314   {
315     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
316     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
317     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
318   }
319   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
320   { \tl_use:N \g__enumext_widest_label_tl }
321   \tl_set_eq:cN { the #2 } #1
322 }
323 \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 Definition of keys `font`, `labelsep`, `labelwidth`, `wrap-label` and `wrap-label*` keys for `enumext` and `keyans` environments.

```

324 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
325 {
326   \keys_define:nn { enumext / #1 }
327   {
328     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
329     font .value_required:n = true,
330     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
331     labelsep .initial:n = {0.3333em},
332     labelsep .value_required:n = true,
333     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
334     labelwidth .value_required:n = true,
335     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
336     wrap-label .initial:n = {##1},
337     wrap-label .value_required:n = true,
338     wrap-label* .code:n = {
339       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
340       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
341     },
342     wrap-label* .value_required:n = true,
343   }
344 }
345 \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.

```

346 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
347 {
348   \keys_define:nn { enumext / #1 }
349   {
350     align .choice:,
351     align / left .code:n =
352       {
353         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
354         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
355       },
356     align / right .code:n =
357       {
358         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
359         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
360       },
361     align / center .code:n =
362       {
363         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
364         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
365       },
366     align .initial:n = left,
367     align .value_required:n = true,
368   }
369 }
370 \clist_map_inline:nn
371 {
372   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
373 }
374 { \__enumext_tmp:nn #1 }

```

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

```

375 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
376 {
377   \keys_define:nn { enumext / #1 }
378   {
379     align .choice:,
380     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
381     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },

```

```

382     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
383     align .initial:n = left,
384     align .value_required:n = true,
385   }
386 }
387 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

## 10.11 Setting label and ref keys

`\__enumext_regex_label_ref_key:`

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

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

```

388 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
389 {
390   \tl_map_inline:Nn \c__enumext_counter_style_tl
391   {
392     \regex_replace_once:nnN { \c{##1}\* }
393     { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
394   }
395 }

```

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

`\__enumext_set_label_ref:n`

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

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

```

396 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
397 {
398   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
399   \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
400   \__enumext_regex_label_ref_key:
401   \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
402   \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
403   {
404     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
405     { \exp_not:V \l__enumext_ref_key_arg_tl }
406   }
407 }

```

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

`\__enumext_use_key_ref:`

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

```

408 \cs_new_protected:Nn \__enumext_use_key_ref:
409 {
410   \tl_if_empty:cF { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
411   {
412     \tl_use:c { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
413   }
414 }

```

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

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

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

`\__enumext_set_label_ref_h:n`

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

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

```

415 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
416 {

```

```

417 \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
418 \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
419 {
420   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
421   \__enumext_regex_label_ref_key:
422   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
423   \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
424   {
425     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
426     { \exp_not:V \l__enumext_ref_key_arg_tl }
427   }
428 }
429 {
430   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
431   \__enumext_regex_label_ref_key:
432   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
433   \tl_put_right:Ne \l__enumext_counter_style_for_ref_viii_tl
434   {
435     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
436     { \exp_not:V \l__enumext_ref_key_arg_tl }
437   }
438 }
439 }

```

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

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

```

440 \cs_new_protected:Nn \__enumext_use_key_ref_h:
441 {
442   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
443   {
444     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
445     {
446       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
447     }
448   }
449   {
450     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
451     {
452       \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
453     }
454   }
455 }

```

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

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

`label` Here we set the default `<labels>` of the four levels of `enumext` environment, along with the default value for `labelwidth` key.

```

\__enumext_label_i_tl
\__enumext_label_ii_tl
\__enumext_label_iii_tl
\__enumext_label_iv_tl
456 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
457 {
458   \keys_define:nn { enumext / #1 }
459   {
460     label .code:n = {
461       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
462       { l__enumext_counter_#2_tl } {##1}
463       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
464       \l__enumext_current_widest_dim
465     },
466     label .initial:n = #3,
467     label .value_required:n = true,
468     ref .code:n = \__enumext_set_label_ref:n {##1},
469     ref .value_required:n = true,
470   }
471 }
472 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
473 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
474 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
475 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

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

```

\l__enumext_label_vii_tl 476 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 477 {
478   \keys_define:nn { enumext / #1 }
479   {
480     label .code:n = {
481       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
482       { \l__enumext_counter_#2_tl } {##1}
483       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
484       \l__enumext_current_widest_dim
485     },
486     label .initial:n = #3,
487     label .value_required:n = true,
488     ref .code:n = \__enumext_set_label_ref_h:n {##1},
489     ref .value_required:n = true,
490   }
491 }
492 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
493 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }
```

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

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

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

```

494 \keys_define:nn { enumext / keyans }
495 {
496   label .code:n = {
497     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
498     { \l__enumext_counter_v_tl } {##1}
499     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
500     \l__enumext_current_widest_dim
501     \__enumext_label_style:cvn { \l__enumext_label_vii_tl }
502     { \l__enumext_counter_vii_tl } {##1}
503     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
504     \l__enumext_current_widest_dim
505   },
506   label .initial:n = (\Alph*),
507   label .value_required:n = true,
508 }
```

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

## 10.12 Setting start and widest keys

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

$\l__enumext_start_from:ccn$  #1:  $\l__enumext_label_X_tl$   
 #2:  $\l__enumext_start_X_int$   
 #3:  $\langle integer \text{ or } string \rangle$

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  $\langle integer \rangle$  or  $\langle string \rangle$  of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start=A` or `start=1` to be used.

```

509 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
510 {
511   \__enumext_if_is_int:nTF { #3 }
512   {
513     \int_set:Nn #2 {##3}
514   }
515   {
516     \regex_match:nVT { \c{Alph} | \c{alph} } {##1}
517     { \int_set:Nn #2 { \int_from_alph:n {##3} } }
518     \regex_match:nVT { \c{Roman} | \c{roman} } {##1}
519     { \int_set:Nn #2 { \int_from_roman:n {##3} } }
520   }
521 }
522 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }
```



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

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

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

- #1: The counter associated with the environment level
- #2: `\l__enumext_label_X_tl`
- #3: `\l__enumext_labelwidth_X_dim`
- #4:  $\langle integer \text{ or } string \rangle$

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

```
523 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
524 {
525   \__enumext_if_is_int:nTF {#4}
526   {
527     \setcounter{enumX#1} { #4 }
528   }
529   {
530     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
531     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
532     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
533     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
534   }
535   \__enumext_label_width_by_box:cv
536   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
537 }
538 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

```
start
widest
```

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

```
\l__enumext_start_X_int
```

```
539 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
540 {
541   \keys_define:nn { enumext / #1 }
542   {
543     start .code:n = {
544       \__enumext_start_from:ccn
545       { \l__enumext_label_#2_tl }
546       { \l__enumext_start_#2_int } {##1}
547     },
548     start .initial:n = 1,
549     widest .code:n = {
550       \__enumext_widest_from:nccn {#2}
551       { \l__enumext_label_#2_tl }
552       { \l__enumext_labelwidth_#2_dim } {##1}
553     },
554     widest .value_required:n = true,
555     start .value_required:n = true,
556   }
557 }
558 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

### 10.13 Setting keys for vertical spaces

```
topsep
partopsep
parsep
noitemsep
nosep
```

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

```
559 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
560 {
561   \keys_define:nn { enumext / #1 }
562   {
563     topsep .skip_set:c = { \l__enumext_topsep_#2_skip },
564     topsep .initial:n = {#3},
565     topsep .value_required:n = true,
566     partopsep .skip_set:c = { \l__enumext_partopsep_#2_skip },
567     partopsep .initial:n = {#4},
568     partopsep .value_required:n = true,
569     parsep .skip_set:c = { \l__enumext_parsep_#2_skip },
570     parsep .initial:n = {#5},
```

```

571     parsep      .value_required:n = true,
572     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
573     itemsep     .initial:n = {#6},
574     itemsep     .value_required:n = true,
575     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
576     noitemsep   .value_forbidden:n = true,
577     nosepe     .meta:n = {
578                         itemsep = 0pt, parsep= 0pt,
579                         topsep = 0pt, partopsep = 0pt,
580                     },
581     nosepe     .value_forbidden:n = true,
582 }
583 }

```

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

```

584 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
585 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
586 { 4.0pt plus 2.0pt minus 1.0pt }
587 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
588 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
589 { 2.0pt plus 1.0pt minus 1.0pt }
590 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
591 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
592 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
593 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
594 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
595 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
596 { 2.0pt plus 1.0pt minus 1.0pt }
597 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
598 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
599 { 4.0pt plus 2.0pt minus 1.0pt }
600 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
601 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
602 { 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.

```

603 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
604 {
605     \keys_define:nn { enumext / #1 }
606     {
607         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
608         itemindent .value_required:n = true,
609         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
610         rightmargin .value_required:n = true,
611         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
612         listparindent .value_required:n = true,
613         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
614         list-offset .value_required:n = true,
615         list-indent .code:n =
616             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
617             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
618         list-indent .value_required:n = true,
619     }
620 }
621 \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.

```

622 \cs_set_protected:Npn \__enumext_tmp:n #1
623 {
624     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
625 }
626 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

### 10.14.1 Functions for setting the fake itemindent

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

```

627 \cs_set_protected:Nn \__enumext_fake_item:
628 {
629   \dim_compare:nNnT
630     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
631     >
632     { \c_zero_dim }
633   {
634     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
635     {
636       \exp_not:N \mode_leave_vertical:
637       \exp_not:n { \skip_horizontal:n }
638       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
639       \ignorespaces
640     }
641   }
642 }
643 \cs_set_protected:Nn \__enumext_keyans_fake_item:
644 {
645   \dim_compare:nNnT
646     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
647   {
648     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
649     {
650       \exp_not:N \mode_leave_vertical:
651       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
652     }
653   }
654 }
655 \cs_set_protected:Nn \__enumext_fake_item_vii:
656 {
657   \dim_compare:nNnT
658     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
659   {
660     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
661     {
662       \exp_not:N \mode_leave_vertical:
663       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
664     }
665   }
666 }
667 \cs_set_protected:Nn \__enumext_fake_item_viii:
668 {
669   \dim_compare:nNnT
670     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
671   {
672     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
673     {
674       \exp_not:N \mode_leave_vertical:
675       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
676     }
677   }
678 }

```

(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.

```

679 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
680 {
681   \keys_define:nn { enumext / #1 }
682   {
683     show-length .bool_set:c = { l__enumext_show_length_#2_bool },

```

```

684         show-length .initial:n = false,
685     }
686 }
687 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

## 10.16 Setting before, after and first keys

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

```

before* 688 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    689 {
first    690     \keys_define:nn { enumext / #1 }
        691     {
        692         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
        693         before .value_required:n = true,
        694         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
        695         before* .value_required:n = true,
        696         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
        697         after .value_required:n = true,
        698         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
        699         first .value_required:n = true,
        700     }
        701 }
        702 \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.

```

__enumext_before_args_exec: 703 \cs_new_protected:Nn \__enumext_before_args_exec:
__enumext_before_keys_exec: 704 {
__enumext_after_stop_list: 705     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
__enumext_after_args_exec: 706 }

```

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  $\langle keys \rangle$ .

```

707 \cs_new_protected:Nn \__enumext_before_keys_exec:
708 {
709     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
710 }

```

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

```

711 \cs_new_protected:Nn \__enumext_after_stop_list:
712 {
713     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
714 }

```

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`.

```

715 \cs_new_protected:Nn \__enumext_after_args_exec:
716 {
717     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
718 }

```

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

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

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

```

__enumext_before_args_exec_v: 719 \cs_new_protected:Nn \__enumext_before_args_exec_v:
__enumext_before_keys_exec_v: 720 {
__enumext_after_stop_list_v: 721     \tl_use:N \l__enumext_before_starred_key_v_tl
__enumext_after_args_exec_v: 722 }

```

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

```
723 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
724 {
725     \tl_use:N \l__enumext_before_no_starred_key_v_tl
726 }
```

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

```
727 \cs_new_protected:Nn \__enumext_after_stop_list_v:
728 {
729     \tl_use:N \l__enumext_after_stop_list_v_tl
730 }
```

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

```
731 \cs_new_protected:Nn \__enumext_after_args_exec_v:
732 {
733     \tl_use:N \l__enumext_after_list_args_v_tl
734 }
```

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

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

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

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

```
735 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
736 {
737     \tl_use:N \l__enumext_before_starred_key_vii_tl
738 }
739 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
740 {
741     \tl_use:N \l__enumext_before_starred_key_viii_tl
742 }
```

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

```
743 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
744 {
745     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
746 }
747 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
748 {
749     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
750 }
```

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

```
751 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
752 {
753     \tl_use:N \l__enumext_after_stop_list_vii_tl
754 }
755 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
756 {
757     \tl_use:N \l__enumext_after_stop_list_viii_tl
758 }
```

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

```
759 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
760 {
761     \tl_use:N \l__enumext_after_list_args_vii_tl
762 }
763 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
764 {
765     \tl_use:N \l__enumext_after_list_args_viii_tl
766 }
```

(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

```

767 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
768 {
769   \keys_define:nn { enumext / #1 }
770   {
771     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
772     mini-env .value_required:n = true,
773     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
774     mini-sep .initial:n = 0.3333em,
775     mini-sep .value_required:n = true,
776     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
777     columns-sep .value_required:n = true,
778     columns .int_set:c = { \__enumext_columns_#2_int },
779     columns .initial:n = 1,
780     columns .value_required:n = true,
781   }
782 }
783 \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`.

```

784 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
785 {
786   \keys_define:nn { enumext / #1 }
787   {
788     columns .initial:n = 2,
789     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
790     miniright .value_required:n = true,
791     miniright* .code:n = {
792       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
793       \keys_set:nn { enumext / #1 } { miniright = {##1} }
794     },
795     miniright* .value_required:n = true,
796   }
797 }
798 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

## 10.18 Adjustment of vertical spaces for multicols

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

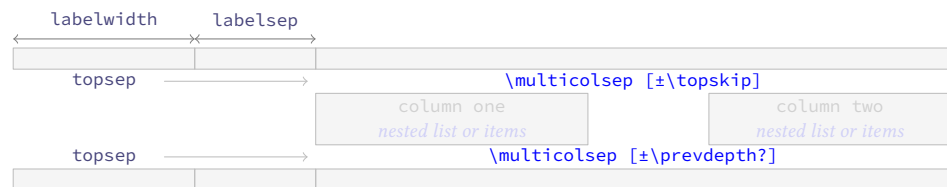


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

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

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

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

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

```

799 \cs_new_protected:Nn \__enumext_multi_set_vskip:
800 {
801   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
802   {
803     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
804   }
805   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
806   {
807     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
808   }
809   \__enumext_add_pre_parsep:
810 }
```

(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*.

```

811 \cs_new_protected:Nn \__enumext_add_pre_parsep:
812 {
813   \int_case:nn { \l__enumext_level_int }
814   {
815     { 2 }{
816       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
817       {
818         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
819       }
820     }
821     { 3 }{
822       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
823       {
824         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
825       }
826     }
827     { 4 }{
828       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
829       {
830         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
831       }
832     }
833   }
834 }
```

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

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

```

835 \cs_new_protected:Nn \__enumext_multi_addvspace:
836 {
837   \__enumext_multi_set_vskip:
838   \mode_if_vertical:T
839   {
840     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
841     {
842       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
843     }
844     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
845     {
846       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
847     }
848   }
```



```

849 \par\nopagebreak
850 \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: \skip } }
851 }

```

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

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

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

```

852 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
853 {
854   \skip_set:Nn \l__enumext_multicols_above_v_skip
855   {
856     \l__enumext_topsep_v_skip
857   }
858   \skip_set:Nn \l__enumext_multicols_below_v_skip
859   {
860     \l__enumext_topsep_v_skip
861   }
862 }
863 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
864 {
865   \l__enumext_keyans_multi_set_vskip:
866   \mode_if_vertical:T
867   {
868     \skip_add:Nn \l__enumext_multicols_above_v_skip
869     {
870       \skip_use:N \l__enumext_partopsep_v_skip
871     }
872     \skip_add:Nn \l__enumext_multicols_below_v_skip
873     {
874       \skip_use:N \l__enumext_partopsep_v_skip
875     }
876   }
877   \par\nopagebreak
878   \addvspace{ \l__enumext_multicols_above_v_skip }
879 }

```

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

### 10.19 Adjustment of vertical spaces for minipage

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

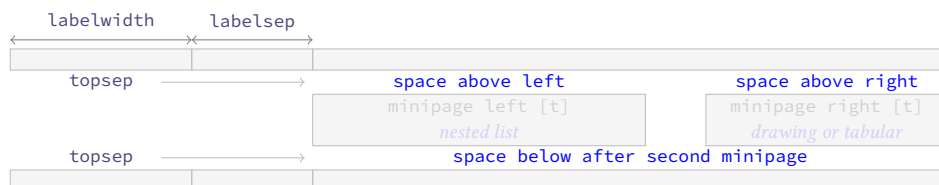


Figure 8: Representation of the `\minipage` spacing adjustment for a nested level.

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

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

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

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

```

880 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
881 {
882     \__enumext_minipage:w [ t ] { #1 }
883     \legacy_if_gset_false:n { @minipage }
884     \vspace { 0pt }
885 }
886 { \__enumext_endminipage: }

```

(End of definition for `__enumext_mini_env*`.)

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

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

We will set the default values taking into account that T<sub>E</sub>X 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`.

```

887 \cs_new_protected:Nn \__enumext_mini_set_vskip:
888 {
889     \int_compare:nNnTF
890     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
891     {

```

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.

```

892     \skip_if_eq:nnTF
893     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
894     {
895         \skip_set:Nn \l__enumext_minipage_left_skip
896         {
897             -0.150\box_dp:N \strutbox
898         }
899         \skip_set:Nn \l__enumext_minipage_right_skip
900         {
901             0.695\box_dp:N \strutbox
902         }
903         \skip_set:Nn \l__enumext_minipage_after_skip
904         {
905             \box_dp:N \strutbox
906         }
907         \__enumext_zero_parsep:
908     }
909     {
910         \skip_set:Nn \l__enumext_minipage_left_skip
911         {
912             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
913         }
914         \skip_set:Nn \l__enumext_minipage_right_skip
915         {
916             0.695\box_dp:N \strutbox
917         }
918         \skip_set:Nn \l__enumext_minipage_after_skip
919         {
920             1.85\box_dp:N \strutbox
921             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
922         }
923     }
924 }
925 {

```

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.

```

926     \skip_if_eq:nnTF
927     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
928     {
929         \skip_set:Nn \l__enumext_minipage_left_skip

```

```

930         {
931             0.5\box_dp:N \strutbox
932             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
933         }
934     \skip_set:Nn \l__enumext_minipage_right_skip
935     {
936         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
937     }
938     \skip_set:Nn \l__enumext_minipage_after_skip
939     {
940         1.6\box_dp:N \strutbox
941     }
942 }
943 {
944     \skip_set:Nn \l__enumext_minipage_left_skip
945     {
946         0.5875\box_dp:N \strutbox
947         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
948     }
949     \skip_set:Nn \l__enumext_minipage_right_skip
950     {
951         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
952         + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
953     }
954     \skip_set:Nn \l__enumext_minipage_after_skip
955     {
956         0.325\box_dp:N \strutbox
957         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
958     }
959 }
960 }
961 }

```

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

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

```

962 \cs_new_protected:Nn \__enumext_zero_parsep:
963 {
964     \int_case:nn { \l__enumext_level_int }
965     {
966         { 2 } {
967             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
968             {
969                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
970             }
971         }
972         { 3 } {
973             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
974             {
975                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
976             }
977         }
978         { 4 } {
979             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
980             {
981                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
982             }
983         }
984     }
985 }

```

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

\\_\_enumext\_mini\_addvspace: The function \\_\_enumext\_mini\_addvspace: will apply the spaces set using \addvspace “above” the \\_\_enumext\_mini\_env\* environment in enumext, taking into account whether T<sub>E</sub>X 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*.

```

986 \cs_new_protected:Nn \__enumext_mini_addvspace:
987 {

```

```

988     \__enumext_mini_set_vskip:
989     \mode_if_vertical:T
990     {
991         \skip_add:Nn \l__enumext_minipage_left_skip
992         {
993             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
994         }
995         \skip_add:Nn \l__enumext_minipage_after_skip
996         {
997             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
998         }
999     }
1000     \par\nopagebreak
1001     \addvspace { \l__enumext_minipage_left_skip }
1002 }

```

(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`.

```

1003 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1004 {
1005     \skip_zero_new:N \l__enumext_minipage_after_skip
1006     \skip_zero_new:N \l__enumext_minipage_left_skip
1007     \skip_zero_new:N \l__enumext_minipage_right_skip
1008     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1009     {
1010         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1011         {
1012             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1013             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1014             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1015             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1016             {
1017                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1018             }
1019         }
1020         {
1021             \skip_set:Nn \l__enumext_minipage_left_skip
1022             {
1023                 \skip_use:N \l__enumext_topsep_v_skip
1024             }
1025             \skip_set:Nn \l__enumext_minipage_right_skip
1026             {
1027                 0.705\box_dp:N \strutbox
1028             }
1029             \skip_set:Nn \l__enumext_minipage_after_skip
1030             {
1031                 1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1032             }
1033         }
1034     }
1035     {
1036         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1037         {
1038             \skip_set:Nn \l__enumext_minipage_left_skip
1039             {
1040                 0.5\box_dp:N \strutbox
1041                 + \l__enumext_partopsep_v_skip
1042             }
1043             \skip_set:Nn \l__enumext_minipage_right_skip
1044             {
1045                 \l__enumext_partopsep_v_skip
1046             }
1047             \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1048         }
1049         {
1050             \skip_set:Nn \l__enumext_minipage_left_skip
1051             {

```

```

1052         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1053     }
1054     \skip_set:Nn \l__enumext_minipage_right_skip
1055     {
1056         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1057     }
1058     \skip_set:Nn \l__enumext_minipage_after_skip
1059     {
1060         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1061     }
1062 }
1063 }
1064 }

```

(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`.

```

1065 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1066 {
1067     \__enumext_keyans_mini_set_vskip:
1068     \mode_if_vertical:T
1069     {
1070         \skip_add:Nn \l__enumext_minipage_left_skip
1071         {
1072             \l__enumext_partopsep_v_skip
1073         }
1074         \skip_add:Nn \l__enumext_minipage_after_skip
1075         {
1076             \l__enumext_partopsep_v_skip
1077         }
1078     }
1079     \par\nopagebreak
1080     \addvspace { \l__enumext_minipage_left_skip }
1081 }

```

(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*`.

```

1082 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1083 {
1084     \skip_zero_new:N \l__enumext_minipage_left_skip
1085     \skip_gzero_new:N \g__enumext_minipage_right_skip
1086     \skip_gzero_new:N \g__enumext_minipage_after_skip
1087     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1088     {
1089         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1090         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1091     }
1092     {
1093         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1094         \skip_gset:Nn \g__enumext_minipage_right_skip
1095         {
1096             \l__enumext_topsep_vii_skip
1097         }
1098         \skip_gset:Nn \g__enumext_minipage_after_skip
1099         {
1100             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1101         }
1102     }
1103 }
1104 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1105 {
1106     \skip_zero_new:N \l__enumext_minipage_after_skip

```

```

1107 \skip_zero_new:N \l__enumext_minipage_left_skip
1108 \skip_zero_new:N \l__enumext_minipage_right_skip
1109 \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1110 {
1111   \skip_set:Nn \l__enumext_minipage_left_skip
1112   {
1113     0.5\box_dp:N \strutbox
1114   }
1115   \skip_set:Nn \l__enumext_minipage_right_skip
1116   {
1117     \l__enumext_partopsep_viii_skip
1118   }
1119   \skip_set:Nn \l__enumext_minipage_after_skip
1120   {
1121     1.6\box_dp:N \strutbox
1122   }
1123 }
1124 {
1125   \skip_set:Nn \l__enumext_minipage_left_skip
1126   {
1127     0.5875\box_dp:N \strutbox
1128   }
1129   \skip_set:Nn \l__enumext_minipage_right_skip
1130   {
1131     \l__enumext_topsep_viii_skip
1132   }
1133   \skip_set:Nn \l__enumext_minipage_after_skip
1134   {
1135     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1136   }
1137 }
1138 }

```

(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.

```

1139 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1140 {
1141   \__enumext_mini_set_vskip_vii:
1142   \par\nopagebreak
1143   \addvspace { \l__enumext_minipage_left_skip }
1144 }
1145 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1146 {
1147   \__enumext_mini_set_vskip_viii:
1148   \par\nopagebreak
1149   \addvspace { \l__enumext_minipage_left_skip }
1150 }

```

(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.

```

1151 \NewDocumentCommand \miniright { s }
1152 {
1153   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1154   {
1155     \msg_error:nnn { enumext } { wrong-miniright-place }

```

```

1156     }
1157     \int_compare:nNt { \__enumext_level_int } = { 0 }
1158     {
1159         \msg_error:nnn { enumext } { wrong-miniright-place }
1160     }
1161     \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1162     {
1163         \__enumext_keyans_mini_right_cmd:n {#1}
1164     }
1165     { \__enumext_mini_right_cmd:n {#1} }
1166 }

```

(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.

```

1167 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1168 {
1169     \dim_compare:nNtF
1170     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1171     {
1172         \__enumext_multicols_stop:
1173         \end{__enumext_mini_env*}
1174         \hfill
1175         \begin{__enumext_mini_env*}
1176         { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1177         \par\addvspace { \__enumext_minipage_right_skip }
1178         \bool_if:nF {#1}
1179         {
1180             \centering
1181         }
1182         \int_gzero:N \g__enumext_minipage_stat_int
1183     }
1184     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1185 }

```

(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.

```

1186 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1187 {
1188     \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1189     {
1190         \__enumext_keyans_multicols_stop:
1191         \end{__enumext_mini_env*}
1192         \hfill
1193         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1194         \par\addvspace { \__enumext_minipage_right_skip }
1195         \bool_if:nF {#1}
1196         {
1197             \centering
1198         }
1199         \int_gzero:N \g__enumext_minipage_stat_int
1200     }
1201     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1202 }

```

(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* 1203 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below  1204 {
below* 1205   \keys_define:nn { enumext / #1 }
        1206   {
        1207     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
        1208     above .value_required:n = true,
        1209     above* .code:n      = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
        1210               \keys_set:nn { enumext / #1 } { above = {##1} },
        1211     above* .value_required:n = true,
        1212     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
        1213     below .value_required:n = true,
        1214     below* .code:n      = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
        1215               \keys_set:nn { enumext / #1 } { below = {##1} },
        1216     below* .value_required:n = true,
        1217   }
        1218 }
1219 \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.

```

1220 \cs_new_protected:Nn \__enumext_vspace_above:
1221 {
1222   \skip_if_eq:nnF
1223   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1224   {
1225     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1226     {
1227       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1228     }
1229     {
1230       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1231     }
1232   }
1233 }

```

(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.

```

1234 \cs_new_protected:Nn \__enumext_vspace_below:
1235 {
1236   \skip_if_eq:nnF
1237   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1238   {
1239     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1240     {
1241       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1242     }
1243     {
1244       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1245     }
1246   }
1247 }

```

(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.

```

1248 \cs_new_protected:Nn \__enumext_vspace_above_v:
1249 {
1250   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1251   {
1252     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1253     {
1254       \vspace*{ \l__enumext_vspace_above_v_skip }
1255     }
1256     { \vspace { \l__enumext_vspace_above_v_skip } }
1257   }
1258 }
```

(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.

```

1259 \cs_new_protected:Nn \__enumext_vspace_below_v:
1260 {
1261   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1262   {
1263     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1264     {
1265       \vspace*{ \l__enumext_vspace_below_v_skip }
1266     }
1267     { \vspace { \l__enumext_vspace_below_v_skip } }
1268   }
1269 }
```

(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:`

```

1270 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1271 {
1272   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1273   {
1274     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1275     {
1276       \vspace*{ \l__enumext_vspace_above_vii_skip }
1277     }
1278     { \vspace { \l__enumext_vspace_above_vii_skip } }
1279   }
1280 }
1281 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1282 {
1283   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1284   {
1285     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1286     {
1287       \vspace*{ \l__enumext_vspace_above_viii_skip }
1288     }
1289     { \vspace { \l__enumext_vspace_above_viii_skip } }
1290   }
1291 }
```

(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:`

```

1292 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1293 {
1294   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1295   {
1296     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1297     {
1298       \vspace*{ \l__enumext_vspace_below_vii_skip }

```

```

1299     }
1300     { \vspace { \l__enumext_vspace_below_vii_skip } }
1301   }
1302 }
1303 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1304 {
1305   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1306   {
1307     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1308     {
1309       \vspace*{ \l__enumext_vspace_below_viii_skip }
1310     }
1311     { \vspace { \l__enumext_vspace_below_viii_skip } }
1312   }
1313 }

```

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

### 10.21 Setting save-ans and resume keys

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

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

```

save-ans \keys_define:nn { enumext / level-1 }
resume   {
resume*   {
1314   \keys_define:nn { enumext / level-1 }
1315   {
1316     save-ans .code:n = \__enumext_storing_set:n {#1},
1317     save-ans .value_required:n = true,
1318     resume  .code:n = \__enumext_resume_counter:,
1319     resume  .value_forbidden:n = true,
1320     resume* .code:n = \__enumext_resume_counter_star:,
1321     resume* .value_forbidden:n = true,
1322   }
1323   \keys_define:nn { enumext / enumext* }
1324   {
1325     save-ans .code:n = \__enumext_storing_set:n {#1},
1326     save-ans .value_required:n = true,
1327     resume  .code:n = \__enumext_resume_counter_vii:,
1328     resume  .value_forbidden:n = true,
1329   }

```

(End of definition for `save-ans`, `resume`, and `resume*`.)

\\_\_enumext\_storing\_set:n

The function `\__enumext_storing_set:n` executed by the `save-ans` key sets the parameters for the operation of `\anskey`, `keyans` and `keyanspic`. The variable `\l__enumext_store_name_tl` will have the “store name” with which the *sequence* and *prop list* will be created.

The boolean var `\l__enumext_store_active_bool` will be set to true activating the entire internal *storage mechanism*, then the integer variable for the `resume` key will be created (if not exist), finally the function `\__enumext_check_ans_int:n` will be called to activate the internal mechanism for checking the answers if the boolean variable `\l__enumext_check_ans_bool` set by `check-ans` key are active.

```

1330 \cs_new_protected:Npn \__enumext_storing_set:n #1
1331 {
1332   \tl_set:Nx \l__enumext_store_name_tl {#1}
1333   \bool_set_true:N \l__enumext_store_active_bool
1334   \int_if_exist:cF { g__enumext_resume_#1_int }
1335   {
1336     \int_new:c { g__enumext_resume_#1_int }
1337   }
1338   \bool_if:NT \l__enumext_check_ans_bool
1339   {
1340     \__enumext_check_ans_int:n {#1}
1341   }
1342 }

```

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

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

The functions `\__enumext_resume_counter:` and `\__enumext_resume_counter_vii:` used by `resume` key in `enumext` and `enumext*`. If `save-ans` key present then set the start value from integer created by `\__enumext_storing_set:n`.

```

1343 \cs_new_protected:Nn \__enumext_resume_counter:

```

```

1344 {
1345     \bool_if:NT \l__enumext_store_active_bool
1346     {
1347         \int_gset:Nn \g__enumext_resume_int
1348         {
1349             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1350         }
1351     }
1352     \bool_set_true:N \l__enumext_resume_bool
1353 }
1354 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1355 {
1356     \bool_if:NT \l__enumext_store_active_bool
1357     {
1358         \int_gset:Nn \g__enumext_resume_int
1359         {
1360             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1361         }
1362     }
1363     \bool_set_true:N \l__enumext_resume_vii_bool
1364 }

```

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

## 10.22 Setting check-ans key

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 increment 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_all_int` must match the integer variable `\g__enumext_count_item_ans_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 increase by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.
- c) If there is the option `no-store` we must add the items within this level plus one to maintain the equality.

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

### 10.22.1 The check answer mechanism

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

```

check-ans no-store 1365 \cs_set_protected:Npn \__enumext_tmp:n #1
1366 {
1367     \keys_define:nn { enumext / #1 }
1368     {
1369         check-ans .bool_set:N = \l__enumext_check_ans_bool,
1370         check-ans .initial:n = false,
1371         no-store .code:n = {
1372             \bool_set_false:N \l__enumext_store_ans_bool
1373             \bool_set_false:N \l__enumext_check_ans_bool
1374         },

```

```

1375         no-store .value_forbidden:n = true,
1376     }
1377 }
1378 \clist_map_inline:nn
1379 {
1380     level-1, level-2, level-3, level-4, enumext*
1381 }
1382 { \__enumext_tmp:n {#1} }

```

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

`\__enumext_check_ans_int:n`

The function `\__enumext_check_ans_int:n` will create the integer variables for the internal checking answer mechanism used by the `check-ans` key. The integer variables take the form `\g__enumext_count_⟨store name⟩_item_ans_int` and `\g__enumext_count_⟨store name⟩_item_X_int`

```

1383 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1384 {
1385     \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1386     { \int_new:c { g__enumext_count_#1_item_ans_int } }
1387     \int_if_exist:cF { g__enumext_count_#1_i_int }
1388     { \int_new:c { g__enumext_count_#1_i_int } }
1389     \int_if_exist:cF { g__enumext_count_#1_ii_int }
1390     { \int_new:c { g__enumext_count_#1_ii_int } }
1391     \int_if_exist:cF { g__enumext_count_#1_iii_int }
1392     { \int_new:c { g__enumext_count_#1_iii_int } }
1393     \int_if_exist:cF { g__enumext_count_#1_iv_int }
1394     { \int_new:c { g__enumext_count_#1_iv_int } }
1395     \int_if_exist:cF { g__enumext_count_#1_vii_int }
1396     { \int_new:c { g__enumext_count_#1_vii_int } }

```

We make `\g__enumext_count_item_all_int` equal to the integer variables `\g__enumext_count_⟨store name⟩_item_i_int` or `\g__enumext_count_⟨store name⟩_item_vii_int` that contains all the occurrences of `\item` and `\item*` in the different levels and we will make `\g__enumext_count_item_with_ans_int` equal to the integer variable handled by the `\anskey` command.

```

1397     \bool_lazy_all:nTF
1398     {
1399         { \g__enumext_starred_bool }
1400         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1401     }
1402     {
1403         \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1404     }
1405     {
1406         \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1407     }
1408     \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1409     \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1410     \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1411     \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1412     \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1413     \int_gset_eq:Nc \g__enumext_count_item_with_ans_int { g__enumext_count_#1_item_ans_int }
1414 }

```

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

### 10.22.2 Set-up check answer mechanism

`\__enumext_check_ans_count:`

The function `\__enumext_check_ans_count:` 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.

```

1415 \cs_new_protected:Npn \__enumext_check_ans_count:
1416 {
1417     \bool_if:NT \l__enumext_check_ans_bool
1418     {
1419         \bool_if:NTF \l__enumext_store_ans_bool
1420         {
1421             \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1422             { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1423         }
1424         { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1425     }
1426 }

```

(End of definition for `\__enumext_check_ans_count:`)

`\__enumext_check_ans_active:` The function `\__enumext_check_ans_active:` compare all `\item`'s plus `\item*`'s and `\item`'s with answer for checking answer mechanism and display the appropriate message on the terminal.

`\__enumext_check_ans_active_vii:`

```

1427 \cs_new_protected:Nn \__enumext_check_ans_active:
1428 {
1429   \int_set:Nn \l__enumext_compare_items_ans_int
1430   {
1431     \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1432     - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1433   }
1434   \int_compare:nNnTF
1435   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1436   {
1437     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1438   }
1439   {
1440     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1441   }

```

After the function is executed, we set the temporary integer variables to zero.

```

1442   \int_gzero:N \g__enumext_count_level_i_int
1443   \int_gzero:N \g__enumext_count_level_ii_int
1444   \int_gzero:N \g__enumext_count_level_iii_int
1445   \int_gzero:N \g__enumext_count_level_iv_int
1446   \int_gzero:N \g__enumext_count_level_vii_int
1447 }

1448 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1449 {
1450   \int_set:Nn \l__enumext_compare_items_ans_int
1451   {
1452     \g__enumext_count_item_all_int - \g__enumext_count_item_i_int
1453     - \g__enumext_count_item_ii_int - \g__enumext_count_item_iii_int
1454     - \g__enumext_count_item_iv_int
1455   }
1456   \int_compare:nNnTF
1457   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1458   {
1459     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1460   }
1461   {
1462     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1463   }
1464   \int_gzero:N \g__enumext_count_level_i_int
1465   \int_gzero:N \g__enumext_count_level_ii_int
1466   \int_gzero:N \g__enumext_count_level_iii_int
1467   \int_gzero:N \g__enumext_count_level_iv_int
1468   \int_gzero:N \g__enumext_count_level_vii_int
1469 }

```

(End of definition for `\__enumext_check_ans_active:` and `\__enumext_check_ans_active_vii:`)

## 10.23 Keys and functions associated with storage

`wrap-ans` `mark-ans` `mark-pos` `show-ans` `show-pos` `mark-ref` and `store-ref` related to the “storage system” and internal mechanism of “label and ref” only at the first level of `enumext` and `enumext*`.

```

1470 \cs_set_protected:Npn \__enumext_tmp:n #1
1471 {
1472   \keys_define:nn { enumext / #1 }
1473   {
1474     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,
1475     wrap-ans .initial:n = \fbox{##1},
1476     wrap-ans .value_required:n = true,
1477     mark-ans .code:n = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1478     mark-ans .initial:n = \textasteriskcentered,
1479     mark-ans .value_required:n = true,
1480     mark-pos .choice:,
1481     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1482     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1483     mark-pos .initial:n = right,

```

```

1484     mark-pos    .value_required:n = true,
1485     show-ans    .code:n          = \bool_set_true:N \__enumext_show_answer_bool
1486                                     \bool_set_false:N \__enumext_show_position_bool,
1487     show-ans    .value_forbidden:n = true,
1488     show-pos    .code:n          = \bool_set_true:N \__enumext_show_position_bool
1489                                     \bool_set_false:N \__enumext_show_answer_bool,
1490     show-pos    .value_forbidden:n = true,
1491     mark-ref    .code:n          = \tl_set:Nn \__enumext_mark_ref_sym_tl {##1},
1492     mark-ref    .initial:n       = \textasteriskcentered,
1493     mark-ref    .value_required:n = true,
1494     store-ref   .bool_set:N      = \__enumext_store_ref_key_bool,
1495     store-ref   .initial:n       = false,
1496   }
1497 }
1498 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1499 \cs_set_protected:Npn \__enumext_tmp:n #1
1500 {
1501   \keys_define:nn { enumext / #1 }
1502   {
1503     mark-pos .choice:,
1504     mark-pos / left .code:n    = \str_set:Nn \__enumext_mark_position_str { l },
1505     mark-pos / right .code:n   = \str_set:Nn \__enumext_mark_position_str { r },
1506     mark-pos      .initial:n   = right,
1507     mark-pos      .value_required:n = true,
1508     show-ans      .code:n      = \bool_set_true:N \__enumext_show_answer_bool
1509                                     \bool_set_false:N \__enumext_show_position_bool,
1510     show-ans      .value_forbidden:n = true,
1511     show-pos      .code:n      = \bool_set_true:N \__enumext_show_position_bool
1512                                     \bool_set_false:N \__enumext_show_answer_bool,
1513     show-pos      .value_forbidden:n = true,
1514   }
1515 }
1516 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1517 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1518 {
1519   \keys_define:nn { enumext / #1 }
1520   {
1521     columns*      .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1522                                     \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1523                                     \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1524                                     {
1525                                         columns = \exp_not:v { l__enumext_store_columns_#2_int },
1526                                     },,
1527     columns*      .value_required:n = true,
1528     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1529                                     \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1530                                     \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1531                                     {
1532                                         columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1533                                     },
1534     columns-sep* .value_required:n = true,
1535   }
1536 }
1537 \clist_map_inline:nn
1538 {
1539   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1540 }
1541 { \__enumext_tmp:nn #1 }

```

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



### 10.23.1 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  $\langle prop\ list \rangle$  defined by `save-ans` key, if it does not exist it will create it globally. The “stored content” is retrieved by means of the `\getkeyans` command.

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

```
1542 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1543 {
1544   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1545   { \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1546   \prop_gput:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1547   {
1548     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } +1 }
1549     } { #1 }
1550 }
1551 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

### 10.23.2 Function for storing content in sequence

```
\__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  $\langle sequence \rangle$  defined by `save-ans` key, if it does not exist it will create it globally. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`. The form in which the content is stored in  $\langle sequence \rangle$  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.

```
1552 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1553 {
1554   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1555   { \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq } }
1556   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1557 }
1558 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

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

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

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

```
1559 \cs_new_protected:Nn \__enumext_store_level_open:
1560 {
1561   \bool_if:NF \l__enumext_store_ans_bool
1562   {
1563     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1564     {
1565       \__enumext_store_addto_seq:n
1566       {
1567         \item \begin{enumext}
1568       }
1569     }
1570     {
1571       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1572       {
1573         \item \begin{enumext} [
1574       }
1575       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1576       {
1577         ]
1578       }
1579       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1580     }
1581   }
1582 }
1583 \cs_new_protected:Nn \__enumext_store_level_close:
```

```

1584 {
1585   \bool_if:NF \l__enumext_store_ans_bool
1586   {
1587     \__enumext_store_addto_seq:n { \end{enumext} }
1588   }
1589 }

```

(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.

```

1590 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1591 {
1592   \bool_if:NF \l__enumext_store_ans_bool
1593   {
1594     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1595     {
1596       \__enumext_store_addto_seq:n
1597       {
1598         \item \mode_leave_vertical:
1599         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1600         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1601       }
1602     }
1603     {
1604       \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1605       {
1606         \item \mode_leave_vertical:
1607         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1608         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1609       ]
1610       \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1611       {
1612         ]
1613       }
1614       \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1615     }
1616   }
1617 }
1618 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1619 {
1620   \bool_if:NF \l__enumext_store_ans_bool
1621   {
1622     \__enumext_store_addto_seq:n { \end{enumext*} }
1623   }
1624 }

```

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

#### 10.23.4 Function for show marks and position

```

\__enumext_print_keyans_box:NN
\__enumext_print_keyans_box:cc

```

The function `\__enumext_print_keyans_box:NN` print a box in the left margin with `\l__enumext_mark_answer_sym_tl` used by the `wrap-ans`, `show-ans` and `show-pos` keys. The function takes two arguments:

- #1: `\l__enumext_labelwidth_X_dim`
- #2: `\l__enumext_labelsep_X_dim`

```

1625 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1626 {
1627   \mode_leave_vertical:
1628   \skip_horizontal:n { -\dim_use:N #2 }
1629   \makebox[\opt][ r ]
1630   {
1631     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1632     {
1633       \tl_use:N \l__enumext_mark_answer_sym_tl
1634     }
1635   }
1636   \skip_horizontal:n { \dim_use:N #2 }

```

```

1637 }
1638 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for `\__enumext_print_keyans_box:NN`.)

## 10.24 The command `\anskey` and internal label and ref

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

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

```

1639 \keys_define:nn { enumext / anskey }
1640 {
1641   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1642   item-sym* .value_required:n = true,
1643   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1644   item-pos* .value_required:n = true,
1645   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1646   store-brk .default:n = true,
1647   store-brk .value_forbidden:n = true,
1648 }

```

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)*.

```

1649 \NewDocumentCommand \anskey { d() s o +m }
1650 {
1651   \bool_if:NF \l__enumext_store_active_bool
1652   {
1653     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1654   }
1655   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1656   {
1657     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1658   }
1659   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1660   {
1661     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1662   }
1663   \group_begin:
1664     \bool_if:NF \l__enumext_store_ans_bool
1665     {
1666       \bool_if:NT \l__enumext_check_ans_bool
1667       {
1668         \int_gincr:N \g__enumext_count_item_with_ans_int
1669       }
1670       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1671     }
1672   \group_end:
1673 }

```

(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 `store-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.

```

1674 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1675 {
1676   \__enumext_store_addto_prop:n {#4}
1677   \bool_if:NT \l__enumext_store_ref_key_bool
1678   {
1679     \__enumext_store_internal_ref:
1680   }
1681   \__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`.

```

1682 \tl_clear:N \l__enumext_store_anskey_arg_tl
1683 \tl_if_novalue:nF {#3}
1684 {
1685   \keys_set:nn { enumext / anskey } {#3}
1686 }
1687 \bool_lazy_and:nnT
1688 { \l__enumext_store_columns_break_bool }
1689 { \bool_not_p:n { \l__enumext_starred_bool } }
1690 {
1691   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1692 }
1693 \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).

```

1694 \tl_if_novalue:nF {#1}
1695 {
1696   \int_set:Nn \l__enumext_store_columns_join_int {#1}
1697   \bool_if:NT \l__enumext_starred_bool
1698   {
1699     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1700     {
1701       ( \exp_not:V \l__enumext_store_columns_join_int )
1702     }
1703   }
1704 }

```

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`.

```

1705 \bool_if:nTF {#2}
1706 {
1707   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1708   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1709   {
1710     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1711     {
1712       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1713     }
1714   }
1715   \dim_compare:nT
1716   {
1717     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1718   }
1719   {
1720     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1721     {
1722       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1723     }
1724   }
1725   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1726 }
1727 {
1728   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1729 }

```

Finally we check if the `store-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*.

```

1730 \bool_lazy_and:nnT
1731 { \l__enumext_store_ref_key_bool }
1732 { \l__enumext_hyperref_bool }
1733 {
1734   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1735   {
1736     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1737     { \exp_not:V \l__enumext_mark_ref_sym_tl }
1738   }

```

```

1739     }
1740     \__enumext_store_addto_seq:V \__enumext_store_anskey_arg_tl
1741 }

```

(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 store-ref and mark-ref keys for \anskey will allow to execute \ref{<store name: position>} and will return 1.(a).i.A.

First we will remove the dots “.” from the current <labels>, we do not want to get double dots in our references, then we will place this in the variable \\_\_enumext\_newlabel\_arg\_two\_tl.

```

1742 \cs_new_protected:Nn \__enumext_store_internal_ref:
1743 {
1744   \cs_set_protected:Npn \__enumext_tmp:n ##1
1745   {
1746     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
1747     \tl_reverse:c { \__enumext_label_copy_##1_tl }
1748     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
1749     \tl_reverse:c { \__enumext_label_copy_##1_tl }
1750   }
1751   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1752   \cs_set:Npn \__enumext_tmp:n ##1
1753   { . \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

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

```

1754   \bool_lazy_all:nT
1755   {
1756     { \g__enumext_starred_bool }
1757     { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
1758   }
1759   {
1760     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1761     { \tl_use:N \__enumext_label_copy_vii_tl }
1762   }
1763   \bool_lazy_all:nT
1764   {
1765     { \l__enumext_standar_bool }
1766     { \g__enumext_starred_bool }
1767     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
1768   }
1769   {
1770     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1771     {
1772       \tl_use:N \__enumext_label_copy_vii_tl
1773       \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
1774     }
1775   }

```

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.

```

1776   \bool_lazy_all:nT
1777   {
1778     { \l__enumext_standar_bool }
1779     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
1780     { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_zero_int } }
1781     { \bool_not_p:n { \l__enumext_starred_bool } }
1782   }
1783   {
1784     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1785     {
1786       \tl_use:N \__enumext_label_copy_i_tl
1787       \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
1788     }
1789   }
1790   \cs_set:Npn \__enumext_tmp:n ##1
1791   { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1792   \bool_lazy_all:nT
1793   {
1794     { \l__enumext_standar_bool }
1795     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }

```

```

1796     { \bool_not_p:n { \g__enumext_starred_bool } }
1797     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1798   }
1799   {
1800     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1801     {
1802       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1803       . \tl_use:N \l__enumext_label_copy_vii_tl
1804     }
1805   }

```

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

```

1806   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1807   {
1808     \l__enumext_store_name_tl \c_colon_str
1809     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1810   }

```

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.

```

1811   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1812   {
1813     \__enumext_newlabel:nn
1814     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1815     { \l__enumext_newlabel_arg_two_tl }
1816   }
1817   \l__enumext_store_write_aux_file_tl
1818 }

```

(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.

```

1819 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1820 {
1821   \par
1822   \bool_if:NT \l__enumext_starred_bool
1823   {
1824     \cs_set:Nn \__enumext_level: { vii }
1825   }
1826   \__enumext_print_keyans_box:cc
1827   { \l__enumext_labelwidth_ \__enumext_level: _dim }
1828   { \l__enumext_labelsep_ \__enumext_level: _dim }
1829   \__enumext_anskey_wrapper:n { #1 }
1830 }

```

(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  $\langle \textit{prop list} \rangle$  when using the `show-pos` key on the left margin next to the “wraps”  $\langle \textit{argument} \rangle$  passed to `\anskey` on the right side when using the `show-ans` key.

```

1831 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1832 {
1833   \bool_if:NT \l__enumext_show_answer_bool
1834   {
1835     \__enumext_store_anskey_show_wrap:n { #1 }
1836   }
1837   \bool_if:NT \l__enumext_show_position_bool
1838   {
1839     \tl_set:Ne \l__enumext_mark_answer_sym_tl
1840     {
1841       \group_begin:
1842       \exp_not:N \normalfont
1843       \exp_not:N \footnotesize [ \int_eval:n
1844       {
1845         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1846       }
1847       ]
1848       \group_end:

```

```

1849     }
1850     \__enumext_store_anskey_show_wrap:n { #1 }
1851   }
1852 }

```

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

## 10.25 Common functions for keyans and keyanspic

### 10.25.1 Storing content in prop list

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

```

1853 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1854 {
1855   \tl_clear:N \__enumext_store_keyans_label_tl
1856   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1857   {
1858     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_vi_tl }
1859   }
1860   {
1861     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_v_tl }
1862   }
1863   \tl_if_novalue:nF { #1 }
1864   {
1865     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \c_space_tl #1 }
1866   }
1867   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
1868 }

```

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

### 10.25.2 The store-ref key for keyans and keyanspic

The internal “*label and ref*” system for the `keyans` and `keyanspic` environments has slight differences with the one implemented for the `\anskey` command, basically because in both environments we are interested in the *current label*.

`\__enumext_keyans_internal_ref:` The function `\__enumext_keyans_internal_ref:` handles the internal “*label and ref*” system used by the `store-ref` and `mark-ref` keys for `\item*` and `\anspic*` commands. The mechanism defined here will allow to execute `\ref{(store name: position)}` and will return `1.` (A).

```

1869 \cs_new_protected:Nn \__enumext_keyans_internal_ref:
1870 {

```

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`.

```

1871   \cs_set_protected:Npn \__enumext_tmp:n ##1
1872   {
1873     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
1874     \tl_reverse:c { \__enumext_label_copy_##1_tl }
1875     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
1876     \tl_reverse:c { \__enumext_label_copy_##1_tl }
1877   }
1878   \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1879   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1880   {
1881     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1882     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
1883   }
1884   {
1885     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1886     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
1887   }

```

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

```

1888   \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
1889   {
1890     \__enumext_store_name_tl \c_colon_str

```



```

1891     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1892   }

```

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.

```

1893   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1894   {
1895     \__enumext_newlabel:nn
1896     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1897     { \l__enumext_newlabel_arg_two_tl }
1898   }
1899   \bool_if:NT \l__enumext_store_ref_key_bool
1900   {
1901     \l__enumext_store_write_aux_file_tl
1902   }
1903 }

```

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

### 10.25.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n`

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the `\l__enumext_label_v_tl` (“current label”) for the `keyans` environment and the `\l__enumext_label_vi_tl` (current label) 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.

```

1904 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1905 {
1906   \tl_clear:N \l__enumext_store_keyans_label_tl
1907   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1908   {
1909     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1910   }
1911   {
1912     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1913   }
1914   \tl_if_novalue:nF { #1 }
1915   {
1916     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1917   }

```

Checks if the `store-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store using the `\__enumext_store_addto_seq:V` function.

```

1918   \bool_lazy_and:nnT
1919   { \l__enumext_store_ref_key_bool }
1920   { \l__enumext_hyperref_bool }
1921   {
1922     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
1923     {
1924       \hfill \exp_not:N \hyperlink
1925       {
1926         \exp_not:V \l__enumext_newlabel_arg_one_tl
1927       }
1928       { \exp_not:V \l__enumext_mark_ref_sym_tl }
1929     }
1930   }
1931   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl

```

Finally, copy the contents of the variable `\l__enumext_store_keyans_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `\__enumext_keyans_check_ans:nn` and increment the value of the integer variable `\g__enumext_count_item_with_ans_int` handled by the `check-ans` key.

```

1932   \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1933   \bool_if:NT \l__enumext_check_ans_bool
1934   {
1935     \int_gincr:N \g__enumext_count_item_with_ans_int
1936   }
1937 }

```

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

#### 10.25.4 Check for starred commands

`\__enumext_keyans_check_ans:nn`

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

```

1938 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1939 {
1940   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1941   {
1942     \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1943   }
1944   { \tl_gclear:N \g__enumext_check_ans_item_tl }
1945 }

```

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

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

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

`\__enumext_keyans_show_left:n`

Common function to show *starred commands* and *position* of stored content in *prop list* for `keyans` and `keyanspic`. Need add `1` to `\l__enumext_mark_answer_sym_tl` for `keyans` environment.

```

1946 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1947 {
1948   \bool_if:NT \l__enumext_show_answer_bool
1949   {
1950     \tl_put_left:Nn \l__enumext_label_v_tl
1951     {
1952       \__enumext_print_keyans_box:NN
1953       \l__enumext_labelwidth_i_dim
1954       \l__enumext_labelsep_i_dim
1955     }
1956     \tl_if_novalue:nF { #1 }
1957     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1958   }
1959   \bool_if:NT \l__enumext_show_position_bool
1960   {
1961     \tl_set:Nc \l__enumext_mark_answer_sym_tl
1962     {
1963       \group_begin:
1964       \exp_not:N \normalfont
1965       \exp_not:N \footnotesize [ \int_eval:n
1966       {
1967         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1968         + \l__enumext_keyans_level_int
1969       }
1970       ]
1971       \group_end:
1972     }
1973     \tl_put_left:Nn \l__enumext_label_v_tl
1974     {
1975       \__enumext_print_keyans_box:NN
1976       \l__enumext_labelwidth_i_dim
1977       \l__enumext_labelsep_i_dim
1978     }
1979     \tl_if_novalue:nF { #1 }
1980     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1981   }
1982 }

```

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

#### 10.26 Setting item-sym\* and item-pos\* keys

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

`item-sym*`

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

`item-pos*`

```

1983 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1984 {
1985   \keys_define:nn { enumext / #1 }
1986   {

```

```

1987         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
1988         item-sym* .value_required:n = true,
1989         item-sym* .initial:n = { $\star$ },
1990         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
1991         item-pos* .value_required:n = true,
1992     }
1993 }
1994 \clist_map_inline:nn
1995 {
1996     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1997 }
1998 { \__enumext_tmp:nn #1 }

```

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

## 10.27 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 @cfr in [footnotes in boxes compatible with hyperref](#).

```

1999 \cs_new_protected:Nn \__enumext_footnotetext:nn
2000 {
2001     \footnotetext[#1]{#2}
2002 }
2003 \cs_new_protected:Nn \__enumext_renew_footnote:
2004 {
2005     \seq_gclear:N \g__enumext_footnote_arg_seq
2006     \seq_gclear:N \g__enumext_footnote_int_seq
2007     \RenewDocumentCommand \footnote { o +m }
2008     {
2009         \tl_if_novalue:nTF {##1}
2010         {
2011             \stepcounter{footnote}
2012             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2013         }
2014         {
2015             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2016         }
2017         \footnotemark [ \g__enumext_footnote_int ]
2018         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2019         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2020     }
2021 }
2022 \cs_new_protected:Nn \__enumext_print_footnote:
2023 {
2024     \seq_if_empty:NF \g__enumext_footnote_int_seq
2025     {
2026         \seq_map_pairwise_function:NNN
2027         \g__enumext_footnote_int_seq
2028         \g__enumext_footnote_arg_seq
2029         \__enumext_footnotetext:nn
2030     }
2031 }

```

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

## 10.28 Redefining `\item` command

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

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

```
\__enumext_default_item:n
```

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `\__enumext_make_label:` (§10.29).

```

2032 \cs_new_protected:Npn \__enumext_default_item:n #1
2033 {
2034   \tl_if_novalue:nTF {#1}
2035   {
2036     \bool_if:NT \l__enumext_check_ans_bool
2037     {
2038       \int_gincr:N \g__enumext_count_item_all_int
2039       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2040     }
2041     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2042     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2043   }
2044   {
2045     \bool_set_eq:cc
2046     { l__enumext_wrap_label_ \__enumext_level: _bool }
2047     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2048     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2049   }
2050 }

```

(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:` (§10.29).

```

2051 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2052 {
2053   \tl_if_novalue:nF {#1}
2054   {
2055     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2056   }
2057   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2058   \tl_if_novalue:nTF {#2}
2059   {
2060     \dim_set_eq:cc
2061     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2062     { l__enumext_labelsep_ \__enumext_level: _dim }
2063   }
2064   {
2065     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2066   }
2067   \bool_if:NT \l__enumext_check_ans_bool
2068   {
2069     \int_gincr:N \g__enumext_count_item_all_int
2070     \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2071   }
2072   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2073   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2074 }

```

(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).

```

2075 \cs_new_protected:Nn \__enumext_redefine_item:
2076 {
2077   \RenewDocumentCommand \item { s o o }
2078   {
2079     \bool_if:nTF {##1}
2080     {
2081       \__enumext_starred_item:nn {##2} {##3}
2082     }
2083     { \__enumext_default_item:n {##2} }
2084   }
2085 }

```

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

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

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

`\__enumext_keyans_default_item:n`

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

```

2086 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2087 {
2088   \tl_if_novalue:nTF { #1 }
2089   {
2090     \bool_set_true:N \__enumext_wrap_label_v_bool
2091     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2092   }
2093   {
2094     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2095     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2096   }
2097 }

```

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

`\__enumext_keyans_starred_item:n`

The function `\__enumext_keyans_starred_item:n` which will make a temporary copy of the “current label”, execute the `show-ans` or `show-pos` keys using the function `\__enumext_keyans_show_left:n` and will display the contents of that item using the internal copy `\__enumext_item_std:w`, this is necessary to prevent incrementing the current “counter” of the original `\label`.

```

2098 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2099 {
2100   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2101   \__enumext_keyans_show_left:n { #1 }
2102   \bool_set_true:N \__enumext_wrap_label_v_bool
2103   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```

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

```

2104   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2105   \__enumext_keyans_addto_prop:n { #1 }
2106   \__enumext_keyans_internal_ref:
2107   \__enumext_keyans_addto_seq:n { #1 }
2108 }

```

(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).

```

2109 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2110 {
2111   \RenewDocumentCommand \item { s o }
2112   {
2113     \bool_if:nTF {##1}
2114     {

```

```

2115         \peek_remove_spaces:n
2116         {
2117             \__enumext_keyans_starred_item:n {##2}
2118         }
2119     }
2120     {
2121         \__enumext_keyans_default_item:n {##2}
2122     }
2123 }
2124 }

```

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

## 10.29 Redefining `\makeLabel` command

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

### 10.29.1 Redefining `\makeLabel` for `enumext`

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

```

2125 \cs_new_protected:Nn \__enumext_item_starred:
2126 {
2127     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2128     {
2129         \mode_leave_vertical:
2130         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2131         \makebox[0pt][r]{ \tl_use:N \g__enumext_item_symbol_tl }
2132         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2133     }
2134 }

```

(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).

```

2135 \cs_new_protected:Nn \__enumext_make_label:
2136 {
2137     \RenewDocumentCommand \makeLabel { m }
2138     {
2139         \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2140         \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2141         \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2142         {
2143             \__enumext_item_starred:
2144             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2145         }
2146         { ##1 }
2147         \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2148         \tl_gclear:N \g__enumext_item_symbol_tl
2149     }
2150 }

```

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

### 10.29.2 Redefining `\makeLabel` for `keyans`

`\__enumext_keyans_make_label:` The function `\__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment. This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.31).

```

2151 \cs_new_protected:Nn \__enumext_keyans_make_label:
2152 {
2153     \RenewDocumentCommand \makeLabel { m }
2154     {
2155         \tl_use:N \l__enumext_label_fill_left_v_tl
2156         \tl_use:N \l__enumext_label_font_style_v_tl
2157         \bool_if:NTF \l__enumext_wrap_label_v_bool
2158         {
2159             \__enumext_wrapper_label_v:n { ##1 }
2160         }
2161         { ##1 }

```

```

2162     \tl_use:N \l__enumext_label_fill_right_v_tl
2163   }
2164 }

```

(End of definition for `\__enumext_keyans_make_label:`)

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

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

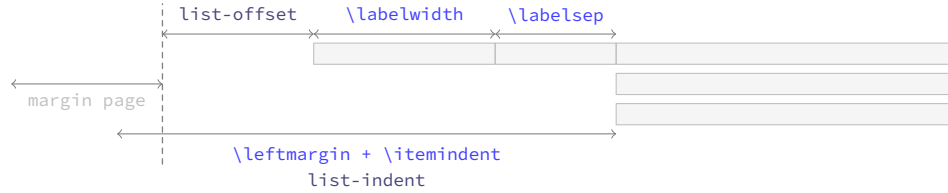


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

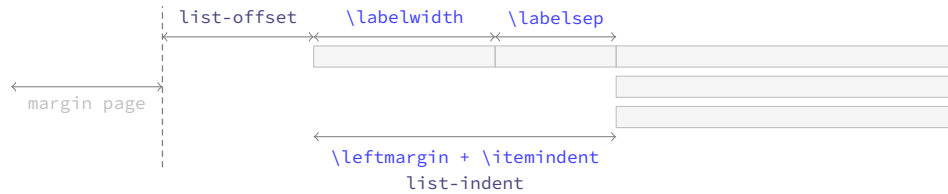


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

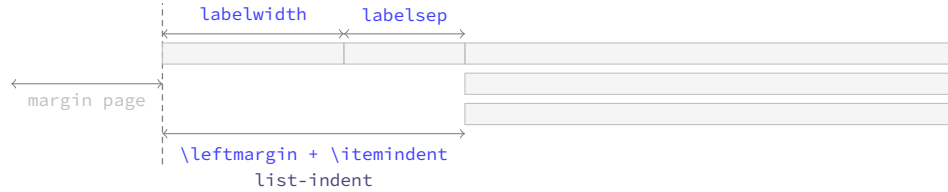


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

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).

```

2165 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2166 {
2167   \dim_compare:nNt { #1 } < { \c_zero_dim }
2168   {
2169     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2170     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2171   }
2172   \dim_compare:nNt { #2 } < { \c_zero_dim }
2173   {
2174     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2175     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2176   }

```

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

```

2177   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```



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

```

2178 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2179 {
2180   \dim_set:Nn #6 { #1 + #2 - #4 }
2181   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2182 }
2183 {
2184   \dim_compare:nNnT { #4 } = { #1 + #2 }
2185   { \dim_set:Nn #6 { \c_zero_dim } }
2186   \dim_compare:nNnT { #4 } < { #1 + #2 }
2187   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2188   \dim_compare:nNnT { #4 } > { #1 + #2 }
2189   {
2190     \dim_set:Nn #6 { -#1 - #2 + #4 }
2191     \dim_set:Nn #6 { #6*-1 }
2192   }
2193   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2194 }
2195 }
2196 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

### 10.31 Setting second argument of the lists

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

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments.

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

```

2197 \cs_set_protected:Npn \__enumext_tmp:n #1
2198 {
2199   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2200   {
2201     \__enumext_calc_hspace:ccccc
2202     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2203     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2204     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2205     { l__enumext_leftmargin_tmp_#1_bool }
2206     \clist_map_inline:nn
2207     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2208     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2209     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2210     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2211     \usecounter { enumX#1 }
2212     \bool_lazy_and:nnTF
2213     { \str_if_eq_p:nn {#1} { i } }
2214     { \bool_if_p:N \__enumext_resume_bool }
2215     { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2216     {
2217       \setcounter { enumX#1 }
2218       { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2219     }
2220     \str_if_eq:nnTF {#1} { v }
2221     {
2222       \__enumext_keyans_redefine_item:
2223       \__enumext_keyans_make_label:
2224       \__enumext_keyans_fake_item:
2225       \bool_if:cT { l__enumext_show_length_#1_bool }
2226       {
2227         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2228       }
2229     }
2230     {
2231       \__enumext_redefine_item:
2232       \__enumext_make_label:
2233       \__enumext_use_key_ref:

```

```

2234         \__enumext_fake_item:
2235         \bool_if:cT { \__enumext_show_length_#1_bool }
2236         {
2237             \msg_term:nne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2238         }
2239     }
2240 }
2241 }
2242 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for \\_\_enumext\_list\_arg\_two\_i: and others.)

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

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

```

2243 \cs_set_protected:Npn \__enumext_tmp:n #1
2244 {
2245     \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
2246     {
2247         \__enumext_calc_hspace:ccccc
2248         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2249         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2250         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2251         { \__enumext_leftmargin_tmp_#1_bool }
2252         \clist_map_inline:nn
2253         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2254         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2255         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2256         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2257         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2258         \skip_zero:N \partopsep
2259         \usecounter { enumX#1 }
2260         \bool_lazy_and:nnTF
2261         { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2262         { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2263         {
2264             \setcounter { enumX#1 }
2265             { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2266         }
2267         \__enumext_use_key_ref_h:
2268         \str_if_eq:nnTF {#1} { vii }
2269         {
2270             \__enumext_fake_item_vii:
2271             \bool_if:cT { \__enumext_show_length_vii_bool }
2272             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2273         }
2274         {
2275             \__enumext_fake_item_viii:
2276             \bool_if:cT { \__enumext_show_length_#1_bool }
2277             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2278         }
2279     }
2280 }
2281 \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.

```

2282 \NewDocumentEnvironment{enumext}{0} {
2283     {
2284         \__enumext_safe_exec:
2285         \__enumext_parse_keys:n {#1}
2286         \__enumext_before_list:
2287         \__enumext_start_store_level:
2288         \__enumext_start_list:nn
2289         { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2290         {
2291             \use:c { \__enumext_list_arg_two_ \__enumext_level: : }
2292             \__enumext_before_keys_exec:

```

```

2293     }
2294     \__enumext_after_args_exec:
2295 }
2296 {
2297     \__enumext_stop_list:
2298     \__enumext_stop_store_level:
2299     \__enumext_after_list:
2300 }

```

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

`\__enumext_safe_exec:` First check the maximum nesting level for the `enumext` environment and set the state of the booleans `\l__enumext_standar_bool` and `\g__enumext_standar_bool` to “true”, the latter only if the environment is NOT nested in the `enumext*` environment.

```

2301 \cs_new_protected:Nn \__enumext_safe_exec:
2302 {
2303     \int_incr:N \l__enumext_level_int
2304     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2305     { \msg_fatal:nn { enumext } { list-too-deep } }
2306     \bool_set_true:N \l__enumext_standar_bool
2307     \bool_lazy_all:nT
2308     {
2309         { \bool_not_p:n { \l__enumext_starred_bool } }
2310         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2311     }
2312     {
2313         \bool_gset_true:N \g__enumext_standar_bool
2314     }
2315 }

```

(End of definition for `\__enumext_safe_exec:`)

`\__enumext_parse_keys:n` Parse [`<key = val>`] by levels in `enumext`. If the variable `\l__enumext_store_active_bool` is true it will call the function `\__enumext_parse_store_keys:n` and reprocess the `<keys>` to pass them to the storage sequence.

```

2316 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2317 {
2318     \exp_args:Ne \keys_set:nn
2319     { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2320     \bool_if:NT \l__enumext_store_active_bool
2321     {
2322         \__enumext_parse_store_keys:n {#1}
2323     }
2324 }

```

(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:`.

```

2325 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2326 {
2327     \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2328     {
2329         \regex_match:nnT { \b columns\b } {#1}
2330         {
2331             \int_set_eq:cc
2332             { \l__enumext_store_columns_ \__enumext_level: _int }
2333             { \l__enumext_columns_ \__enumext_level: _int }
2334             \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2335             {
2336                 columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2337             }
2338         }
2339     }
2340     \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2341     {
2342         \regex_match:nnT { \b columns-sep\b } {#1}
2343         {

```

```

2344         \dim_set_eq:cc
2345         { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2346         { l__enumext_columns_sep_ \__enumext_level: _dim }
2347         \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2348         {
2349             columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2350         }
2351     }
2352 }
2353 }

```

(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.

```

2354 \cs_new_protected:Nn \__enumext_start_store_level:
2355 {
2356     %\bool_lazy_and:nnT
2357     %{ \__enumext_store_active_bool }
2358     %{ \bool_not_p:n { \__enumext_keyans_env_bool } }
2359     %{
2360         %\int_compare:nNnT { \__enumext_level_h_int } = { 1 }
2361         %{
2362             %\bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2363             %\__enumext_store_level_open:
2364         }%}
2365         %\int_compare:nNnT { \__enumext_level_int } > { 1 }
2366         %{
2367             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2368             \__enumext_store_level_open:
2369         }%}
2370     }%}
2371 }
2372 \cs_new_protected:Nn \__enumext_stop_store_level:
2373 {
2374     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2375     {
2376         \__enumext_store_level_close:
2377     }
2378 }

```

(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.

```

2379 \cs_new_protected:Nn \__enumext_before_list:
2380 {
2381     \__enumext_vspace_above:
2382     \__enumext_before_args_exec:

```

The function \\_\_enumext\_check\_ans\_count: will handle the check answer mechanism, which will be activated with the check-ans key.

```

2383     \__enumext_check_ans_count:

```

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.

```

2384     \dim_compare:nNnT
2385     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2386     {
2387         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2388         {
2389             \linewidth
2390             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2391             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2392         }

```

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

- Here we use the plain T<sub>E</sub>X macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a vertical list.

```

2393     \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2394     \int_gincr:N \g__enumext_minipage_stat_int
2395     \__enumext_mini_addvspace:
2396     \nointerlineskip\noindent
2397     \begin{\__enumext_mini_env*}
2398         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2399     }
2400     \__enumext_multicols_start:
2401 }

```

(End of definition for `\__enumext_before_list:`)

`\__enumext_multicols_start:` The function `\__enumext_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key, then set the default value for `\columnsep` when `columns-sep=opt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

2402 \cs_new_protected:Nn \__enumext_multicols_start:
2403 {
2404     \int_compare:nNt
2405     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2406     {
2407         \dim_compare:nNt
2408         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2409         {
2410             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2411             {
2412                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2413                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2414                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2415                   - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2416             }
2417         }
2418         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2419         \skip_zero:N \multicolsep
2420         \int_compare:nNt { \l__enumext_level_int } > { 1 }
2421         {
2422             \dim_zero:N \columnseprule
2423         }
2424     }
2425 }

```

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.

```

2424     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2425     {
2426         \__enumext_multi_addvspace:
2427     }
2428     \raggedcolumns
2429     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2430 }
2431 }

```

(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.

```

2432 \cs_new_protected:Nn \__enumext_multicols_stop:
2433 {
2434     \int_compare:nNt
2435     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2436     {
2437         \end{multicols}
2438     }
2439 }

```

```

2438     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2439     {
2440         \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2441     }
2442 }

```

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

```

2443 \bool_lazy_and:nnT
2444 { \l__enumext_check_ans_bool }
2445 { \bool_not_p:n { \g__enumext_starred_bool } }
2446 {
2447     \bool_gset_true:N \g__enumext_check_ans_show_bool
2448     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2449 }
2450 }

```

(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.

```

2451 \cs_new_protected:Nn \__enumext_after_list:
2452 {
2453     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2454     {
2455         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2456         {
2457             \msg_warning:nn { enumext } { missing-miniright }
2458             \miniright
2459         }
2460         \int_gzero:N \g__enumext_minipage_stat_int
2461         \end{__enumext_mini_env*}
2462         \par\addvspace { \l__enumext_minipage_after_skip }
2463     }
2464     { \__enumext_multicols_stop: }

```

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

```

2465 \__enumext_after_stop_list:
2466 \__enumext_vspace_below:

```

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

```

2467 \bool_set_false:N \l__enumext_standar_bool
2468 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2469 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2470 {
2471     \int_gset_eq:cN
2472     { g__enumext_resume_ \l__enumext_store_name_tl _int }
2473     { \value{enumXi} }
2474 }
2475 }

```

(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`.

```

2476 \__enumext_after_env:nn {enumext}
2477 {
2478     \bool_if:NT \g__enumext_check_ans_show_bool
2479     {
2480         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2481         {
2482             \__enumext_check_ans_active:
2483         }
2484     }

```

```

2485     \bool_gset_false:N \g__enumext_check_ans_show_bool
2486     \tl_gclear:N \g__enumext_store_name_tl
2487 }

```

### 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.

```

2488 \NewDocumentEnvironment{keyans}{ 0{} }
2489 {
2490     \__enumext_keyans_safe_exec:
2491     \__enumext_keyans_parse_keys:n {#1}
2492     \__enumext_before_list_v:
2493     \__enumext_start_list:nn
2494     { \tl_use:N \l__enumext_label_v_tl }
2495     {
2496         \__enumext_list_arg_two_v:
2497         \__enumext_before_keys_exec_v:
2498     }
2499     \__enumext_after_args_exec_v:
2500 }
2501 {
2502     \__enumext_keyans_check_ans:nn { item }{ keyans }
2503     \__enumext_stop_list:
2504     \__enumext_after_list_v:
2505 }

```

(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.

```

2506 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2507 {
2508     \bool_if:NF \l__enumext_store_active_bool
2509     {
2510         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2511     }
2512     \int_incr:N \l__enumext_keyans_level_int
2513     \bool_set_true:N \l__enumext_keyans_env_bool
2514     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2515     \bool_set_false:N \l__enumext_store_active_bool
2516     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2517     {
2518         \msg_error:nn { enumext } { keyans-nested }
2519     }
2520     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2521     {
2522         \msg_error:nn { enumext } { keyans-wrong-level }
2523     }
2524 }

```

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

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

```

2525 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2526 {
2527     \keys_set:nn { enumext / keyans } {#1}
2528 }

```

(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.

```

2529 \cs_new_protected:Nn \__enumext_before_list_v:
2530 {
2531     \__enumext_vspace_above_v:
2532     \__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.

```

2533   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2534   {
2535     \dim_set:Nn \l__enumext_minipage_left_v_dim
2536     {
2537       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2538     }

```

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*.

```

2539     \bool_set_true:N \l__enumext_minipage_active_v_bool
2540     \int_gincr:N \g__enumext_minipage_stat_int
2541     \__enumext_keyans_mini_addvspace:
2542     \nointerlineskip\noindent
2543     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2544   }

```

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

```

2545   \__enumext_keyans_multicols_start:
2546 }

```

(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.

```

2547 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2548 {
2549   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2550   {

```

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

```

2551     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2552     {
2553       \dim_set:Nn \l__enumext_columns_sep_v_dim
2554       {
2555         (
2556           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2557         ) / \l__enumext_columns_v_int
2558         - \l__enumext_listoffset_v_dim
2559       }
2560     }
2561     \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).

```

2562     \skip_zero:N \multicolsep
2563     \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.

```

2564     \bool_if:NF \l__enumext_minipage_active_v_bool
2565     {
2566       \__enumext_keyans_multi_addvspace:
2567     }
2568     \raggedcolumns
2569     \begin{multicols}{ \l__enumext_columns_v_int }
2570   }
2571 }

```

(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.

```

2572 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2573 {
2574   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2575   {
2576     \end{multicols}
2577     \bool_if:NF \l__enumext_minipage_active_v_bool
2578     {
2579       \par\addvspace{ \l__enumext_multicols_below_v_skip }
2580     }
2581   }
2582 }

```

(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.

```

2583 \cs_new_protected:Nn \__enumext_after_list_v:
2584 {
2585   \bool_if:NTF \l__enumext_minipage_active_v_bool
2586   {
2587     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2588     {
2589       \msg_warning:nn { enumext } { missing-miniright }
2590       \miniright
2591     }
2592     \int_gzero:N \g__enumext_minipage_stat_int
2593     \end{__enumext_mini_env*}
2594     \par\addvspace{ \l__enumext_minipage_after_skip }
2595   }
2596   { \__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.

```

2597   \bool_set_false:N \l__enumext_keyans_env_bool
2598   \__enumext_after_stop_list_v:
2599   \__enumext_vspace_below_v:
2600 }

```

(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.

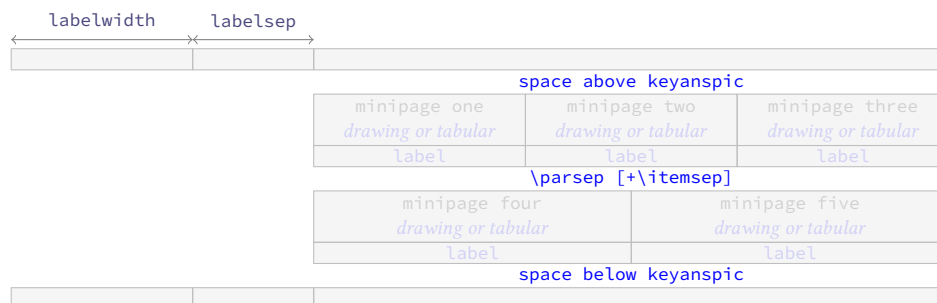


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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.

```
2601 \NewDocumentCommand \anspic { s o +m }
2602 {
```

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

```
2603   \bool_if:NF \__enumext_store_active_bool
2604   {
2605     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2606   }
2607   \int_compare:nNnT { \__enumext_level_int } > { 1 }
2608   {
2609     \msg_error:nn { enumext } { keyanspic-wrong-level }
2610   }
2611   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2612   {
2613     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2614   }
```

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.

```
2615   \seq_put_right:Nn \__enumext_keyans_pic_body_seq
2616   {
2617     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2618   }
2619 }
```

(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.

```
2620 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2621 {
2622   \stepcounter { enumXvi }
2623   #3 \\\
2624   \bool_if:nT { #1 }
2625   {
2626     \__enumext_keyans_addto_prop:n { #2 }
2627     \__enumext_keyans_internal_ref:
2628     \__enumext_keyans_addto_seq:n { #2 }
2629     \bool_lazy_or:nnT
2630     { \__enumext_show_answer_bool }
2631     { \__enumext_show_position_bool }
2632     {
2633       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
2634       \__enumext_keyans_show_left:n { #2 }
2635       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
2636     }
2637   }
2638   \tl_use:N \__enumext_label_font_style_v_tl
2639   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
2640 }
```

(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.

```
2641 \NewDocumentEnvironment{keyanspic}{ o }
2642 {
2643   \__enumext_keyans_pic_safe_exec:
2644   \__enumext_start_list:nn
2645   { }
2646   {
2647     \__enumext_keyans_pic_arg_two:
2648   }
```

We apply the “adjusted” vertical spacing above the environment

```
2649 \vspace { \l__enumext_keyans_pic_above_skip }
2650 }
```

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.

```
2651 {
2652   \tl_if_novalue:nTF { #1 }
2653   {
2654     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2655   }
2656   { \__enumext_keyans_pic_do:n { #1 } }
2657   \__enumext_stop_list:
2658   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2659   \setcounter { enumXvi } { 0 }
2660   \vspace { \l__enumext_topsep_v_skip }
2661   %\bool_set_false:N \l__enumext_store_active_bool
2662 }
```

(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.

```
2663 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2664 {
2665   \int_incr:N \l__enumext_keyans_pic_level_int
2666   \int_compare:nNt { \l__enumext_keyans_pic_level_int } > { 1 }
2667   {
2668     \msg_error:nn { enumext } { keyanspic-nested }
2669   }
2670 }
```

(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`.

```
2671 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2672 {
2673   \dim_compare:nNt { #1 } < { 0pt }
2674   { \skip_set:Nn #1 { -#1 } }
2675 }
```

(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.

```
2676 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2677 {
```

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.

```
2678   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2679   \__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.

```
2680   \skip_add:Nn \parsep { \itemsep }
2681   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2682   \dim_zero:N \labelwidth
2683   \dim_zero:N \listparindent
2684   \dim_zero:N \labelsep
2685   \skip_zero:N \partopsep
2686   \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.

```

2687 \__enumext_keyans_pic_skip_abs:N \parsep
2688 \skip_set:Nn \l__enumext_keyans_pic_above_skip
2689 {
2690   \box_dp:N \strutbox
2691   + \l__enumext_topsep_v_skip
2692   - \parsep
2693 }
2694 \__enumext_item_std:w \scan_stop:
2695 }

```

(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`.

```

2696 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2697 {
2698   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2699 }
2700 \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.

```

2701 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2702 {
2703   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2704   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2705   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2706   \int_step_inline:nnn
2707     { \l__enumext_keyans_pic_above_int + 1 }
2708     { \l__enumext_keyans_pic_below_int }
2709     {
2710       \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2711       \centering
2712       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2713       \__enumext_endminipage:
2714     }
2715   \par
2716 }

```

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

## 10.35 The enumext\* and keyans\* environments

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

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

2717 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2718 {
2719   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2720   {
2721     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2722     {
2723       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )

```

```

2724         / \l__enumext_columns_vii_int
2725     }
2726 }
2727 \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2728 \dim_set:Nn \l__enumext_item_width_vii_dim
2729 {
2730     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2731     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2732     - \l__enumext_labelsep_vii_dim
2733 }
2734 \dim_zero_new:N \itemwidth
2735 }

```

(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`.

```

2736 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2737 {
2738     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2739     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2740     {
2741         \msg_warning:nnee { enumext } { item-joined }
2742         { \int_use:N \l__enumext_joined_item_vii_int }
2743         { \int_use:N \l__enumext_columns_vii_int }
2744         \int_set:Nn \l__enumext_joined_item_vii_int
2745         {
2746             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2747         }
2748     }
2749     \int_compare:nNnT
2750     { \l__enumext_joined_item_vii_int }
2751     >
2752     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2753     {
2754         \msg_warning:nnee { enumext } { item-joined-columns }
2755         { \int_use:N \l__enumext_joined_item_vii_int }
2756         {
2757             \int_eval:n
2758             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2759         }
2760         \int_set:Nn \l__enumext_joined_item_vii_int
2761         {
2762             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2763         }
2764     }

```

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

```

2765     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2766     {
2767         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2768         \int_decr:N \l__enumext_joined_item_aux_vii_int
2769         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2770         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2771         \dim_set:Nn \l__enumext_joined_width_vii_dim
2772         {
2773             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2774             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2775                 + \l__enumext_columns_sep_vii_dim
2776             ) * \l__enumext_joined_item_aux_vii_int
2777         }
2778         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2779     }
2780     {
2781         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2782         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2783     }
2784 }

```

(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`.

```

2785 \cs_new_protected:Nn \__enumext_start_mini_vii:
2786 {
2787   \dim_compare:nNtT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2788   {
2789     \dim_set:Nn \l__enumext_minipage_left_vii_dim
2790     {
2791       \linewidth
2792       - \l__enumext_minipage_right_vii_dim
2793       - \l__enumext_minipage_hsep_vii_dim
2794     }
2795     \bool_set_true:N \l__enumext_minipage_active_vii_bool
2796     \dim_gset_eq:NN
2797       \g__enumext_minipage_right_vii_dim
2798       \l__enumext_minipage_right_vii_dim
2799     \__enumext_mini_addvspace_vii:
2800     \nointerlineskip\noindent
2801     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2802   }
2803 }

```

(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*”.

```

2804 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2805 {
2806   \bool_if:NT \l__enumext_minipage_active_vii_bool
2807   {
2808     \end{__enumext_mini_env*}
2809     \hfill
2810     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2811   }
2812 }

```

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*”.

```

2813 \__enumext_after_env:nn {enumext*}
2814 {
2815   \bool_if:NT \g__enumext_minipage_active_vii_bool
2816   {
2817     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2818     \par\addvspace { \g__enumext_minipage_right_skip }
2819     \bool_if:NF \g__enumext_minipage_center_vii_bool
2820     {
2821       \centering
2822     }
2823     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2824     \end{__enumext_mini_env*}
2825     \par\addvspace{ \g__enumext_minipage_after_skip }
2826   }
2827   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2828   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2829   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2830   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2831 }

```

(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.

```

2832 \NewDocumentEnvironment{enumext*}{ o }
2833 {
2834   \__enumext_safe_exec_vii:

```



```

2835     \__enumext_parse_keys_vii:n {#1}
2836     \__enumext_before_list_vii:
2837     \__enumext_start_store_level_vii:
2838     \__enumext_start_list:nn { }
2839     {
2840         \__enumext_list_arg_two_vii:
2841         \__enumext_before_keys_exec_vii:
2842     }
2843     \__enumext_starred_columns_set_vii:
2844     \item[] \scan_stop:
2845     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2846     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2847 }
2848 {
2849     \__enumext_stop_item_tmp_vii:
2850     \__enumext_remove_extra_parsep_vii:
2851     \__enumext_stop_list:
2852     \__enumext_stop_store_level_vii:
2853     \__enumext_after_list_vii:
2854 }

```

(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`.

```

2855 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2856 {
2857     \int_incr:N \l__enumext_level_h_int
2858     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2859     {
2860         \msg_error:nn { enumext } { nested }
2861     }
2862     \bool_set_true:N \l__enumext_starred_bool
2863     \bool_lazy_all:nT
2864     {
2865         { \bool_not_p:n { \l__enumext_standar_bool } }
2866         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2867     }
2868     {
2869         \bool_gset_true:N \g__enumext_starred_bool
2870     }
2871 }

```

(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.

```

2872 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2873 {
2874     \tl_if_novalue:nF {#1}
2875     {
2876         \keys_set:nn { enumext / enumext* } {#1}
2877         \bool_if:NT \l__enumext_store_active_bool
2878         {
2879             \__enumext_parse_store_keys_vii:n {#1}
2880         }
2881     }
2882 }

```

(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:`.

```

2883 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2884 {
2885     \bool_if:NF \l__enumext_store_columns_vii_bool

```

```

2886     {
2887         \regex_match:nnT { \b columns\b } {#1}
2888         {
2889             \int_set_eq:NN
2890             \l__enumext_store_columns_vii_int
2891             \l__enumext_columns_vii_int
2892             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2893             {
2894                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2895             }
2896         }
2897     }
2898     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2899     {
2900         \regex_match:nnT { \b columns-sep\b } {#1}
2901         {
2902             \dim_set_eq:NN
2903             \l__enumext_store_columns_sep_vii_dim
2904             \l__enumext_columns_sep_vii_dim
2905             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2906             {
2907                 columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2908             }
2909         }
2910     }
2911 }

```

(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`.

```

2912 \cs_new_protected:Nn \__enumext_before_list_vii:
2913 {
2914     \__enumext_vspace_above_vii:
2915     \__enumext_before_args_exec_vii:
2916     \__enumext_start_mini_vii:
2917 }

```

(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.

```

2918 \cs_new_protected:Nn \__enumext_after_list_vii:
2919 {
2920
2921     \__enumext_stop_mini_vii:
2922     \__enumext_after_stop_list_vii:
2923     \__enumext_vspace_below_vii:
2924     \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2925     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2926     {
2927         \int_gset_eq:cN
2928         { g__enumext_resume_ \l__enumext_store_name_tl _int }
2929         { \value{enumXvii} }
2930     }
2931     \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2932     {
2933         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2934         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2935     }
2936     \bool_gset_false:N \g__enumext_starred_bool
2937     \bool_set_false:N \l__enumext_starred_bool
2938 }

```

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

\\_\_enumext\_start\_store\_level\_vii:  
 \\_\_enumext\_stop\_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.

```

2939 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2940 {
2941   \bool_if:NT \l__enumext_store_active_bool
2942   {
2943     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2944     {
2945       \__enumext_store_level_open_vii:
2946     }
2947   }
2948 }
2949 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2950 {
2951   \bool_if:NT \l__enumext_store_active_bool
2952   {
2953     \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2954     {
2955       \__enumext_store_level_close_vii:
2956     }
2957   }
2958 }

```

(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.

```

2959 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2960 {
2961   \__enumext_stop_item_tmp_vii:
2962   \int_incr:N \l__enumext_item_column_pos_vii_int
2963   \int_gincr:N \g__enumext_item_count_all_vii_int
2964   \__enumext_item_peek_args_vii:
2965 }

```

(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).

```

2966 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2967 {
2968   \peek_meaning:NTF (
2969     { \__enumext_joined_item_vii:w }
2970     { \__enumext_joined_item_vii:w (1) }
2971   }

```

(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.

```

2972 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2973 {
2974   \__enumext_starred_joined_item_vii:n {#1}
2975   \peek_meaning_remove:NTF *
2976     { \__enumext_starred_item_vii:w }
2977     { \__enumext_standard_item_vii:w }
2978 }

```

(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 ].

```

2979 \cs_new_protected:Npn \__enumext_standard_item_vii:w
2980 {
2981   \bool_set_false:N \l__enumext_item_starred_vii_bool
2982   \peek_meaning:NTF [
2983     {
2984       \bool_set_eq:NN
2985         \l__enumext_wrap_label_vii_bool
2986         \l__enumext_wrap_label_opt_vii_bool
2987       \__enumext_start_item_vii:w
2988     }
2989     {
2990       \bool_set_true:N \l__enumext_wrap_label_vii_bool
2991       \legacy_if_set_true:n { @noitemarg }
2992       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
2993     }
2994   }

```

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

\\_\_enumext\_starred\_item\_vii:w  
 \\_\_enumext\_starred\_item\_vii\_aux\_i:w  
 \\_\_enumext\_starred\_item\_vii\_aux\_ii:w  
 \\_\_enumext\_starred\_item\_vii\_aux\_iii:w

The function \\_\_enumext\_starred\_item\_vii:w together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute \item\*, \item\*[⟨*symbol*⟩] and \item\*[⟨*symbol*⟩][⟨*offset*⟩].

```

2995 \cs_new_protected:Npn \__enumext_starred_item_vii:w
2996 {
2997   \bool_set_true:N \l__enumext_item_starred_vii_bool
2998   \bool_set_true:N \l__enumext_wrap_label_vii_bool
2999   \peek_meaning:NTF [
3000     { \__enumext_starred_item_vii_aux_i:w }
3001     { \__enumext_starred_item_vii_aux_ii:w }
3002   }
3003   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3004   {
3005     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3006     \__enumext_starred_item_vii_aux_ii:w
3007   }
3008   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3009   {
3010     \peek_meaning:NTF [
3011       { \__enumext_starred_item_vii_aux_iii:w }
3012       {
3013         \dim_set_eq:NN
3014           \l__enumext_item_symbol_sep_vii_dim
3015           \l__enumext_labelsep_vii_dim
3016         \legacy_if_set_true:n { @noitemarg }
3017         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3018       }
3019     }
3020   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3021   {
3022     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3023     \legacy_if_set_true:n { @noitemarg }
3024     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3025   }

```

(End of definition for \\_\_enumext\_starred\_item\_vii:w and others.)

### Real definition of \item

The functions \\_\_enumext\_start\_item\_vii:w and \\_\_enumext\_stop\_item\_vii: executing the true definition of \item inside the `enumext*` environment.

\\_\_enumext\_start\_item\_vii:w

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.

```

3026 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3027 {
3028   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3029   \legacy_if:nT { @noitemarg }
3030   {
3031     \legacy_if_set_false:n { @noitemarg }
3032     \legacy_if:nT { @nmbrrlist }
3033     {
3034       \bool_if:NT \l__enumext_hyperref_bool
3035       {
3036         \legacy_if_set_true:n { @hyper@item }
3037       }
3038       \refstepcounter{enumXvii}
3039       % code for check-ans
3040       \bool_if:NT \l__enumext_check_ans_bool
3041       {
3042         % If true |no-store| key => nested in |enumext|
3043         \bool_if:NTF \l__enumext_store_ans_bool
3044         {
3045           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3046           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3047         }
3048         {
3049           \int_gincr:N \g__enumext_count_item_all_int
3050           \int_gincr:N \g__enumext_count_level_vii_int
3051         }
3052       }
3053     }
3054   }

```

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.

```

3055 \group_begin:
3056 \lrbox{ \l__enumext_item_text_vii_box }
3057 \bool_if:NF \l__enumext_footnotes_key_bool
3058 {
3059   \__enumext_renew_footnote:
3060 }
3061 \bool_if:NT \l__enumext_item_starred_vii_bool
3062 {
3063   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3064   {
3065     \tl_gset_eq:NN
3066     \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3067   }
3068   \mode_leave_vertical:
3069   \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3070   \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3071   \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3072   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3073 }
3074 \group_begin:
3075 \tl_use:N \l__enumext_label_font_style_vii_tl
3076 \bool_if:NTF \l__enumext_wrap_label_vii_bool
3077 {
3078   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3079   { \__enumext_wrapper_label_vii:n {#1} }
3080 }
3081 {
3082   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3083 }
3084 \group_end:
3085 \skip_horizontal:N \l__enumext_labelsep_vii_dim
3086 \tl_use:N \l__enumext_after_list_args_vii_tl

```

```

3087     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3088     \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3089     \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3090     \tl_use:N \__enumext_fake_item_indent_vii_tl
3091 }

```

(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.

```

3092 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3093 {
3094     \__enumext_endminipage:
3095     \endlrbox
3096     \group_end:
3097     \box_set_wd:Nn \__enumext_item_text_vii_box
3098     {
3099         \l__enumext_joined_width_vii_dim
3100         + \l__enumext_labelwidth_vii_dim
3101         + \l__enumext_labelsep_vii_dim
3102     }
3103     \int_set:Nn \hbadness { 10000 }
3104     \box_use:N \__enumext_item_text_vii_box
3105     \bool_if:NF \__enumext_footnotes_key_bool
3106     {
3107         \__enumext_print_footnote:
3108     }
3109     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3110     {
3111         \par\noindent
3112         \int_zero:N \__enumext_item_column_pos_vii_int
3113     }
3114     { \hspace{ \__enumext_columns_sep_vii_dim } }
3115 }

```

(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.

```

3116 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3117 {
3118     \int_compare:nNnT
3119     {
3120         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3121     }
3122     =
3123     { \c_zero_int }
3124     {
3125         \par
3126         \vspace{ -\l__enumext_itemsep_vii_skip }
3127         \int_gzero:N \g__enumext_item_count_all_vii_int
3128     }
3129 }

```

(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.

```

3130 \__enumext_after_env:nn {enumext*}
3131 {
3132     \bool_if:NT \g__enumext_check_ans_show_h_bool
3133     {
3134         \int_compare:nNnT { \__enumext_level_int } = { 0 }
3135         {
3136             \__enumext_check_ans_active_vii:
3137         }
3138     }
3139     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3140     \tl_gclear:N \g__enumext_store_name_tl
3141 }

```

## 10.36 The keyans\* environment

### 10.36.1 Functions for item box width

\\_enumext\_starred\_columns\_set\_viii:

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3142 \cs_new_protected:Nn \_enumext_starred_columns_set_viii:
3143 {
3144   \dim_compare:nNt { \_enumext_columns_sep_viii_dim } = { \c_zero_dim }
3145   {
3146     \dim_set:Nn \_enumext_columns_sep_viii_dim
3147     {
3148       ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim )
3149       / \_enumext_columns_viii_int
3150     }
3151   }
3152   \int_set:Nn \_enumext_tmpa_viii_int { \_enumext_columns_viii_int - \c_one_int }
3153   \dim_set:Nn \_enumext_item_width_viii_dim
3154   {
3155     ( \linewidth - \_enumext_columns_sep_viii_dim * \_enumext_tmpa_viii_int )
3156     / \_enumext_columns_viii_int - \_enumext_labelwidth_viii_dim
3157     - \_enumext_labelsep_viii_dim
3158   }
3159   \dim_zero_new:N \itemwidth
3160 }

```

(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`.

```

3161 \cs_new_protected:Npn \_enumext_starred_joined_item_viii:n #1
3162 {
3163   \int_set:Nn \_enumext_joined_item_viii_int {#1}
3164   \int_compare:nNt { \_enumext_joined_item_viii_int } > { \_enumext_columns_viii_int }
3165   {
3166     \msg_warning:nnee { enumext } { item-joined }
3167     { \int_use:N \_enumext_joined_item_viii_int }
3168     { \int_use:N \_enumext_columns_viii_int }
3169     \int_set:Nn \_enumext_joined_item_viii_int
3170     {
3171       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3172     }
3173   }
3174   \int_compare:nNt
3175   { \_enumext_joined_item_viii_int }
3176   >
3177   { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3178   {
3179     \msg_warning:nnee { enumext } { item-joined-columns }
3180     { \int_use:N \_enumext_joined_item_viii_int }
3181     {
3182       \int_eval:n
3183       { \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int }
3184     }
3185     \int_set:Nn \_enumext_joined_item_viii_int
3186     {
3187       \_enumext_columns_viii_int - \_enumext_item_column_pos_viii_int + \c_one_int
3188     }
3189   }

```

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

```

3190 \int_compare:nNtTF { \_enumext_joined_item_viii_int } > { \c_one_int }
3191 {
3192   \int_set_eq:NN \_enumext_joined_item_aux_viii_int \_enumext_joined_item_viii_int
3193   \int_decr:N \_enumext_joined_item_aux_viii_int
3194   \int_add:Nn \_enumext_item_column_pos_viii_int { \_enumext_joined_item_aux_viii_int }
3195   \int_gadd:Nn \g__enumext_item_count_all_viii_int { \_enumext_joined_item_aux_viii_int }
3196   \dim_set:Nn \_enumext_joined_width_viii_dim
3197   {
3198     \_enumext_item_width_viii_dim * \_enumext_joined_item_viii_int
3199     + ( \_enumext_labelwidth_viii_dim + \_enumext_labelsep_viii_dim
3200       + \_enumext_columns_sep_viii_dim

```



```

3201         )*\__enumext_joined_item_aux_viii_int
3202     }
3203     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3204 }
3205 {
3206     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3207     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3208 }
3209 }

```

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

The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:
3210 \cs_new_protected:Nn \__enumext_start_mini_viii:
3211 {
3212     \dim_compare:nNt { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3213     {
3214         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3215         {
3216             \linewidth
3217             - \l__enumext_minipage_right_viii_dim
3218             - \l__enumext_minipage_hsep_viii_dim
3219         }
3220         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3221         \dim_gset_eq:NN
3222             \g__enumext_minipage_right_viii_dim
3223             \l__enumext_minipage_right_viii_dim
3224         \__enumext_mini_addvspace_viii:
3225         \nointerlineskip\noindent
3226         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3227     }
3228 }
3229 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3230 {
3231     \bool_if:NT \l__enumext_minipage_active_viii_bool
3232     {
3233         \end{__enumext_mini_env*}
3234         \hfill
3235         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3236     }
3237 }
3238 \__enumext_after_env:nn {keyans*}
3239 {
3240     \bool_if:NT \g__enumext_minipage_active_viii_bool
3241     {
3242         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3243         \par\addvspace { \g__enumext_minipage_right_skip }
3244         \bool_if:NF \g__enumext_minipage_center_viii_bool
3245         {
3246             \centering
3247         }
3248         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3249         \end{__enumext_mini_env*}
3250         \par\addvspace{ \g__enumext_minipage_after_skip }
3251     }
3252     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3253     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3254     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3255     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3256 }

```

(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.

```

3257 \NewDocumentEnvironment{keyans*}{ o }
3258 {
3259     \__enumext_safe_exec_viii:
3260     \__enumext_parse_keys_viii:n {#1}
3261     \__enumext_before_list_viii:

```

```

3262     \__enumext_start_list:nn { }
3263     {
3264         \__enumext_list_arg_two_viii:
3265         \__enumext_before_keys_exec_viii:
3266     }
3267     \__enumext_starred_columns_set_viii:
3268     \item[] \scan_stop:
3269     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3270     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3271 }
3272 {
3273     \__enumext_stop_item_tmp_viii:
3274     \__enumext_remove_extra_parsep_viii:
3275     \__enumext_stop_list:
3276     \__enumext_after_list_viii:
3277 }

```

(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.

```

3278 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3279 {
3280     \int_incr:N \l__enumext_keyans_level_h_int
3281     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3282     {
3283         \msg_error:nn { enumext } { nested }
3284     }
3285     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3286     \bool_set_false:N \l__enumext_store_active_bool
3287     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3288     {
3289         \msg_error:nn { enumext } { keyans-wrong-level }
3290     }
3291 }

```

(End of definition for `\__enumext_safe_exec_viii:`)

`\__enumext_parse_keys_viii:n` Parse [`⟨key = val⟩`] for `keyans*`.

```

3292 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3293 {
3294     \tl_if_novalue:nF {#1}
3295     {
3296         \keys_set:nn { enumext / keyans* } {#1}
3297     }
3298 }

```

(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`.

```

3299 \cs_new_protected:Nn \__enumext_before_list_viii:
3300 {
3301     \__enumext_vspace_above_viii:
3302     \__enumext_before_args_exec_viii:
3303     \__enumext_start_mini_viii:
3304 }

```

(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.

```

3305 \cs_new_protected:Nn \__enumext_after_list_viii:
3306 {
3307
3308     \__enumext_stop_mini_viii:
3309     \__enumext_after_stop_list_viii:
3310     \__enumext_vspace_below_viii:
3311 }

```

(End of definition for `\__enumext_after_list_viii:`)

### 10.36.2 The command `\item` in keyans\*

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the optional argument (`\langle number \rangle`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\langle label \rangle` next to the `[ \langle content \rangle ]` if it is present in the `\langle sequence \rangle` and `\langle prop list \rangle` defined by `save-ans` key for `\item*`, `\item* [ \langle content \rangle ]`, `\item ( \langle number \rangle ) *` and `\item ( \langle number \rangle ) * [ \langle content \rangle ]` commands.

`\_enumext_start_item_tmp_viii:`

First we will call the function `\_enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l\_enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g\_enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\_enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
3312 \cs_new_protected_nopar:Nn \_enumext_start_item_tmp_viii:
3313 {
3314   \_enumext_stop_item_tmp_viii:
3315   \int_incr:N \l\_enumext_item_column_pos_viii_int
3316   \int_gincr:N \g\_enumext_item_count_all_viii_int
3317   \_enumext_item_peek_args_viii:
3318 }
```

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

`\_enumext_item_peek_args_viii:`

The function `\_enumext_item_peek_args_viii:` will handle the `\item ( \langle number \rangle )`. Look for the argument “(”, if it is present we will call the function `\_enumext_joined_item_viii:w ( \langle number \rangle )`, which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```
3319 \cs_new_protected:Nn \_enumext_item_peek_args_viii:
3320 {
3321   \peek_meaning:NTF (
3322     { \_enumext_joined_item_viii:w }
3323     { \_enumext_joined_item_viii:w (1) }
3324 }
```

(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`.

```
3325 \cs_new_protected:Npn \_enumext_joined_item_viii:w (#1)
3326 {
3327   \_enumext_starred_joined_item_viii:n {#1}
3328   \peek_meaning_remove:NTF *
3329     { \_enumext_starred_item_viii:w }
3330     { \_enumext_standard_item_viii:w }
3331 }
```

(End of definition for `\_enumext_joined_item_viii:w`.)

`\_enumext_standard_item_viii:w`

The function `\_enumext_standard_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l\_enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l\_enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item [ \langle custom \rangle ]` by means of the function `\_enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l\_enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the *enumerated* version of `\item` by means of the function `\_enumext_start_item_viii:w [ \l\_enumext_label_viii_tl ]`.

```
3332 \cs_new_protected:Npn \_enumext_standard_item_viii:w
3333 {
3334   \bool_set_false:N \l\_enumext_item_starred_viii_bool
3335   \peek_meaning:NTF [
3336     {
3337       \bool_set_eq:NN
3338         \l\_enumext_wrap_label_viii_bool
3339         \l\_enumext_wrap_label_opt_viii_bool
3340       \_enumext_start_item_viii:w
3341     }
3342 }
```

```

3343         \bool_set_true:N \l__enumext_wrap_label_viii_bool
3344         \legacy_if_set_true:n { @noitemarg }
3345         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3346     }
3347 }

```

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

The function \\_\_enumext\_starred\_item\_viii:w together with the specified auxiliary functions aux\_i:w, aux\_ii:w, and aux\_iii:w execute \item\*, \item\* and \item\*[\langle content \rangle].

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
\__enumext_starred_item_viii_aux_iii:w
3348 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3349 {
3350     \bool_set_true:N \l__enumext_item_starred_viii_bool
3351     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3352     \peek_meaning:NTF [
3353     { \__enumext_starred_item_viii_aux_i:w }
3354     { \__enumext_starred_item_viii_aux_ii:w }
3355 }
3356 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3357 {
3358     %\tl_gset:Nn \g__enumext_item_symbol_aux_viii_tl {#1}
3359     \__enumext_starred_item_viii_aux_ii:w
3360 }
3361 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3362 {
3363     \peek_meaning:NTF [
3364     { \__enumext_starred_item_viii_aux_iii:w }
3365     {
3366         %%\dim_set_eq:NN
3367         %% \l__enumext_item_symbol_sep_viii_dim
3368         %% \l__enumext_labelsep_viii_dim
3369         \legacy_if_set_true:n { @noitemarg }
3370         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3371     }
3372 }
3373 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_iii:w [#1]
3374 {
3375     %%\dim_set:Nn \l__enumext_item_symbol_sep_viii_dim {#1}
3376     \legacy_if_set_true:n { @noitemarg }
3377     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3378 }

```

(End of definition for \\_\_enumext\_starred\_item\_viii:w and others.)

### Real definition of \item

The functions \\_\_enumext\_start\_item\_viii:w and \\_\_enumext\_stop\_item\_viii: executing the true definition of \item inside the `keyans*` environment.

\\_\_enumext\_start\_item\_viii:w The first thing we will do is set the value of \\_\_enumext\_stop\_item\_tmp\_viii: equal to the value of \\_\_enumext\_stop\_item\_viii: which we will define later and add the `hyperref` compatible `enumXviii` 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.

```

3379 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3380 {
3381     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3382     \legacy_if:nT { @noitemarg }
3383     {
3384         \legacy_if_set_false:n { @noitemarg }
3385         \legacy_if:nT { @nmbrrlist }
3386         {
3387             \bool_if:NT \l__enumext_hyperref_bool
3388             {
3389                 \legacy_if_set_true:n { @hyper@item }
3390             }
3391             \refstepcounter{enumXviii}
3392             % code for check-ans
3393             %\bool_if:NT \l__enumext_check_ans_bool
3394             %{
3395                 %% If true |no-store| key => nested in |enumext|

```

```

3396         %\bool_if:NTF \l__enumext_store_ans_bool
3397         %{
3398             %\int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3399             %{ \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3400         }%}
3401         %{
3402             %\int_gincr:N \g__enumext_count_item_all_int
3403             %\int_gincr:N \g__enumext_count_level_viii_int
3404         }%}
3405     }%}
3406 }
3407 }

```

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

```

3408 \group_begin:
3409 \lrbox{ \l__enumext_item_text_viii_box }
3410 \bool_if:NT \l__enumext_item_starred_viii_bool
3411 {
3412     %\tl_if_blank:VT \g__enumext_item_symbol_aux_viii_tl
3413     %{
3414         %\tl_gset_eq:NN
3415         %\g__enumext_item_symbol_aux_viii_tl \l__enumext_item_symbol_viii_tl
3416     }%}
3417     \mode_leave_vertical:
3418     %%\skip_horizontal:n { -\l__enumext_item_symbol_sep_viii_dim }
3419     %\makebox[0pt][r]{ \g__enumext_item_symbol_aux_viii_tl }
3420     %%\skip_horizontal:N \l__enumext_item_symbol_sep_viii_dim
3421     %\tl_gclear:N \g__enumext_item_symbol_aux_viii_tl
3422 }
3423 \group_begin:
3424 \tl_use:N \l__enumext_label_font_style_viii_tl
3425 \bool_if:NTF \l__enumext_wrap_label_viii_bool
3426 {
3427     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3428     { \__enumext_wrapper_label_viii:n {#1} }
3429 }
3430 {
3431     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3432 }
3433 \group_end:
3434 \skip_horizontal:N \l__enumext_labelsep_viii_dim
3435 \tl_use:N \l__enumext_after_list_args_viii_tl
3436 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3437 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3438 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3439 \tl_use:N \l__enumext_fake_item_indent_viii_tl
3440 }

```

(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.

```

3441 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3442 {
3443     \__enumext_endminipage:
3444     \endlrbox
3445     \group_end:
3446     \box_set_wd:Nn \l__enumext_item_text_viii_box
3447     {
3448         \l__enumext_joined_width_viii_dim
3449         + \l__enumext_labelwidth_viii_dim
3450         + \l__enumext_labelsep_viii_dim
3451     }
3452     \int_set:Nn \hbadness { 10000 }
3453     \box_use:N \l__enumext_item_text_viii_box
3454     \bool_if:NF \l__enumext_footnotes_key_bool
3455     {
3456         \__enumext_print_footnote:
3457     }
3458     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }

```

```

3459     {
3460         \par\noindent
3461         \int_zero:N \l__enumext_item_column_pos_viii_int
3462     }
3463     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3464 }

```

(End of definition for \l\_\_enumext\_stop\_item\_viii:.)

\l\_\_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.

```

3465 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_viii:
3466 {
3467     \int_compare:nNnT
3468     {
3469         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3470     }
3471     =
3472     { \c_zero_int }
3473     {
3474         \par
3475         \vspace{ -\l__enumext_itemsep_viii_skip }
3476         \int_gzero:N \g__enumext_item_count_all_viii_int
3477     }
3478 }

```

(End of definition for \l\_\_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.

```

3479 \NewDocumentCommand \getkeyans { m }
3480 {
3481     \exp_args:Ne \l__enumext_getkeyans_aux:n
3482     { \tl_to_str:e { \text_expand:n {#1} } }
3483 }

```

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

\l\_\_enumext\_getkeyans\_aux:n

The internal function `\l__enumext_getkeyans_aux:n` is in charge of *splitting* the `⟨argument⟩` using “:”. If “:” is omitted it will return an error.

```

3484 \cs_new_protected:Npn \l__enumext_getkeyans_aux:n #1
3485 {
3486     \str_if_in:nnTF {#1} { : }
3487     {
3488         \use:e
3489         {
3490             \cs_set:Npn \exp_not:N \l__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3491             { {##1} {##2} }
3492         }
3493         \exp_after:wN \l__enumext_getkeyans:nn \l__enumext_tmp:w #1 \scan_stop:
3494     }
3495     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3496 }

```

(End of definition for \l\_\_enumext\_getkeyans\_aux:n.)

\l\_\_enumext\_getkeyans:nn

The internal function `\l__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⟩`.

```

3497 \cs_new_protected:Npn \l__enumext_getkeyans:nn #1 #2
3498 {
3499     \prop_if_exist:cF { g__enumext_#1_prop }
3500     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3501     \group_begin:
3502     \prop_item:cn { g__enumext_#1_prop }{#2}
3503     \group_end:
3504 }

```

(End of definition for \l\_\_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`.

```

3505 \keys_define:nn { keyanskey / print }
3506 {
3507   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3508                   {
3509                     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3510                   },
3511   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3512   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3513                   {
3514                     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3515                   },
3516   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3517   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3518                   {
3519                     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3520                   },
3521   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3522   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3523                   {
3524                     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3525                   },
3526   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3527   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3528                   {
3529                     \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3530                   },
3531   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3532 }

```

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

```

3533 \NewDocumentCommand \printkeyans { s O{} m }
3534 {
3535   \group_begin:
3536     \tl_use:N \l__enumext_print_keyans_i_tl
3537     \tl_use:N \l__enumext_print_keyans_ii_tl
3538     \tl_use:N \l__enumext_print_keyans_iii_tl
3539     \tl_use:N \l__enumext_print_keyans_iv_tl
3540     \tl_use:N \l__enumext_print_keyans_vii_tl
3541     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3542   \group_end:
3543 }

```

(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

```

3544 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3545 {
3546   \seq_if_exist:cTF { g__enumext_#3_seq }
3547   {
3548     \seq_if_empty:cF { g__enumext_#3_seq }
3549     {
3550       \bool_if:nTF {#1}
3551       {
3552         \begin{enumext*}[#2]
3553         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3554         \end{enumext*}
3555       }
3556       {
3557         \begin{enumext}[#2]

```



```

3558         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3559     \end{enumext}
3560 }
3561 }
3562 }
3563 {
3564     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3565 }
3566 }

```

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

### 10.39 The command `\setenumext`

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

```

3567 \keys_define:nn { enumext / meta-families }
3568 {
3569     level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3570     level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
3571     level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
3572     level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
3573     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },
3574     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} },
3575     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
3576     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } },
3577     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } },
3578     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } },
3579     print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } },
3580     print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } },
3581     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
3582 }

```

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

```

3583 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3584 {
3585     level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3586     keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3587 }

```

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

```

3588 \NewDocumentCommand \setenumext { o +m }
3589 {
3590     \tl_if_novalue:nTF {#1}
3591     {
3592         \seq_map_inline:Nn \c__enumext_all_families_seq
3593     }
3594     {
3595         \seq_clear:N \l__enumext_setkey_tmpa_seq
3596         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3597         \int_set:Nn \l__enumext_setkey_tmpa_int
3598         {
3599             \seq_count:N \l__enumext_setkey_tmpb_seq
3600         }
3601         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3602         {
3603             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3604             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
3605             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3606             {
3607                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3608             }
3609         }
3610         {
3611             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3612         }
3613         \seq_if_empty:NnTF \l__enumext_setkey_tmpa_seq
3614         { \seq_map_inline:Nn \c__enumext_all_families_seq }
3615         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3616     }
3617     {
3618         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3619     }
3620 }

```

(End of definition for `\setenumext`. This function is documented on page 5.)

```

__enumext_set_parse:n Internal functions used by the \setenumext command.
__enumext_set_error:nn
3621 \cs_new_protected:Npn __enumext_set_parse:n #1
3622 {
3623   \tl_set:Nc \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3624   \int_step_inline:nnn { 0 } { 4 } % <- max level
3625   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3626   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3627   {
3628     \seq_put_right:Nc \l__enumext_setkey_tmpa_seq
3629     { \tl_trim_spaces:n {#1} }
3630   }
3631   { __enumext_set_error:nn {#1} { } }
3632 }
3633 \cs_new_protected:Npn __enumext_set_error:nn #1 #2
3634 { \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.

```

3635 \msg_new:nnn { enumext } { package-load }
3636 {
3637   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3638 }
3639 \msg_new:nnn { enumext } { package-not-load }
3640 {
3641   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3642 }
3643 \msg_new:nnn { enumext } { package-load-foot }
3644 {
3645   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3646 }

```

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

```

3647 \msg_new:nnn { enumext } { counters }
3648 {
3649   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3650   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3651 }

```

Message used by `[(key = val)]` system and `\setenumext` command.

```

3652 \msg_new:nnn { enumext } { invalid-key }
3653 {
3654   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3655 }
3656 \msg_new:nnn { enumext } { unknown-key-family }
3657 {
3658   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3659 }

```

Messages used in length calculation.

```

3660 \msg_new:nnn { enumext } { width-negative }
3661 {
3662   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3663   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
3664 }
3665 \msg_new:nnn { enumext } { width-zero }
3666 {
3667   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3668   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
3669 }

```

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

```

3670 \msg_new:nnn { enumext } { list-lengths }
3671 {
3672   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3673   __enumext_show_length:nnn { dim } { labelsep } {#1}
3674   __enumext_show_length:nnn { dim } { labelwidth } {#1}
3675   __enumext_show_length:nnn { dim } { itemindent } {#1}

```

```

3676 \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3677 \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3678 \__enumext_show_length:nnn { dim } { listparindent } {#1}
3679 \__enumext_show_length:nnn { skip } { topsep } {#1}
3680 \__enumext_show_length:nnn { skip } { parsep } {#1}
3681 \__enumext_show_length:nnn { skip } { partopsep } {#1}
3682 \__enumext_show_length:nnn { skip } { itemsep } {#1}
3683 *****
3684 }

```

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

```

3685 \msg_new:nnn { enumext } { list-lengths-not-nested }
3686 {
3687   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
3688   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3689   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3690   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3691   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3692   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3693   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3694   \__enumext_show_length:nnn { skip } { topsep } {#1}
3695   \__enumext_show_length:nnn { skip } { parsep } {#1}
3696   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3697   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3698   *****
3699 }

```

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

```

3700 \msg_new:nnn { enumext } { items-same-answer }
3701 {
3702   *****~Checking~answers~on~'#1'~OK~*****\\
3703   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3704   *****
3705   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3706 }
3707 \msg_new:nnn { enumext } { item-different-answer }
3708 {
3709   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3710   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3711 }

```

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

```

3712 \msg_new:nnn { enumext } { missing-starred }
3713 {
3714   Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3715 }

```

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

```

3716 \msg_new:nnn { enumext } { list-too-deep }
3717 {
3718   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
3719   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3720 }

```

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

```

3721 \msg_new:nnn { enumext } { anskey-wrong-place }
3722 {
3723   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3724   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3725 }
3726 \msg_new:nnn { enumext } { anspic-wrong-place }
3727 {
3728   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3729   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3730 }
3731 \msg_new:nnn { enumext } { command-wrong-place }
3732 {
3733   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3734   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3735 }

```

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

```

3736 \msg_new:nnn { enumext } { keyans-nested }
3737 {

```

```

3738     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3739 }
3740 \msg_new:nnn { enumext } { keyans-wrong-level }
3741 {
3742     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3743     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3744 }
3745 \msg_new:nnn { enumext } { wrong-place }
3746 {
3747     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3748     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3749 }
3750 \msg_new:nnn { enumext } { keyanspic-nested }
3751 {
3752     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
3753 }
3754 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3755 {
3756     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
3757     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3758 }

```

Messages used by `\getkeyans` command.

```

3759 \msg_new:nnn { enumext } { undefined-storage-anskey }
3760 {
3761     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3762 }

```

Messages used by `\miniright` command.

```

3763 \msg_new:nnn { enumext } { missing-miniright }
3764 {
3765     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
3766     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3767 }
3768 \msg_new:nnn { enumext } { wrong-miniright-place }
3769 {
3770     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3771     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3772 }
3773 \msg_new:nnn { enumext } { wrong-miniright-use }
3774 {
3775     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3776     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3777 }

```

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

```

3778 \msg_new:nnn { enumext } { nested }
3779 {
3780     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3781 }
3782 \msg_new:nnn { enumext } { item-joined }
3783 {
3784     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3785 }
3786 \msg_new:nnn { enumext } { item-joined-columns }
3787 {
3788     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3789 }

```

## 10.41 Finish package

Finish package implementation.

```

3790 \file_input_stop:
3791 </package>

```



enumXv . . . . .	22, 29	\c_zero_dim	632, 646, 658, 670, 1170, 1188, 1717, 2167, 2172, 2178, 2185, 2385, 2408, 2533, 2551, 2719, 2787, 3144, 3212
cs commands:		<b>E</b>	
\cs_generate_variant:Nn	307, 323, 522, 538, 1551, 1558, 1638, 2196, 2700	\end . .	1173, 1191, 1587, 1622, 2437, 2461, 2576, 2593, 2808, 2824, 3233, 3249, 3554, 3559
\cs_if_exist:NTF . . . . .	277	\endlist . . . . .	27
\cs_new:Nn . . . . .	186	\endlist . . . . .	214
\cs_new:Npn . . . . .	190, 195, 205	\endlrbox . . . . .	3095, 3444
\cs_new_eq:NN	213, 214, 215, 219, 220, 252, 253, 256, 257	\endminipage . . . . .	27
\cs_new_protected:Nn .	224, 388, 408, 440, 703, 707, 711, 715, 719, 723, 727, 731, 735, 739, 743, 747, 751, 755, 759, 763, 799, 811, 835, 852, 863, 887, 962, 986, 1003, 1065, 1082, 1104, 1139, 1145, 1220, 1234, 1248, 1259, 1270, 1281, 1292, 1303, 1343, 1354, 1415, 1427, 1448, 1559, 1583, 1590, 1618, 1625, 1742, 1869, 1999, 2003, 2022, 2075, 2109, 2125, 2135, 2151, 2301, 2354, 2372, 2379, 2402, 2432, 2451, 2506, 2529, 2547, 2572, 2583, 2620, 2663, 2676, 2696, 2701, 2717, 2785, 2804, 2855, 2912, 2918, 2939, 2949, 2966, 3116, 3142, 3210, 3229, 3278, 3299, 3305, 3319, 3465	\endminipage . . . . .	220
\cs_new_protected:Npn	178, 182, 260, 275, 292, 302, 308, 396, 415, 509, 523, 1167, 1186, 1330, 1383, 1542, 1552, 1674, 1819, 1831, 1853, 1904, 1938, 1946, 2032, 2051, 2086, 2098, 2165, 2199, 2245, 2316, 2325, 2525, 2671, 2736, 2872, 2883, 2972, 2979, 2995, 3003, 3008, 3020, 3161, 3292, 3325, 3332, 3348, 3356, 3361, 3373, 3484, 3497, 3544, 3621, 3633	enumext . . . . .	4, <u>2282</u>
\cs_new_protected_nopar:Nn . . .	2959, 3092, 3312, 3441	enumext internal commands:	
\cs_new_protected_nopar:Npn . . . . .	3026, 3379	__enumext_add_pre_parsep: . . .	40, 809, <u>811</u> , 811
\cs_set:Nn . . . . .	193, 1824	__enumext_after_args_exec: .	37, <u>703</u> , 715, 2294
\cs_set:Npn . . . . .	1752, 1790, 3490	__enumext_after_args_exec_v: <u>38</u> , <u>719</u> , 731, 2499	
\cs_set_eq:NN	192, 197, 2845, 2846, 3028, 3269, 3270, 3381	__enumext_after_args_exec_vii: . . .	<u>735</u> , 759
\cs_set_protected:Nn . . . . .	627, 643, 655, 667	__enumext_after_args_exec_viii: . . . . .	763
\cs_set_protected:Npn .	30, 44, 52, 64, 70, 94, 123, 134, 146, 153, 324, 346, 375, 456, 476, 539, 559, 603, 622, 679, 688, 767, 784, 1203, 1365, 1470, 1499, 1517, 1744, 1871, 1983, 2197, 2243	__enumext_after_env:n . . . . .	74
\cs_to_str:N . . . . .	294, 317	__enumext_after_env:nn . .	<u>74</u> , <u>88</u> , <u>182</u> , 182, 2476, 2813, 3130, 3238
<b>D</b>		__enumext_after_hyperref: . . .	27, 222, 224, 224
\d . . . . .	201	__enumext_after_list: <u>74</u> , <u>84</u> , <u>91</u> , 2299, <u>2451</u> , 2451	
\DeclareDocumentEnvironment . . . . .	880	\l_enumext_after_list_args_v_tl . . . . .	733
dim commands:		\l_enumext_after_list_args_vii_tl	761, 3086
\dim_abs:n . . . . .	2170, 2175	\l_enumext_after_list_args_viii_tl	765, 3435
\dim_add:Nn . . . . .	2681	__enumext_after_list_v: . .	<u>77</u> , 2504, <u>2583</u> , 2583
\dim_compare:nNnTF .	629, 645, 657, 669, 1169, 1188, 2167, 2172, 2178, 2184, 2186, 2188, 2384, 2407, 2533, 2551, 2673, 2719, 2787, 3144, 3212	__enumext_after_list_vii: . . .	2853, <u>2918</u> , 2918
\dim_compare:nTF . . . . .	1715	__enumext_after_list_viii: . .	3276, <u>3305</u> , 3305
\dim_gset_eq:NN . . . . .	2796, 3221	__enumext_after_star_env:nn . . . . .	82
\dim_gzero:N . . . . .	2830, 3255	__enumext_after_stop_list: . . .	<u>37</u> , <u>38</u> , <u>703</u> , 711, 2465
\dim_new:N .	40, 47, 48, 49, 66, 101, 111, 162, 163, 169	__enumext_after_stop_list_v: <u>38</u> , <u>719</u> , 727, 2598	
\dim_set:Nn . .	305, 617, 1529, 2065, 2170, 2175, 2177, 2180, 2181, 2185, 2187, 2190, 2191, 2193, 2387, 2410, 2535, 2553, 2703, 2721, 2728, 2771, 2789, 3022, 3146, 3153, 3196, 3214, 3375	\l_enumext_after_stop_list_v_tl . . . . .	729
\dim_set_eq:NN	463, 483, 499, 503, 2060, 2208, 2254, 2344, 2418, 2561, 2778, 2781, 2782, 2902, 3013, 3203, 3206, 3207, 3366	__enumext_after_stop_list_vii: <u>735</u> , 751, 2922	
\dim_use:N	630, 638, 1170, 1176, 1628, 1631, 1636, 2130, 2132, 2385, 2390, 2391, 2398, 2408, 2412, 2413, 2415	\l_enumext_after_stop_list_viii: . . .	753
\dim_zero:N . . . . .	2422, 2563, 2682, 2683, 2684	__enumext_after_stop_list_viii_tl . . .	757
\dim_zero_new:N . . . . .	2734, 3159	\l_enumext_align_label_vii_str . .	3078, 3082
		\l_enumext_align_label_viii_str .	3427, 3431
		\l_enumext_align_label_X_str . . . . .	153
		\c__enumext_all_envs_clist . .	<u>173</u> , 345, 558, 621, 687, 702, 783, 1219
		\c__enumext_all_families_seq . .	97, 3583, 3592, 3614
		__enumext_anskey_wrapper:n . . . . .	1474, 1829
		__enumext_at_begin_document:n . .	27, <u>178</u> , 178, 211, 217
		__enumext_before_args_exec: <u>37</u> , <u>703</u> , 703, 2382	
		__enumext_before_args_exec_v: <u>37</u> , <u>38</u> , <u>719</u> , 719, 2532	
		__enumext_before_args_exec_vii: . .	<u>735</u> , 735, 2915
		__enumext_before_args_exec_viii: .	739, 3302
		__enumext_before_keys_exec: <u>37</u> , <u>703</u> , 707, 2292	
		__enumext_before_keys_exec_v: . .	<u>38</u> , <u>719</u> , 723, 2497
		__enumext_before_keys_exec_vii . . . . .	<u>735</u>
		__enumext_before_keys_exec_vii: <u>38</u> , 743, 2841	
		__enumext_before_keys_exec_viii: . .	<u>38</u> , 747, 3265
		__enumext_before_list: . . .	72, 2286, <u>2379</u> , 2379
		__enumext_before_list_v: .	75, 2492, <u>2529</u> , 2529



`\__enumext_before_list_vii:` [84](#), [2836](#), [2912](#), [2912](#)  
`\__enumext_before_list_viii:` [..](#) [91](#), [3261](#), [3299](#), [3299](#)  
`\l__enumext_before_no_starred_key_v_tl` [725](#)  
`\l__enumext_before_no_starred_key_vii_-tl` [.....](#) [745](#)  
`\l__enumext_before_no_starred_key_viii_-tl` [.....](#) [749](#)  
`\l__enumext_before_starred_key_v_tl ...` [721](#)  
`\l__enumext_before_starred_key_vii_tl` [..](#) [737](#)  
`\l__enumext_before_starred_key_viii_tl` [741](#)  
`\__enumext_calc_hspace:NNNNNN` [68](#), [2165](#), [2165](#), [2196](#), [2201](#), [2247](#)  
`\__enumext_check_ans_active:` [..](#) [53](#), [1427](#), [1427](#), [2482](#)  
`\__enumext_check_ans_active_vii:` [..](#) [1427](#), [1448](#), [3136](#)  
`\l__enumext_check_ans_bool` [50](#), [64](#), [65](#), [115](#), [1338](#), [1369](#), [1373](#), [1417](#), [1666](#), [1933](#), [2036](#), [2067](#), [2444](#), [2931](#), [3040](#), [3393](#)  
`\__enumext_check_ans_count:` [..](#) [52](#), [72](#), [1415](#), [1415](#), [2383](#)  
`\__enumext_check_ans_int:n` [..](#) [50](#), [52](#), [1340](#), [1383](#), [1383](#)  
`\g__enumext_check_ans_item_tl` [..](#) [62](#), [115](#), [1932](#), [1940](#), [1944](#)  
`\g__enumext_check_ans_show_bool` [74](#), [115](#), [2447](#), [2478](#), [2485](#)  
`\g__enumext_check_ans_show_h_bool` [115](#), [2933](#), [3132](#), [3139](#)  
`\l__enumext_columns_sep_v_dim` [2551](#), [2553](#), [2561](#)  
`\l__enumext_columns_sep_vii_dim` [..](#) [2719](#), [2721](#), [2730](#), [2775](#), [2904](#), [3114](#)  
`\l__enumext_columns_sep_viii_dim` [..](#) [3144](#), [3146](#), [3155](#), [3200](#), [3463](#)  
`\l__enumext_columns_v_int` [1008](#), [2549](#), [2557](#), [2569](#), [2574](#)  
`\l__enumext_columns_vii_int` [..](#) [2724](#), [2727](#), [2731](#), [2739](#), [2743](#), [2746](#), [2752](#), [2758](#), [2762](#), [2891](#), [3109](#), [3120](#)  
`\l__enumext_columns_viii_int` [..](#) [3149](#), [3152](#), [3156](#), [3164](#), [3168](#), [3171](#), [3177](#), [3183](#), [3187](#), [3458](#), [3469](#)  
`\l__enumext_compare_items_ans_int` [115](#), [1429](#), [1435](#), [1450](#), [1457](#)  
`\g__enumext_count_item_all_int` [115](#), [1403](#), [1406](#), [1431](#), [1452](#), [2038](#), [2069](#), [3049](#), [3402](#)  
`\g__enumext_count_item_i_int` [.....](#) [1408](#), [1452](#)  
`\g__enumext_count_item_ii_int` [1409](#), [1431](#), [1453](#)  
`\g__enumext_count_item_iii_int` [1410](#), [1432](#), [1453](#)  
`\g__enumext_count_item_iv_int` [1411](#), [1432](#), [1454](#)  
`\g__enumext_count_item_vii_int` [.....](#) [1412](#)  
`\g__enumext_count_item_with_ans_int` [52](#), [57](#), [62](#), [115](#), [1413](#), [1435](#), [1457](#), [1668](#), [1935](#)  
`\g__enumext_count_item_X_int` [.....](#) [115](#)  
`\g__enumext_count_level_i_int` [....](#) [1442](#), [1464](#)  
`\g__enumext_count_level_ii_int` [..](#) [1443](#), [1465](#)  
`\g__enumext_count_level_iii_int` [..](#) [1444](#), [1466](#)  
`\g__enumext_count_level_iv_int` [..](#) [1445](#), [1467](#)  
`\g__enumext_count_level_vii_int` [..](#) [1446](#), [1468](#), [3050](#)  
`\g__enumext_count_level_viii_int` [.....](#) [3403](#)  
`\g__enumext_count_level_X_int` [.....](#) [115](#)  
`\l__enumext_counter_i_tl` [.....](#) [30](#), [284](#)  
`\l__enumext_counter_ii_tl` [.....](#) [30](#), [285](#)  
`\l__enumext_counter_iii_tl` [.....](#) [30](#), [286](#)  
`\l__enumext_counter_iv_tl` [.....](#) [30](#), [287](#)  
`\l__enumext_counter_style_for_ref_vii_-tl` [.....](#) [423](#), [433](#), [444](#), [446](#)  
`\l__enumext_counter_style_for_ref_viii_-tl` [.....](#) [450](#), [452](#)  
`\l__enumext_counter_style_for_ref_X_tl` [142](#)  
`\c__enumext_counter_style_tl` [....](#) [31](#), [142](#), [390](#)  
`\g__enumext_counter_styles_tl` [..](#) [22](#), [29](#), [40](#), [295](#), [313](#)  
`\l__enumext_counter_v_tl` [.....](#) [30](#), [288](#)  
`\l__enumext_counter_vi_tl` [.....](#) [30](#), [289](#)  
`\l__enumext_counter_vii_tl` [.....](#) [30](#), [290](#), [420](#)  
`\l__enumext_counter_viii_tl` [.....](#) [30](#), [291](#), [430](#)  
`\l__enumext_current_widest_dim` [22](#), [40](#), [319](#), [464](#), [484](#), [500](#), [504](#)  
`\__enumext_default_item:n` [....](#) [2032](#), [2032](#), [2083](#)  
`\__enumext_define_counters:Nn` [22](#), [275](#), [275](#), [284](#), [285](#), [286](#), [287](#), [288](#), [289](#), [290](#), [291](#)  
`\__enumext_endminipage:` [..](#) [27](#), [217](#), [220](#), [886](#), [2713](#), [3094](#), [3443](#)  
`\__enumext_fake_item:` [.....](#) [627](#), [627](#), [2234](#)  
`\l__enumext_fake_item_indent_v_dim` [646](#), [651](#)  
`\l__enumext_fake_item_indent_v_tl` [648](#), [2091](#), [2095](#), [2103](#)  
`\l__enumext_fake_item_indent_vii_dim` [658](#), [663](#)  
`\l__enumext_fake_item_indent_vii_tl` [660](#), [3090](#)  
`\l__enumext_fake_item_indent_viii_dim` [..](#) [670](#), [675](#)  
`\l__enumext_fake_item_indent_viii_tl` [..](#) [672](#), [3439](#)  
`\l__enumext_fake_item_indent_X_tl` [.....](#) [70](#)  
`\__enumext_fake_item_vii:` [....](#) [627](#), [655](#), [2270](#)  
`\__enumext_fake_item_viii:` [....](#) [627](#), [667](#), [2275](#)  
`\g__enumext_footnote_arg_seq` [..](#) [139](#), [2005](#), [2018](#), [2028](#)  
`\g__enumext_footnote_int` [..](#) [139](#), [2012](#), [2015](#), [2017](#), [2019](#)  
`\g__enumext_footnote_int_seq` [..](#) [139](#), [2006](#), [2019](#), [2024](#), [2027](#)  
`\__enumext_footnotes_key_bool` [.....](#) [27](#)  
`\l__enumext_footnotes_key_bool` [24](#), [27](#), [87](#), [129](#), [233](#), [238](#), [247](#), [3057](#), [3105](#), [3454](#)  
`\__enumext_footnotetext:nn` [....](#) [1999](#), [1999](#), [2029](#)  
`\__enumext_getkeyans:nn` [....](#) [95](#), [3493](#), [3497](#), [3497](#)  
`\__enumext_getkeyans_aux:n` [..](#) [95](#), [3481](#), [3484](#), [3484](#)  
`\l__enumext_hyperref_bool` [..](#) [24](#), [27](#), [28](#), [129](#), [229](#), [250](#), [267](#), [1732](#), [1920](#), [3034](#), [3387](#)  
`\__enumext_hypertarget:nn` [28](#), [224](#), [252](#), [256](#), [272](#)  
`\__enumext_if_is_int:n` [.....](#) [199](#)  
`\__enumext_if_is_int:nTF` [.....](#) [199](#), [511](#), [525](#)  
`\l__enumext_item_column_pos_vii_int` [85](#), [2746](#), [2752](#), [2758](#), [2762](#), [2769](#), [2962](#), [3109](#), [3112](#)  
`\l__enumext_item_column_pos_viii_int` [....](#) [92](#), [3171](#), [3177](#), [3183](#), [3187](#), [3194](#), [3315](#), [3458](#), [3461](#)  
`\l__enumext_item_column_pos_X_int` [.....](#) [153](#)  
`\g__enumext_item_count_all_vii_int` [85](#), [2770](#), [2963](#), [3120](#), [3127](#)  
`\g__enumext_item_count_all_viii_int` [92](#), [3195](#), [3316](#), [3469](#), [3476](#)  
`\g__enumext_item_count_all_X_int` [.....](#) [153](#)  
`\__enumext_item_peek_args_vii:` [85](#), [2964](#), [2966](#), [2966](#)  
`\__enumext_item_peek_args_viii:` [92](#), [3317](#), [3319](#), [3319](#)  
`\__enumext_item_starred:` [..](#) [67](#), [2125](#), [2125](#), [2143](#)

`\l__enumext_item_starred_vii_bool` 2981, 2997, 3061  
`\l__enumext_item_starred_viii_bool` 3334, 3350, 3410  
`\l__enumext_item_starred_X_bool` ..... 153  
`\__enumext_item_std:w` 27, 64–66, 80, 211, 215, 2042, 2048, 2073, 2091, 2095, 2103, 2694  
`\g__enumext_item_symbol_aux_vii_tl` 3005, 3063, 3066, 3070, 3072  
`\g__enumext_item_symbol_aux_viii_tl` .. 3358, 3412, 3415, 3419, 3421  
`\g__enumext_item_symbol_aux_X_tl` ..... 153  
`\l__enumext_item_symbol_sep_vii_dim` .. 3014, 3022, 3069, 3071  
`\l__enumext_item_symbol_sep_viii_dim` . 3367, 3375, 3418, 3420  
`\g__enumext_item_symbol_tl` 22, 65, 35, 2057, 2131, 2148  
`\l__enumext_item_symbol_vii_tl` ..... 3066  
`\l__enumext_item_symbol_viii_tl` ..... 3415  
`\l__enumext_item_text_vii_box` 3056, 3097, 3104  
`\l__enumext_item_text_viii_box` 3409, 3446, 3453  
`\l__enumext_item_text_X_box` ..... 153  
`\l__enumext_item_width_vii_dim` ... 2728, 2773, 2781, 2782  
`\l__enumext_item_width_viii_dim` .. 3153, 3198, 3206, 3207  
`\l__enumext_item_width_X_dim` ..... 153  
`\l__enumext_itemindent_X_dim` ..... 44  
`\l__enumext_itemsep_vii_skip` ..... 3126  
`\l__enumext_itemsep_viii_skip` ..... 3475  
`\l__enumext_joined_item_aux_vii_int` .. 2767, 2768, 2769, 2770, 2776  
`\l__enumext_joined_item_aux_viii_int` . 3192, 3193, 3194, 3195, 3201  
`\l__enumext_joined_item_aux_X_int` .... 153  
`\__enumext_joined_item_vii:w` .. 85, 2969, 2970, 2972, 2972  
`\l__enumext_joined_item_vii_int` .. 2738, 2739, 2742, 2744, 2750, 2755, 2760, 2765, 2767, 2773  
`\__enumext_joined_item_viii:w` . 92, 3322, 3323, 3325, 3325  
`\l__enumext_joined_item_viii_int` . 3163, 3164, 3167, 3169, 3175, 3180, 3185, 3190, 3192, 3198  
`\l__enumext_joined_item_X_int` ..... 153  
`\l__enumext_joined_width_vii_dim` . 2771, 2778, 2781, 3087, 3099  
`\l__enumext_joined_width_viii_dim` 3196, 3203, 3206, 3436, 3448  
`\l__enumext_joined_width_X_dim` ..... 153  
`\__enumext_keyans_addto_prop:n` 61, 1853, 1853, 2105, 2626  
`\__enumext_keyans_addto_seq:n` . 62, 1904, 1904, 2107, 2628  
`\__enumext_keyans_anspic_code:nnn` . 78, 2617, 2620, 2620  
`\__enumext_keyans_check_ans:nn` .. 62, 63, 1938, 1938, 2502, 2658  
`\__enumext_keyans_default_item:n` .. 66, 2086, 2086, 2121  
`\l__enumext_keyans_env_bool` 20, 2358, 2513, 2597  
`\__enumext_keyans_fake_item:` .. 627, 643, 2224  
`\__enumext_keyans_internal_ref:` 61, 1869, 1869, 2106, 2627  
`\l__enumext_keyans_level_h_int` 20, 3280, 3281  
`\l__enumext_keyans_level_int` .. 20, 1161, 1655, 1968, 2512, 2516, 2611  
`\__enumext_keyans_make_label:` 30, 67, 2151, 2151, 2223  
`\__enumext_keyans_mini_addvspace:` 45, 76, 1065, 1065, 2541  
`\__enumext_keyans_mini_right_cmd:n` 47, 1163, 1186, 1186  
`\__enumext_keyans_mini_set_vskip:` . 44, 1003, 1003, 1067  
`\__enumext_keyans_multi_addvspace:` . 76, 852, 863, 2566  
`\__enumext_keyans_multi_set_vskip:` . 41, 852, 852, 865  
`\__enumext_keyans_multicols_start:` 76, 2545, 2547, 2547  
`\__enumext_keyans_multicols_stop:` . 77, 1190, 2572, 2572, 2596  
`\__enumext_keyans_parse_keys:n` 2491, 2525, 2525  
`\l__enumext_keyans_pic_above_int` . 110, 2704, 2705, 2707  
`\l__enumext_keyans_pic_above_skip` .. 80, 110, 2649, 2688  
`\__enumext_keyans_pic_arg_two:` 79, 2647, 2676, 2676  
`\l__enumext_keyans_pic_below_int` . 110, 2704, 2705, 2708  
`\l__enumext_keyans_pic_body_seq` .. 78–80, 110, 2615, 2654, 2712  
`\__enumext_keyans_pic_do:n` 80, 2654, 2656, 2696, 2696, 2700  
`\l__enumext_keyans_pic_level_int` .. 20, 1153, 1659, 1856, 1879, 1907, 2665, 2666  
`\__enumext_keyans_pic_row:n` 80, 2698, 2701, 2701  
`\__enumext_keyans_pic_safe_exec:` .. 79, 2643, 2663, 2663  
`\__enumext_keyans_pic_skip_abs:N` .. 79, 2671, 2671, 2687  
`\l__enumext_keyans_pic_width_dim` . 110, 2703, 2710  
`\__enumext_keyans_redefine_item:` .. 66, 2109, 2109, 2222  
`\__enumext_keyans_safe_exec:` . 2490, 2506, 2506  
`\__enumext_keyans_show_left:n` . 66, 1946, 1946, 2101, 2634  
`\__enumext_keyans_starred_item:n` .. 66, 2098, 2098, 2117  
`\l__enumext_keyans_tmpa_tl` .. 23, 82, 2100, 2104  
`\l__enumext_label_copy_i_tl` .. 1786, 1882, 1886  
`\l__enumext_label_copy_v_tl` ..... 1886  
`\l__enumext_label_copy_vi_tl` ..... 1882  
`\l__enumext_label_copy_vii_tl` 1761, 1772, 1803  
`\l__enumext_label_copy_X_tl` ..... 131  
`\l__enumext_label_fill_left_v_tl` ..... 2155  
`\l__enumext_label_fill_left_X_tl` ..... 70  
`\l__enumext_label_fill_right_v_tl` .... 2162  
`\l__enumext_label_fill_right_X_tl` ..... 70  
`\l__enumext_label_font_style_v_tl` 2156, 2638  
`\l__enumext_label_font_style_vii_tl` ... 3075  
`\l__enumext_label_font_style_viii_tl` .. 3424  
`\l__enumext_label_i_tl` ..... 456  
`\l__enumext_label_ii_tl` ..... 456  
`\l__enumext_label_iii_tl` ..... 456



`\l__enumext_label_iv_tl` ..... [456](#)  
`\__enumext_label_style:Nnn` [22](#), [29](#), [308](#), [308](#), [323](#),  
[461](#), [481](#), [497](#), [501](#)  
`\l__enumext_label_v_tl` .. [61](#), [62](#), [494](#), [1861](#), [1912](#),  
[1950](#), [1957](#), [1973](#), [1980](#), [2100](#), [2104](#), [2494](#), [2633](#), [2635](#)  
`\l__enumext_label_vi_tl` . [61](#), [62](#), [494](#), [1858](#), [1909](#),  
[2633](#), [2635](#), [2639](#)  
`\l__enumext_label_vii_tl` . [476](#), [2992](#), [3017](#), [3024](#)  
`\l__enumext_label_viii_tl` [476](#), [3345](#), [3370](#), [3377](#)  
`\l__enumext_label_width_by_box` .. [40](#), [304](#), [305](#)  
`\__enumext_label_width_by_box:Nn` [29](#), [302](#), [302](#),  
[307](#), [319](#), [535](#)  
`\l__enumext_labelsep_i_dim` ..... [1954](#), [1977](#)  
`\l__enumext_labelsep_v_dim` ..... [2556](#)  
`\l__enumext_labelsep_vii_dim` . [2723](#), [2732](#), [2774](#),  
[3015](#), [3085](#), [3101](#)  
`\l__enumext_labelsep_viii_dim` [3148](#), [3157](#), [3199](#),  
[3368](#), [3434](#), [3450](#)  
`\l__enumext_labelwidth_i_dim` ..... [1953](#), [1976](#)  
`\l__enumext_labelwidth_v_dim` ..... [2556](#)  
`\l__enumext_labelwidth_vii_dim` ... [2723](#), [2731](#),  
[2774](#), [3078](#), [3082](#), [3100](#)  
`\l__enumext_labelwidth_viii_dim` .. [3148](#), [3156](#),  
[3199](#), [3427](#), [3431](#), [3449](#)  
`\l__enumext_leftmargin_tmp_v_bool` . [79](#), [2678](#)  
`\l__enumext_leftmargin_tmp_X_bool` ..... [44](#)  
`\l__enumext_leftmargin_tmp_X_dim` ..... [44](#)  
`\l__enumext_leftmargin_X_dim` ..... [44](#)  
`\__enumext_level:` [186](#), [186](#), [192](#), [193](#), [197](#), [399](#), [401](#),  
[402](#), [410](#), [412](#), [630](#), [634](#), [638](#), [705](#), [709](#), [713](#), [717](#), [801](#),  
[803](#), [805](#), [807](#), [840](#), [842](#), [844](#), [846](#), [850](#), [890](#), [893](#), [912](#),  
[921](#), [927](#), [932](#), [936](#), [947](#), [951](#), [952](#), [957](#), [993](#), [997](#), [1170](#),  
[1176](#), [1223](#), [1225](#), [1227](#), [1230](#), [1237](#), [1239](#), [1241](#), [1244](#),  
[1421](#), [1422](#), [1424](#), [1563](#), [1571](#), [1575](#), [1579](#), [1824](#), [1827](#),  
[1828](#), [2039](#), [2041](#), [2042](#), [2046](#), [2047](#), [2048](#), [2055](#), [2057](#),  
[2061](#), [2062](#), [2065](#), [2070](#), [2072](#), [2073](#), [2127](#), [2130](#), [2132](#),  
[2139](#), [2140](#), [2141](#), [2144](#), [2147](#), [2289](#), [2291](#), [2327](#), [2332](#),  
[2333](#), [2334](#), [2336](#), [2340](#), [2345](#), [2346](#), [2347](#), [2349](#), [2362](#),  
[2367](#), [2374](#), [2385](#), [2387](#), [2390](#), [2391](#), [2393](#), [2398](#), [2405](#),  
[2408](#), [2410](#), [2412](#), [2413](#), [2414](#), [2415](#), [2418](#), [2424](#), [2429](#),  
[2435](#), [2438](#), [2440](#), [2453](#), [3045](#), [3046](#), [3398](#), [3399](#)  
`\__enumext_level_` ..... [192](#)  
`\__enumext_level_end:n` ..... [190](#), [195](#)  
`\l__enumext_level_h_int` [20](#), [418](#), [442](#), [1780](#), [1797](#),  
[2310](#), [2360](#), [2857](#), [2858](#)  
`\l__enumext_level_int` [20](#), [188](#), [813](#), [964](#), [1157](#), [1400](#),  
[1757](#), [1767](#), [1773](#), [1779](#), [1787](#), [1795](#), [1802](#), [2237](#), [2303](#),  
[2304](#), [2319](#), [2365](#), [2420](#), [2480](#), [2520](#), [2607](#), [2866](#), [2943](#),  
[2953](#), [3134](#), [3287](#)  
`\__enumext_level_set:n` ..... [190](#), [190](#)  
`\__enumext_list_arg_two_i:` ..... [2197](#)  
`\__enumext_list_arg_two_ii:` ..... [2197](#)  
`\__enumext_list_arg_two_iii:` ..... [2197](#)  
`\__enumext_list_arg_two_iv:` ..... [2197](#)  
`\__enumext_list_arg_two_v:` . [66](#), [2197](#), [2496](#), [2679](#)  
`\__enumext_list_arg_two_vii:` ..... [2243](#), [2840](#)  
`\__enumext_list_arg_two_viii:` .... [2243](#), [3264](#)  
`\l__enumext_listoffset_v_dim` ..... [2558](#)  
`\l__enumext_listparindent_vii_dim` .... [3088](#)  
`\l__enumext_listparindent_viii_dim` ... [3437](#)  
`\__enumext_make_label:` [30](#), [64](#), [65](#), [67](#), [2135](#), [2135](#),  
[2232](#)  
`\l__enumext_mark_answer_sym_tl` ... [56](#), [63](#), [105](#),  
[1477](#), [1633](#), [1839](#), [1961](#)  
`\l__enumext_mark_position_str` [105](#), [1481](#), [1482](#),  
[1504](#), [1505](#), [1631](#)  
`\l__enumext_mark_ref_sym_tl` .. [105](#), [1491](#), [1737](#),  
[1928](#)  
`\__enumext_mini_addvspace:` [43](#), [73](#), [986](#), [986](#), [2395](#)  
`\__enumext_mini_addvspace_vii:` [46](#), [1139](#), [1139](#),  
[2799](#)  
`\__enumext_mini_addvspace_viii:` [46](#), [1139](#), [1145](#),  
[3224](#)  
`\__enumext_mini_env*` ..... [880](#)  
`\__enumext_mini_right_cmd:n` [47](#), [1165](#), [1167](#), [1167](#)  
`\__enumext_mini_set_vskip:` ... [42](#), [887](#), [887](#), [988](#)  
`\__enumext_mini_set_vskip_vii:` [45](#), [1082](#), [1082](#),  
[1141](#)  
`\__enumext_mini_set_vskip_viii:` [45](#), [1082](#), [1104](#),  
[1147](#)  
`\__enumext_minipage:w` [27](#), [217](#), [219](#), [882](#), [2710](#), [3087](#),  
[3436](#)  
`\l__enumext_minipage_active_v_bool` ... [76](#), [77](#),  
[2539](#), [2564](#), [2577](#), [2585](#)  
`\g__enumext_minipage_active_vii_bool` ... [82](#),  
[2810](#), [2815](#), [2827](#)  
`\l__enumext_minipage_active_vii_bool` . [2795](#),  
[2806](#)  
`\g__enumext_minipage_active_viii_bool` [3235](#),  
[3240](#), [3252](#)  
`\l__enumext_minipage_active_viii_bool` [3220](#),  
[3231](#)  
`\g__enumext_minipage_active_X_bool` ... [153](#)  
`\l__enumext_minipage_active_X_bool` .... [58](#)  
`\g__enumext_minipage_after_skip` [58](#), [1086](#), [1098](#),  
[2825](#), [3250](#)  
`\l__enumext_minipage_after_skip` [42](#), [43](#), [74](#), [77](#),  
[58](#), [903](#), [918](#), [938](#), [954](#), [969](#), [975](#), [981](#), [995](#), [1005](#), [1014](#),  
[1017](#), [1029](#), [1047](#), [1058](#), [1074](#), [1106](#), [1119](#), [1133](#), [2462](#),  
[2594](#)  
`\g__enumext_minipage_center_vii_bool` . [2819](#),  
[2828](#)  
`\g__enumext_minipage_center_viii_bool` [3244](#),  
[3253](#)  
`\g__enumext_minipage_center_X_bool` ... [153](#)  
`\l__enumext_minipage_hsep_v_dim` ... [76](#), [2537](#)  
`\l__enumext_minipage_hsep_vii_dim` .... [2793](#)  
`\l__enumext_minipage_hsep_viii_dim` ... [3218](#)  
`\l__enumext_minipage_left_skip` [42](#), [76](#), [58](#), [895](#),  
[910](#), [929](#), [944](#), [991](#), [1001](#), [1006](#), [1012](#), [1021](#), [1038](#), [1050](#),  
[1070](#), [1080](#), [1084](#), [1089](#), [1093](#), [1107](#), [1111](#), [1125](#), [1143](#),  
[1149](#)  
`\l__enumext_minipage_left_v_dim` [76](#), [2535](#), [2543](#)  
`\l__enumext_minipage_left_vii_dim` [2789](#), [2801](#)  
`\l__enumext_minipage_left_viii_dim` [3214](#), [3226](#)  
`\l__enumext_minipage_left_X_dim` ..... [58](#)  
`\g__enumext_minipage_right_skip` [58](#), [1085](#), [1090](#),  
[1094](#), [2818](#), [3243](#)  
`\l__enumext_minipage_right_skip` .. [42](#), [58](#), [899](#),  
[914](#), [934](#), [949](#), [1007](#), [1013](#), [1025](#), [1043](#), [1054](#), [1108](#),  
[1115](#), [1129](#), [1177](#), [1194](#)  
`\l__enumext_minipage_right_v_dim` .. [76](#), [1188](#),  
[1193](#), [2533](#), [2537](#)  
`\g__enumext_minipage_right_vii_dim` [82](#), [2797](#),  
[2817](#), [2830](#)  
`\l__enumext_minipage_right_vii_dim` [82](#), [2787](#),  
[2792](#), [2798](#)  
`\g__enumext_minipage_right_viii_dim` .. [3222](#),  
[3242](#), [3255](#)

`\l__enumext_minipage_right_viii_dim` . . 3212, 3217, 3223  
`\g__enumext_minipage_right_X_dim` . . . . . 153  
`\g__enumext_minipage_right_X_skip` . . . . . 153  
`\g__enumext_minipage_stat_int` . 73, 76, 58, 1182, 1199, 2394, 2455, 2460, 2540, 2587, 2592  
`\g__enumext_miniright_code_vii_tl` . 82, 2823, 2829  
`\g__enumext_miniright_code_viii_tl` 3248, 3254  
`\g__enumext_miniright_code_X_tl` . . . . . 153  
`\__enumext_multi_addvspace`: . . . 40, 73, 835, 835, 2426  
`\__enumext_multi_set_vskip`: . . 40, 799, 799, 837  
`\l__enumext_multicols_above_ii_skip` . . . 818  
`\l__enumext_multicols_above_iii_skip` . . . 824  
`\l__enumext_multicols_above_iv_skip` . . . 830  
`\l__enumext_multicols_above_v_skip` 854, 868, 878  
`\l__enumext_multicols_above_X_skip` . . . . . 52  
`\l__enumext_multicols_below_v_skip` 858, 872, 2579  
`\l__enumext_multicols_below_X_skip` . . . . . 52  
`\__enumext_multicols_start`: 73, 2400, 2402, 2402  
`\__enumext_multicols_stop`: 73, 1172, 2432, 2432, 2464  
`\__enumext_newlabel:nn` . . 24, 28, 60, 62, 260, 260, 1813, 1895  
`\l__enumext_newlabel_arg_one_tl` 24, 28, 60, 61, 131, 1736, 1806, 1814, 1888, 1896, 1926  
`\l__enumext_newlabel_arg_two_tl` 24, 28, 59, 61, 131, 1760, 1770, 1784, 1800, 1815, 1881, 1885, 1897  
`\__enumext_parse_keys:n` . . . . . 2285, 2316, 2316  
`\__enumext_parse_keys_vii:n` . . 2835, 2872, 2872  
`\__enumext_parse_keys_viii:n` . 3260, 3292, 3292  
`\__enumext_parse_store_keys:n` . 71, 2322, 2325, 2325  
`\__enumext_parse_store_keys_vii:n` . 83, 2879, 2883, 2883  
`\l__enumext_parsep_i_skip` . 816, 818, 967, 1015  
`\l__enumext_parsep_ii_skip` . . . . . 822, 824, 973  
`\l__enumext_parsep_iii_skip` . . . . . 828, 830, 979  
`\l__enumext_parsep_vii_skip` . . . . . 3089  
`\l__enumext_parsep_viii_skip` . . . . . 3438  
`\l__enumext_partopsep_v_skip` . . 870, 874, 1041, 1045, 1052, 1056, 1072, 1076  
`\l__enumext_partopsep_viii_skip` . . . . . 1117  
`\__enumext_phantomsection`: 28, 224, 253, 257, 273  
`\__enumext_print_footnote`: . . . 1999, 2022, 3107, 3456  
`\__enumext_print_keyans_box:NN` 56, 1625, 1625, 1638, 1826, 1952, 1975  
`\l__enumext_print_keyans_i_tl` . . . . . 3507, 3536  
`\l__enumext_print_keyans_ii_tl` . . . 3512, 3537  
`\l__enumext_print_keyans_iii_tl` . . 3517, 3538  
`\l__enumext_print_keyans_iv_tl` . . . 3522, 3539  
`\l__enumext_print_keyans_vii_tl` . . 3527, 3540  
`\l__enumext_print_keyans_X_tl` . . . . . 94  
`\__enumext_printkeyans:nnn` . 96, 3541, 3544, 3544  
`\__enumext_redefine_item`: . 65, 2075, 2075, 2231  
`\l__enumext_ref_aux_tl` 142, 399, 401, 404, 420, 422, 425, 430, 432, 435  
`\l__enumext_ref_key_arg_tl` . . 142, 393, 398, 405, 417, 426, 436  
`\__enumext_regex_label_ref_key`: . 31, 388, 388, 400, 421, 431  
`\__enumext_register_counter_style:Nn` . . 292, 292, 297, 298, 299, 300, 301  
`\__enumext_remove_extra_parsep_vii`: . . 2850, 3116, 3116  
`\__enumext_remove_extra_parsep_viii`: . 3274, 3465, 3465  
`\__enumext_renew_footnote`: . . . 1999, 2003, 3059  
`\l__enumext_resume_bool` . . . . . 22, 35, 1352, 2214  
`\__enumext_resume_counter`: . 50, 1318, 1343, 1343  
`\__enumext_resume_counter_star`: . . . . . 1320  
`\__enumext_resume_counter_vii`: 50, 1327, 1343, 1354  
`\g__enumext_resume_int` 22, 74, 35, 1347, 1358, 2215, 2468  
`\l__enumext_resume_vii_bool` . . . 35, 1363, 2261  
`\g__enumext_resume_vii_int` . . 84, 35, 2262, 2924  
`\__enumext_safe_exec`: . . . . . 2284, 2301, 2301  
`\__enumext_safe_exec_vii`: . . . 2834, 2855, 2855  
`\__enumext_safe_exec_viii`: . . . 3259, 3278, 3278  
`\__enumext_set_error:nn` . . . . . 3621, 3631, 3633  
`\__enumext_set_label_ref:n` . . . 31, 396, 396, 468  
`\__enumext_set_label_ref_h:n` . 31, 415, 415, 488  
`\__enumext_set_parse:n` . . . . . 3604, 3621, 3621  
`\l__enumext_setkey_tmpa_int` . . . 89, 3597, 3601  
`\l__enumext_setkey_tmpa_seq` 89, 3595, 3605, 3611, 3613, 3615, 3628  
`\l__enumext_setkey_tmpa_tl` . . . . . 89, 3603, 3607  
`\l__enumext_setkey_tmpb_seq` 89, 3596, 3599, 3603, 3604  
`\l__enumext_setkey_tmpb_tl` 89, 3623, 3625, 3626  
`\l__enumext_show_answer_bool` . 105, 1485, 1489, 1508, 1512, 1833, 1948, 2630  
`\__enumext_show_length:nnn` . . 36, 205, 205, 3673, 3674, 3675, 3676, 3677, 3678, 3679, 3680, 3681, 3682, 3688, 3689, 3690, 3691, 3692, 3693, 3694, 3695, 3696, 3697  
`\l__enumext_show_position_bool` 105, 1486, 1488, 1509, 1511, 1837, 1959, 2631  
`\g__enumext_standar_bool` . . . . . 20, 2313  
`\l__enumext_standar_bool` . 20, 1765, 1778, 1794, 2306, 2467, 2865  
`\__enumext_standard_item_vii:w` . . 85, 86, 2977, 2979, 2979  
`\__enumext_standard_item_viii:w` 92, 3330, 3332, 3332  
`\g__enumext_starred_bool` . 83, 84, 20, 1399, 1756, 1766, 1796, 2445, 2869, 2931, 2936  
`\l__enumext_starred_bool` . 83, 84, 20, 1689, 1697, 1781, 1822, 2309, 2862, 2937  
`\__enumext_starred_columns_set_vii`: . . 2717, 2717, 2843  
`\__enumext_starred_columns_set_viii`: . 3142, 3142, 3267  
`\__enumext_starred_item:nn` . . . 2051, 2051, 2081  
`\__enumext_starred_item_vii:w` 85, 86, 2976, 2995, 2995  
`\__enumext_starred_item_vii_aux_i:w` . . 2995, 3000, 3003  
`\__enumext_starred_item_vii_aux_ii:w` . 2995, 3001, 3006, 3008  
`\__enumext_starred_item_vii_aux_iii:w` 2995, 3011, 3020

\\_\_enumext\_starred\_item\_viii:w . . 92, 93, 3329, 3348, 3348  
 \\_\_enumext\_starred\_item\_viii\_aux\_i:w . 3348, 3353, 3356  
 \\_\_enumext\_starred\_item\_viii\_aux\_ii:w 3348, 3354, 3359, 3361  
 \\_\_enumext\_starred\_item\_viii\_aux\_iii:w 3348, 3364, 3373  
 \\_\_enumext\_starred\_joined\_item\_vii:n . 81, 85, 2736, 2736, 2974  
 \\_\_enumext\_starred\_joined\_item\_viii:n 89, 92, 3161, 3161, 3327  
 \\_\_enumext\_start\_from:NNn 33, 509, 509, 522, 544  
 \\_\_enumext\_start\_item\_tmp\_vii: 82, 2846, 2959, 2959  
 \\_\_enumext\_start\_item\_tmp\_viii: 90, 3270, 3312, 3312  
 \\_\_enumext\_start\_item\_vii:w 86, 2987, 2992, 3017, 3024, 3026, 3026  
 \\_\_enumext\_start\_item\_viii:w 92, 93, 3340, 3345, 3370, 3377, 3379, 3379  
 \\_\_enumext\_start\_list:nn 27, 69, 79, 211, 213, 2288, 2493, 2644, 2838, 3262  
 \\_\_enumext\_start\_mini\_vii: . 84, 2785, 2785, 2916  
 \\_\_enumext\_start\_mini\_viii: 91, 3210, 3210, 3303  
 \\_\_enumext\_start\_store\_level: . 72, 2287, 2354, 2354  
 \\_\_enumext\_start\_store\_level\_vii: . 85, 2837, 2939, 2939  
 \l\_\_enumext\_start\_X\_int . . . . . 70, 539  
 \\_\_enumext\_stop\_item\_tmp\_vii: . 82, 85, 86, 2845, 2849, 2961, 3028  
 \\_\_enumext\_stop\_item\_tmp\_viii: 90, 92, 93, 3269, 3273, 3314, 3381  
 \\_\_enumext\_stop\_item\_vii: 86, 88, 3028, 3092, 3092  
 \\_\_enumext\_stop\_item\_viii: . . 93, 94, 3381, 3441, 3441  
 \\_\_enumext\_stop\_list: . . 27, 211, 214, 2297, 2503, 2657, 2851, 3275  
 \\_\_enumext\_stop\_mini\_vii: 82, 84, 2804, 2804, 2921  
 \\_\_enumext\_stop\_mini\_viii: . 91, 3210, 3229, 3308  
 \\_\_enumext\_stop\_store\_level: . . 72, 2298, 2354, 2372  
 \\_\_enumext\_stop\_store\_level\_vii: . . 85, 2852, 2939, 2949  
 \l\_\_enumext\_store\_active\_bool 23, 50, 71, 83, 82, 1333, 1345, 1356, 1651, 2320, 2357, 2508, 2515, 2603, 2661, 2877, 2941, 2951, 3286  
 \\_\_enumext\_store\_addto\_prop:n 55, 61, 1542, 1542, 1551, 1676, 1867  
 \\_\_enumext\_store\_addto\_seq:n 55, 62, 1552, 1552, 1558, 1565, 1579, 1587, 1596, 1614, 1622, 1740, 1931  
 \l\_\_enumext\_store\_ans\_bool 115, 1372, 1419, 1561, 1585, 1592, 1620, 1664, 3043, 3396  
 \l\_\_enumext\_store\_anskey\_arg\_tl . . 23, 58, 82, 1682, 1691, 1693, 1699, 1707, 1710, 1720, 1725, 1728, 1734, 1740  
 \\_\_enumext\_store\_anskey\_code:nnnn . 57, 1670, 1674, 1674  
 \\_\_enumext\_store\_anskey\_show\_left:n 60, 1681, 1831, 1831  
 \\_\_enumext\_store\_anskey\_show\_wrap:n 60, 1819, 1819, 1835, 1850  
 \l\_\_enumext\_store\_columns\_break\_bool . 1645, 1688  
 \l\_\_enumext\_store\_columns\_join\_int 82, 1696, 1701  
 \l\_\_enumext\_store\_columns\_sep\_vii\_bool 2898  
 \l\_\_enumext\_store\_columns\_sep\_vii\_dim 2903, 2907  
 \l\_\_enumext\_store\_columns\_sep\_X\_bool . . . 94  
 \l\_\_enumext\_store\_columns\_sep\_X\_dim . . . . 94  
 \l\_\_enumext\_store\_columns\_vii\_bool . . . 2885  
 \l\_\_enumext\_store\_columns\_vii\_int 2890, 2894  
 \l\_\_enumext\_store\_columns\_X\_bool . . . . . 94  
 \l\_\_enumext\_store\_columns\_X\_int . . . . . 94  
 \\_\_enumext\_store\_internal\_ref: . . 57, 59, 1679, 1742, 1742  
 \l\_\_enumext\_store\_item\_symbol\_sep\_dim 1643, 1717, 1722  
 \l\_\_enumext\_store\_item\_symbol\_tl . 1641, 1708, 1712  
 \l\_\_enumext\_store\_keyans\_label\_tl 23, 61, 62, 82, 1855, 1858, 1861, 1865, 1867, 1906, 1909, 1912, 1916, 1922, 1931, 1932  
 \\_\_enumext\_store\_level\_close: . 55, 1559, 1583, 2376  
 \\_\_enumext\_store\_level\_close\_vii: 1590, 1618, 2955  
 \\_\_enumext\_store\_level\_open: . . 54, 55, 71, 1559, 1559, 2363, 2368  
 \\_\_enumext\_store\_level\_open\_vii: . . 83, 1590, 1590, 2945  
 \g\_\_enumext\_store\_name\_tl 23, 74, 82, 1437, 1440, 1459, 1462, 2448, 2486, 2934, 3140  
 \l\_\_enumext\_store\_name\_tl 23, 50, 82, 1332, 1349, 1360, 1544, 1545, 1546, 1548, 1554, 1555, 1556, 1808, 1809, 1845, 1890, 1891, 1967, 2448, 2469, 2472, 2925, 2928, 2934  
 \l\_\_enumext\_store\_opt\_vii\_tl . 1594, 1604, 1610, 1614, 2892, 2905  
 \l\_\_enumext\_store\_opt\_X\_tl . . . . . 94  
 \l\_\_enumext\_store\_ref\_key\_bool 57, 1494, 1677, 1731, 1899, 1919  
 \l\_\_enumext\_store\_upper\_level\_X\_bool . . . 94  
 \l\_\_enumext\_store\_write\_aux\_file\_tl 24, 60, 62, 131, 1811, 1817, 1893, 1901  
 \\_\_enumext\_storing\_set:n . 50, 1316, 1325, 1330, 1330  
 \l\_\_enumext\_the\_counter\_vii\_tl . . . . . 422  
 \l\_\_enumext\_the\_counter\_viii\_tl . . . . . 432  
 \l\_\_enumext\_the\_counter\_X\_tl . . . . . 142  
 \\_\_enumext\_tmp:n 30, 34, 44, 51, 52, 57, 64, 69, 70, 81, 94, 104, 123, 128, 134, 138, 146, 152, 153, 172, 622, 626, 1365, 1382, 1470, 1498, 1499, 1516, 1744, 1751, 1752, 1773, 1787, 1790, 1802, 1871, 1878, 2197, 2242, 2243, 2281  
 \\_\_enumext\_tmp:nn 324, 345, 346, 374, 375, 387, 539, 558, 603, 621, 679, 687, 688, 702, 767, 783, 784, 798, 1203, 1219, 1517, 1541, 1983, 1998  
 \\_\_enumext\_tmp:nnn 456, 472, 473, 474, 475, 476, 492, 493  
 \\_\_enumext\_tmp:nnnnn 559, 584, 587, 590, 592, 594, 597, 600  
 \\_\_enumext\_tmp:w . . . . . 3490, 3493  
 \l\_\_enumext\_tmpa\_vii\_int . . . . . 2727, 2730  
 \l\_\_enumext\_tmpa\_viii\_int . . . . . 3152, 3155  
 \l\_\_enumext\_tmpa\_X\_int . . . . . 153

`\l__enumext_topsep_v_skip` 856, 860, 1010, 1023, 1031, 1036, 1056, 1060, 2660, 2691

`\l__enumext_topsep_vii_skip` .. 1087, 1096, 1100

`\l__enumext_topsep_viii_skip` . 1109, 1131, 1135

`\__enumext_use_key_ref:` .... 31, 408, 408, 2233

`\__enumext_use_key_ref_h:` .. 32, 440, 440, 2267

`\l__enumext_vspace_a_star_v_bool` ..... 1252

`\l__enumext_vspace_a_star_vii_bool` ... 1274

`\l__enumext_vspace_a_star_viii_bool` ... 1285

`\l__enumext_vspace_a_star_X_bool` ..... 70

`\__enumext_vspace_above:` .. 48, 1220, 1220, 2381

`\__enumext_vspace_above_v:` . 49, 1248, 1248, 2531

`\l__enumext_vspace_above_v_skip` .. 1250, 1254, 1256

`\__enumext_vspace_above_vii:` .. 49, 1270, 1270, 2914

`\l__enumext_vspace_above_vii_skip` 1272, 1276, 1278

`\__enumext_vspace_above_viii:` . 49, 1270, 1281, 3301

`\l__enumext_vspace_above_viii_skip` 1283, 1287, 1289

`\l__enumext_vspace_b_star_v_bool` ..... 1263

`\l__enumext_vspace_b_star_vii_bool` ... 1296

`\l__enumext_vspace_b_star_viii_bool` ... 1307

`\l__enumext_vspace_b_star_X_bool` ..... 70

`\__enumext_vspace_below:` .. 48, 1234, 1234, 2466

`\__enumext_vspace_below_v:` . 49, 1259, 1259, 2599

`\l__enumext_vspace_below_v_skip` .. 1261, 1265, 1267

`\__enumext_vspace_below_vii:` .. 49, 1292, 1292, 2923

`\l__enumext_vspace_below_vii_skip` 1294, 1298, 1300

`\__enumext_vspace_below_viii:` . 49, 1292, 1303, 3310

`\l__enumext_vspace_below_viii_skip` 1305, 1309, 1311

`\__enumext_widest_from:nNn` .. 34, 523, 523, 538, 550

`\g__enumext_widest_label_tl` 22, 29, 40, 312, 316, 320

`\l__enumext_wrap_label_opt_v_bool` .... 2094

`\l__enumext_wrap_label_opt_vii_bool` 86, 2986

`\l__enumext_wrap_label_opt_viii_bool` 92, 3339

`\l__enumext_wrap_label_opt_X_bool` ..... 70

`\l__enumext_wrap_label_v_bool` 2090, 2094, 2102, 2157

`\l__enumext_wrap_label_vii_bool` 86, 2985, 2990, 2998, 3076

`\l__enumext_wrap_label_viii_bool` .. 92, 3338, 3343, 3351, 3425

`\l__enumext_wrap_label_X_bool` ..... 70

`\__enumext_wrapper_label_v:n` ..... 2159, 2639

`\__enumext_wrapper_label_vii:n` ..... 3079

`\__enumext_wrapper_label_viii:n` ..... 3428

`\__enumext_zero_parsep:` ..... 43, 907, 962, 962

`enumext*` ..... 4, 2832

`enumXi` ..... 284

`enumXii` ..... 284

`enumXiii` ..... 284

`enumXiv` ..... 284

`enumXv` ..... 284

`enumXvi` ..... 284

`enumXvii` ..... 284

`enumXviii` ..... 284

Environments provide by `enumext`:

`enumext*` .. 21, 22, 24–26, 29–33, 35, 36, 38, 39, 45, 46, 49–51, 53–59, 63, 64, 71, 72, 83, 85, 86, 88, 90, 92, 96, 99, 100

`enumext` 21, 22, 24, 26, 29–32, 34–37, 39–48, 50–59, 63–72, 74, 75, 79, 80, 82, 85, 96, 98, 99

`keyans*` 21–23, 25, 26, 29–33, 35, 36, 38, 39, 45, 46, 49, 51, 54, 55, 64, 91, 93, 99, 100

`keyanspic` 21–24, 29, 30, 33, 46, 50, 51, 55, 61–63, 77–80, 99

`keyans` 21–24, 26, 29, 30, 33–39, 41, 44–51, 54, 55, 61–63, 66–69, 75, 77–79, 82, 92, 99

Environments:

`enumext*` ..... 70

`keyans*` ..... 70

`list` ..... 26, 27, 68–70

`lrbox` ..... 80, 87, 88, 94

`minipage` ..... 26, 27, 39, 41, 77–80, 87, 88, 94

`multicols` ..... 39–42, 47, 73, 74, 76, 77

exp commands:

`\exp_after:wN` ..... 3493

`\exp_args:Ne` ..... 2318, 3481

`\exp_not:N` 150, 315, 404, 425, 435, 636, 650, 651, 662, 663, 674, 675, 1736, 1842, 1843, 1924, 1964, 1965, 3490

`\exp_not:n` 404, 405, 425, 426, 435, 436, 637, 1525, 1532, 1701, 1712, 1722, 1736, 1737, 1814, 1896, 1926, 1928, 2336, 2349, 2894, 2907

**F**

`\fbox` ..... 1475

file commands:

`\file_input_stop:` ..... 3790

`first` ..... 688

`font` ..... 324

`\footnote` ..... 64

`\footnote` ..... 64, 2007

`\footnotemark` ..... 2017

`\footnotesize` ..... 1843, 1965

`\footnotetext` ..... 2001

**G**

`\getkeyans` ..... 13, 95, 3479

group commands:

`\group_begin:` .. 1663, 1841, 1963, 3055, 3074, 3408, 3423, 3501, 3535

`\group_end:` 1672, 1848, 1971, 3084, 3096, 3433, 3445, 3503, 3542

**H**

`\hbadness` ..... 3103, 3452

hbox commands:

`\hbox_set:Nn` ..... 304

`\hfill` 354, 358, 363, 364, 1174, 1192, 1736, 1924, 2809, 3234

hook commands:

`\hook_gput_code:nnn` ..... 9, 180, 184, 222

`\hook_gset_rule:nnnn` ..... 223

`\hspace` ..... 3114, 3463

`\hyperlink` ..... 58, 62

`\hyperlink` ..... 1736, 1924

`\hypertarget` ..... 28

`\hypertarget` ..... 252

**I**

`\IfHyperBoolean` ..... 230

K	
keyans	11, <u>2488</u>
keyans*	11, <u>3257</u>
keyanspic	12, <u>2641</u>

above*	23, 48, 49
above	23, 48, 49, 72, 75, 84, 91
after	37, 38, 74, 77, 84, 91
align	23, 30, 67, 87
before*	37, 38, 72, 84, 91
before	37, 38, 75
below*	23, 48, 49
below	23, 48, 49, 74, 77, 84, 91
check-ans	23, 24, 26, 50–52, 57, 62–65, 72, 74, 88, 99
columns-sep*	24, 54, 71, 83
columns-sep	39, 55, 71, 73, 76, 83
columns*	24, 54, 71, 83
columns	22, 39, 42, 48, 55, 71, 73, 76, 83
first	37, 38, 87
font	30, 67, 87
item-pos*	57, 58, 63
item-sym*	22, 57, 58, 63, 65
item*-sep	65
itemindent	23, 35, 36, 66, 87
itemsep	34, 70
labelsep	30, 65, 68, 87
labelwidth	29, 30, 32–34, 68
label	22, 29, 33, 34, 80
lisparindent	70
list-indent	22, 35, 79
list-offset	35
listparindent	35, 87
mark-ans	24, 53, 60
mark-pos	53, 54
mark-ref	24, 53, 59, 61
mini-env	23, 39, 41, 47, 48, 64, 72, 76, 82, 84, 90, 91
mini-sep	23, 39, 72, 76
miniright*	23, 39
miniright	23, 39, 46, 82
minirigth*	26
minirigth	26
no-store	24, 51, 52
noitemsep	34, 43
nosep	34, 43
parindent	70
parsep	34, 70, 87
partopsep	34
ref	25, 31
resume	22, 50, 69, 74, 84
rightmargin	35
save-ans	23, 50, 55, 57, 61, 62, 66, 74, 75, 78, 84, 92, 95, 96
save-key	24
show-ans	24, 53, 54, 56, 57, 60, 66
show-length	26, 36, 69, 98, 99
show-pos	24, 53, 54, 56, 57, 60, 66
start	23, 26, 33, 34, 69
store-brk	57, 58
store-ref	24, 28, 53, 57–59, 61, 62, 66
topsep	34
widest	22, 26, 34
wrap-ans	53, 56, 60
wrap-label*	30, 64, 67, 86, 87, 92
wrap-label	30, 67, 86, 87, 92

```
\keys_define:nn 326, 348, 377, 458, 478, 494, 541, 561,
    605, 624, 681, 690, 769, 786, 1205, 1314, 1323, 1367,
    1472, 1501, 1519, 1639, 1985, 3505, 3567
\l_keys_key_str ..... 3658
```



\keys_set:nn	340, 793, 1210, 1215, 1685, 2318, 2527, 2876, 3296, 3569, 3570, 3571, 3572, 3573, 3574, 3575, 3576, 3577, 3578, 3579, 3580, 3618
L	
label	456, 476, 494
Labels provide by enumext:	
\Alph*	29
\Roman*	29
\alph*	29
\arabic*	29, 31
\roman*	29
\labelsep	79
\labelsep	2681, 2684
labelsep	324
\labelwidth	29, 79
\labelwidth	2681, 2682
labelwidth	324
\leftmargin	22, 69
\leftmargin	68, 2681
legacy commands:	
\legacy_if:nTF	3029, 3032, 3382, 3385
\legacy_if_gset_false:n	883
\legacy_if_set_false:n	3031, 3384
\legacy_if_set_true:n	2991, 3016, 3023, 3036, 3344, 3369, 3376, 3389
\linewidth	72, 76
\linewidth	2389, 2537, 2703, 2730, 2791, 3155, 3216
\list	27
\list	213
list-indent	603
list-offset	603
\listparindent	2683
listparindent	603
\lrbx	3056, 3409
M	
\makebox	80
\makebox	1629, 1631, 2131, 3070, 3078, 3082, 3419, 3427, 3431
\makelabel	64, 67, 80
\makelabel	67, 2137, 2153
\makesavenoteenv	246
mark-ans	1470
mark-pos	1470, 1499
mark-ref	1470
mini-env	767
mini-sep	767
\minipage	27
\minipage	219
\miniright	9, 46, 1151, 2458, 2590
\miniright*	9
mode commands:	
\mode_if_vertical:TF	838, 866, 989, 1068
\mode_leave_vertical:	636, 650, 662, 674, 1598, 1606, 1627, 2129, 3068, 3417
msg commands:	
\msg_error:nn	2518, 2522, 2609, 2668, 2860, 3283, 3289, 3581
\msg_error:nnn	1155, 1159, 1184, 1201, 3495, 3500, 3564, 3634
\msg_error:nnnn	1653, 1657, 1661, 2510, 2605, 2613
\msg_fatal:nn	2305
\msg_fatal:nnn	278
\msg_info:nnn	13, 16, 228, 242
\msg_line_context:	3662, 3667, 3672, 3687, 3710, 3714, 3718, 3723, 3728, 3733, 3738, 3742, 3747, 3752, 3756, 3761, 3765, 3770, 3775, 3780, 3784, 3788
\msg_new:nnn	3635, 3639, 3643, 3647, 3652, 3656, 3660, 3665, 3670, 3685, 3700, 3707, 3712, 3716, 3721, 3726, 3731, 3736, 3740, 3745, 3750, 3754, 3759, 3763, 3768, 3773, 3778, 3782, 3786
\msg_term:nnn	1437, 1459
\msg_term:nnnn	2227, 2237, 2272, 2277
\msg_warning:nn	2457, 2589
\msg_warning:nnn	1440, 1462
\msg_warning:nnnn	1942, 2169, 2174, 2741, 2754, 3166, 3179
\multicolsep	73, 76
\multicolsep	2419, 2562
N	
\NeedsTeXFormat	3
\newcounter	281
\NewDocumentCommand	1151, 1649, 2601, 3479, 3533, 3588
\NewDocumentEnvironment	2282, 2488, 2641, 2832, 3257
\newlabel	28
\newlabel	264
no-store	1365
\noindent	82, 90
\noindent	2396, 2542, 2800, 2845, 3111, 3225, 3269, 3460
\nointerlineskip	2396, 2542, 2800, 3225
noitemsep	559
\nopagebreak	849, 877, 1000, 1079, 1142, 1148
\normalfont	1842, 1964
nosep	559
P	
Packages:	
enumext	21, 50, 68, 77, 98
enumitem	28, 29
expl3	80
footnotehyper	27
hyperref	24, 26-28, 31, 58, 62, 86, 87, 93, 98
lua-visual-debug	41
multicol	21, 98
shortlst	80
\par	849, 877, 1000, 1079, 1142, 1148, 1177, 1194, 1821, 2440, 2462, 2579, 2594, 2715, 2818, 2825, 3111, 3125, 3243, 3250, 3460, 3474
\parindent	3088, 3437
\parsep	40, 43, 78, 79
\parsep	1599, 1607, 2257, 2680, 2687, 2692
parsep	559
\parskip	3089, 3438
\partopsep	79
\partopsep	2258, 2685
partopsep	559
peek commands:	
\peek_meaning:NTF	2968, 2982, 2999, 3010, 3321, 3335, 3352, 3363
\peek_meaning_remove:NTF	2975, 3328
\peek_remove_spaces:n	2115
\phantomsection	28
\phantomsection	253
prg commands:	
\prg_do_nothing:	257
\prg_new_protected_conditional:Npnn	199
\prg_replicate:nn	208, 3705
\prg_return_false:	203
\prg_return_true:	202

\printkeyans . . . . . 13, 96, 3533  
 prop commands:  
   \prop\_count:N . . . . . 1548, 1809, 1845, 1891, 1967  
   \prop\_gput:Nnn . . . . . 1546  
   \prop\_if\_exist:Ntf . . . . . 1544, 3499  
   \prop\_item:Nn . . . . . 3502  
   \prop\_new:N . . . . . 1545  
 \ProvidesExplPackage . . . . . 4

## R

\raggedcolumns . . . . . 2428, 2568  
 \ref . . . . . 59, 61  
 ref . . . . . 456, 476  
 \refstepcounter . . . . . 3038, 3391  
 regex commands:  
   \regex\_match:nnTF 201, 516, 518, 530, 532, 2329, 2342,  
     2887, 2900  
   \regex\_replace\_once:nnN . . . . . 392  
 \renewcommand . . . . . 404, 425, 435  
 \RenewDocumentCommand . . . . . 2007, 2077, 2111, 2137, 2153  
 \RequirePackage . . . . . 17  
 resume . . . . . 1314  
 resume\* . . . . . 1314  
 rightmargin . . . . . 603  
 \Roman . . . . . 29, 33, 34  
 \Roman . . . . . 300  
 \roman . . . . . 29, 33, 34  
 \roman . . . . . 301, 474, 3521

## S

save-ans . . . . . 1314  
 scan commands:  
   \scan\_stop: . . . . . 80, 2694, 2844, 3268, 3490, 3493  
 seq commands:  
   \seq\_clear:N . . . . . 3595  
   \seq\_const\_from\_clist:Nn . . . . . 3583  
   \seq\_count:N . . . . . 2654, 3599  
   \seq\_gclear:N . . . . . 2005, 2006  
   \seq\_gput\_right:Nn . . . . . 1556, 2018, 2019  
   \seq\_if\_empty:Ntf . . . . . 2024, 3548, 3613  
   \seq\_if\_exist:Ntf . . . . . 1554, 3546  
   \seq\_item:Nn . . . . . 2712  
   \seq\_map\_function:NN . . . . . 3604  
   \seq\_map\_inline:Nn . . . . . 3553, 3558, 3592, 3614, 3615  
   \seq\_map\_pairwise\_function:NNN . . . . . 2026  
   \seq\_new:N . . . . . 92, 93, 110, 140, 141, 1555  
   \seq\_pop\_left:NN . . . . . 3603  
   \seq\_put\_right:Nn . . . . . 2615, 3611, 3628  
   \seq\_set\_from\_clist:Nn . . . . . 3596  
   \seq\_set\_map\_e:NNn . . . . . 3605  
 \setcounter . . . . . 527, 531, 533, 2215, 2217, 2262, 2264, 2659  
 \setenumext . . . . . 5–8, 97, 3509, 3514, 3519, 3524, 3529, 3588  
 \setlength . . . . . 1600, 1608  
 show-ans . . . . . 1470, 1499  
 show-length . . . . . 679  
 skip commands:  
   \skip\_add:Nn . . . . . 818, 824, 830, 840, 844, 868, 872, 969,  
     975, 981, 991, 995, 1017, 1070, 1074, 2680  
   \skip\_eval:n . . . . . 1599, 1607  
   \skip\_gset:Nn . . . . . 1090, 1094, 1098  
   \skip\_gzero\_new:N . . . . . 1085, 1086  
   \skip\_horizontal:N 651, 663, 675, 3071, 3085, 3420,  
     3434  
   \skip\_horizontal:n . . . . . 637, 1628, 1636, 2130, 2132,  
     3069, 3418

\skip\_if\_eq:nnTF . . . . . 816, 822, 828, 892, 926, 967, 973,  
   979, 1010, 1015, 1036, 1087, 1109, 1222, 1236, 1250,  
   1261, 1272, 1283, 1294, 1305  
 \skip\_new:N . . . . . 54, 55, 59, 60, 61, 62, 63, 114, 170  
 \skip\_set:Nn . . . . . 801, 805, 854, 858, 895, 899, 903, 910,  
   914, 918, 929, 934, 938, 944, 949, 954, 1012, 1013,  
   1014, 1021, 1025, 1029, 1038, 1043, 1047, 1050, 1054,  
   1058, 1089, 1093, 1111, 1115, 1119, 1125, 1129, 1133,  
   2674, 2688  
 \skip\_set\_eq:NN 2210, 2256, 2257, 3088, 3089, 3437,  
   3438  
 \skip\_use:N 803, 807, 842, 846, 850, 870, 874, 893, 912,  
   921, 927, 932, 936, 947, 951, 952, 957, 993, 997, 1023,  
   1223, 1227, 1230, 1237, 1241, 1244, 2440  
 \skip\_zero:N . . . . . 2258, 2419, 2562, 2685, 2686  
 \skip\_zero\_new:N 1005, 1006, 1007, 1084, 1106, 1107,  
   1108  
 \c\_zero\_skip . . . . . 816, 822, 828, 893, 927, 967, 973, 979,  
   1010, 1015, 1036, 1087, 1109, 1223, 1237, 1250, 1261,  
   1272, 1283, 1294, 1305  
 \small . . . . . 3511, 3516, 3521, 3526, 3531  
 \star . . . . . 1989  
 start . . . . . 539  
 \stepcounter . . . . . 2011, 2622  
 store-ref . . . . . 1470  
 str commands:  
   \c\_backslash\_str 3714, 3723, 3724, 3728, 3729, 3733,  
     3734, 3765, 3766, 3770, 3775, 3776  
   \c\_colon\_str . . . . . 1808, 1890, 3490  
   \str\_count:n . . . . . 208, 3705  
   \str\_if\_eq:nnTF . . . . . 2220, 2268  
   \str\_if\_eq\_p:nn . . . . . 2213, 2261  
   \str\_if\_in:nnTF . . . . . 3486  
   \str\_new:N . . . . . 109, 165  
   \str\_set:Nn . . . . . 380, 381, 382, 1481, 1482, 1504, 1505  
 \string . . . . . 246  
 \strutbox 897, 901, 905, 916, 920, 931, 940, 946, 956, 969, 975,  
   981, 1012, 1013, 1014, 1017, 1027, 1031, 1040, 1047,  
   1052, 1060, 1089, 1090, 1093, 1100, 1113, 1121, 1127,  
   1135, 2690

## T

TeX and  $\TeX$  2<sub>ε</sub> commands:

\@auxout . . . . . 262  
 \protected@write . . . . . 262

text commands:

\text\_expand:n . . . . . 3482  
 \textasteriskcentered . . . . . 1478, 1492  
 \thepage . . . . . 268

tl commands:

\c\_space\_tl . . . . . 1865, 1916, 1957, 1980, 3672, 3687  
 \tl\_clear:N . . . . . 353, 359, 1682, 1855, 1906  
 \tl\_clear\_new:N . . . . . 310  
 \tl\_const:Nn . . . . . 142, 294  
 \tl\_gclear:N 1944, 2148, 2486, 2829, 3072, 3140, 3254,  
   3421  
 \tl\_gput\_right:Nn . . . . . 295  
 \tl\_greplace\_all:Nnn . . . . . 316  
 \tl\_gset:Nn . . . . . 1932, 2448, 2934, 3005, 3358  
 \tl\_gset\_eq:NN . . . . . 312, 2057, 3065, 3414  
 \tl\_if\_blank:nTF . . . . . 3063, 3412  
 \tl\_if\_empty:Ntf . . . . . 410, 444, 450, 1563, 1594, 1708,  
   1940, 2127, 3626  
 \tl\_if\_novalue:nTF . . . . . 1683, 1694, 1863, 1914, 1956,  
   1979, 2009, 2034, 2053, 2058, 2088, 2652, 2874, 3294,  
   3590

<code>\tl_map_inline:Nn</code> .....	313, 390
<code>\tl_new:N</code> 32, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 90, 96, 97, 107, 108, 119, 131, 132, 133, 136, 144, 145, 148, 149, 164, 167	3090, 3248, 3424, 3435, 3439, 3536, 3537, 3538, 3539, 3540, 3607
<code>\tl_put_left:Nn</code> ....	1571, 1604, 1691, 1950, 1973
<code>\tl_put_right:Nn</code> 311, 402, 423, 433, 1523, 1530, 1575, 1610, 1693, 1699, 1707, 1710, 1720, 1725, 1728, 1734, 1760, 1770, 1784, 1800, 1806, 1811, 1858, 1861, 1865, 1881, 1885, 1888, 1893, 1909, 1912, 1916, 1922, 1957, 1980, 2334, 2347, 2892, 2905, 3507, 3512, 3517, 3522, 3527	token commands: <code>\token_to_str:N</code> ..... 264
<code>\tl_remove_all:Nn</code> .....	3625
<code>\tl_remove_once:Nn</code> .....	1748, 1875
<code>\tl_replace_all:Nnn</code> .....	315
<code>\tl_reverse:N</code> .....	1747, 1749, 1874, 1876
<code>\tl_set:Nn</code> 150, 280, 354, 358, 363, 364, 398, 417, 634, 648, 660, 672, 1332, 1477, 1491, 1839, 1961, 2055, 3623	<code>\topsep</code> ..... 1600, 1608 <code>topsep</code> ..... 559 <code>\typeout</code> ..... 232, 235, 245, 246
<code>\tl_set_eq:NN</code> 321, 399, 401, 420, 422, 430, 432, 1746, 1873, 2100, 2104, 2633, 2635	<b>U</b>
<code>\tl_to_str:n</code> .....	<code>\u</code> ..... 393
<code>\tl_trim_spaces:n</code> .....	use commands: <code>\use:N</code> ..... 209, 2144, 2291 <code>\use:n</code> ..... 3488 <code>\use_none:nn</code> ..... 256 <code>\usecounter</code> ..... 2211, 2259
<code>\tl_use:N</code> . 317, 320, 412, 446, 452, 705, 709, 713, 717, 721, 725, 729, 733, 737, 741, 745, 749, 753, 757, 761, 765, 1633, 1753, 1761, 1772, 1786, 1791, 1803, 2042, 2048, 2073, 2091, 2095, 2103, 2131, 2139, 2140, 2147, 2155, 2156, 2162, 2289, 2494, 2638, 2823, 3075, 3086,	<b>V</b> <code>\value</code> ..... 2468, 2473, 2924, 2929 <code>\vspace</code> 884, 1227, 1230, 1241, 1244, 1254, 1256, 1265, 1267, 1276, 1278, 1287, 1289, 1298, 1300, 1309, 1311, 1599, 1607, 2649, 2660, 3126, 3475

<b>W</b>	
<code>widest</code> .....	539
<code>wrap-ans</code> .....	1470
<code>wrap-label</code> .....	324
<code>wrap-label*</code> .....	324