

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-02<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>102</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 2024-05-02.

<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 / 113

1.1 Description and usage

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

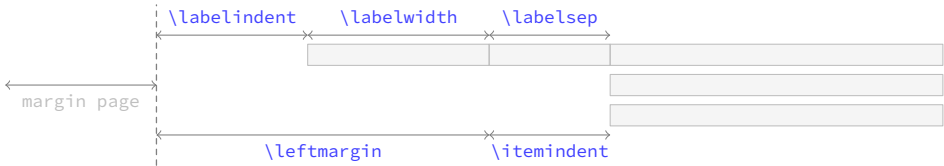


Figure 1: Representation of horizontal lengths in `enumitem`.

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

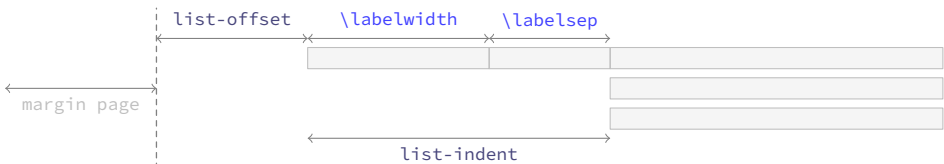


Figure 2: Representation of horizontal lengths concept in `enumext`.

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

### 1.3.2 Support for multicols

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

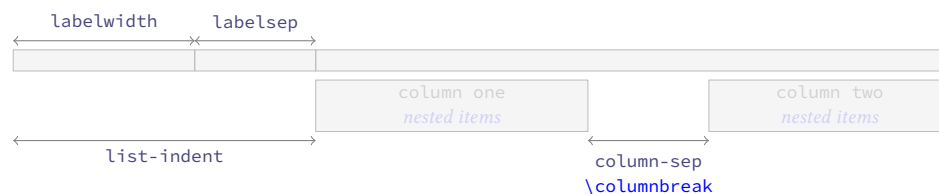


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

### 1.3.3 Support for minipage

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



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

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

### 1.3.4 The \label and \ref system

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§3.1), the standard  $\TeX$  `\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 *key* that sets a *rubber or rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and *keyans* and *keyanspic* environments. All *keys* related to vertical or horizontal spacing accept a “*skip*” or “*dim*” expression if passed between braces, i.e. you do not need to use `\dimexpr` or `\dimeval` to perform calculations.

### 3.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3.2 Keys for spaces

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

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

### 3.2.1 Vertical spaces

`topsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

`parsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

`partopsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

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



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

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

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

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

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

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

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



4.6 The environment keyanspic

keyanspic

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

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

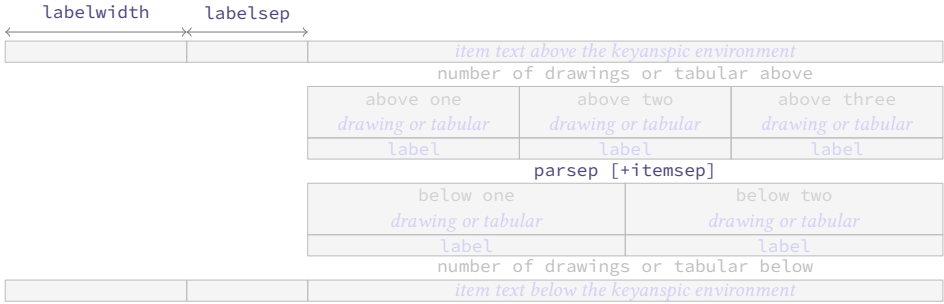


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

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

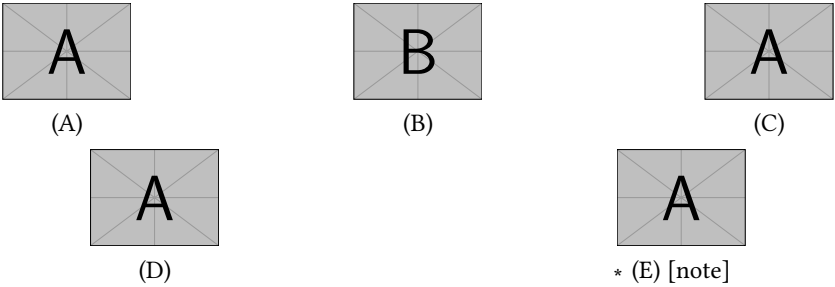
The `\anspic` command take three arguments, the *starred version* “\*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

The “content” can only be accessed “after” it is stored, if the `<store name>` does not exist the command will return an error. The form taken by the argument `<store name> : <position>` is the same as that used to generate the internal “label and ref” system when `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:



A



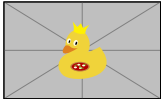
D



B



C



E

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

B value

C value

D correct

E value



Test keys

1. (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 :) .



Factor  $x^2 - 2x + 1$



Factor  $3x + 3y + 3z$

The following questions need to be cuaqtified :)



True False

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



Related to Linux

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

ii. `xsim`

iii. `exsheets`

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

1. (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  $\LaTeX$  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 2024-05-02** – 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 / 113

<b>R</b>	<b>\rightmargin</b> ..... 8
<b>\raggedcolumns</b> ..... 4	<b>T</b>
<b>\ref</b> ..... 4	<b>\topsep</b> ..... 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   {2024-05-02}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

### 10.4 Definition of variables

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

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

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

```

27 \bool_new:N \l__enumext_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
\l__enumext_keyans_tmppb_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
89 \tl_new:N \l__enumext_keyans_tmppb_tl
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmppb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmppb_seq
```

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

```
90 \tl_new:N \l__enumext_setkey_tmpa_tl
91 \tl_new:N \l__enumext_setkey_tmppb_tl
92 \int_new:N \l__enumext_setkey_tmpa_int
93 \seq_new:N \l__enumext_setkey_tmpa_seq
94 \seq_new:N \l__enumext_setkey_tmppb_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*`.

```
95 \cs_set_protected:Npn \l__enumext_tmp:n #1
96 {
97   \tl_new:c { \l__enumext_store_opt_#1_tl }
98   \tl_new:c { \l__enumext_print_keyans_#1_tl }
99   \bool_new:c { \l__enumext_store_columns_#1_bool }
100  \int_new:c { \l__enumext_store_columns_#1_int }
101  \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
102  \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
103  \bool_new:c { \l__enumext_store_upper_level_#1_bool }
104 }
105 \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).

```
106 \bool_new:N \l__enumext_show_answer_bool
107 \bool_new:N \l__enumext_show_position_bool
108 \tl_new:N \l__enumext_mark_ref_sym_tl
109 \tl_new:N \l__enumext_mark_answer_sym_tl
110 \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).

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

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

```
\l__enumext_store_ans_bool
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\l__enumext_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.

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

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

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

```
130 \bool_new:N \l__enumext_hyperref_bool
131 \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.

```

132 \tl_new:N \l__enumext_newlabel_arg_one_tl
133 \tl_new:N \l__enumext_newlabel_arg_two_tl
134 \tl_new:N \l__enumext_store_write_aux_file_tl
135 \cs_set_protected:Npn \__enumext_tmp:n #1
136 {
137     \tl_new:c { l__enumext_label_copy_#1_tl }
138 }
139 \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`.

```

140 \int_new:N \g__enumext_footnote_int
141 \seq_new:N \g__enumext_footnote_arg_seq
142 \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).

```

143 \tl_const:Nn \c__enumext_counter_style_tl
144 { { arabic } { roman } { Roman } { alph } { Alph } }
145 \tl_new:N \l__enumext_ref_key_arg_tl
146 \tl_new:N \l__enumext_ref_aux_tl
147 \cs_set_protected:Npn \__enumext_tmp:n #1
148 {
149     \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
150     \tl_new:c { l__enumext_the_counter_#1_tl }
151     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
152 }
153 \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.

```

154 \cs_set_protected:Npn \__enumext_tmp:n #1
155 {
156     \bool_new:c { l__enumext_item_starred_#1_bool }
157     \int_new:c { l__enumext_item_column_pos_#1_int }
158     \int_new:c { g__enumext_item_count_all_#1_int }
159     \int_new:c { l__enumext_joined_item_#1_int }
160     \int_new:c { l__enumext_joined_item_aux_#1_int }
161     \int_new:c { l__enumext_tmpa_#1_int }
162     \box_new:c { l__enumext_item_text_#1_box }
163     \dim_new:c { l__enumext_joined_width_#1_dim }
164     \dim_new:c { l__enumext_item_width_#1_dim }
165     \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
166     \str_new:c { l__enumext_align_label_#1_str }
167     \bool_new:c { g__enumext_minipage_active_#1_bool }
168     \tl_new:c { g__enumext_miniright_code_#1_tl }
169     \bool_new:c { g__enumext_minipage_center_#1_bool }
170     \dim_new:c { g__enumext_minipage_right_#1_dim }
171     \skip_new:c { g__enumext_minipage_right_#1_skip }
172 }
173 \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`.

```

174 \clist_const:Nn \c__enumext_all_envs_clist
175 {
176     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
177     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
178 }

```

(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.

```
179 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
180 {
181   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
182 }
```

(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.

```
183 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
184 {
185   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
186 }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
187 \cs_new:Nn \__enumext_level:
188 {
189   \int_to_roman:n { \__enumext_level_int }
190 }
```

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

`\__enumext_level_set:n`

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

`\__enumext_level_end:n`

```
191 \cs_new:Npn \__enumext_level_set:n #1
192 {
193   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
194   \cs_set:Nn \__enumext_level: { #1 }
195 }
196 \cs_new:Npn \__enumext_level_end:n #1
197 {
198   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
199 }
```

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

`\__enumext_if_is_int:nT`

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

`\__enumext_if_is_int:nF`

`\__enumext_if_is_int:nTF`

```
200 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
201 {
202   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
203   { \prg_return_true: }
204   { \prg_return_false: }
205 }
```

(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.

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

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

`\__enumext_zero_count_level:`

Internal function used by `check-ans` key.

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

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



## 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.

```
220 \__enumext_at_begin_document:n
221 {
222   \cs_new_eq:NN \__enumext_start_list:nn \list
223   \cs_new_eq:NN \__enumext_stop_list: \endlist
224   \cs_new_eq:NN \__enumext_item_std:w \item
225 }
```

(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.

```
226 \__enumext_at_begin_document:n
227 {
228   \cs_new_eq:NN \__enumext_minipage:w \minipage
229   \cs_new_eq:NN \__enumext_endminipage: \endminipage
230 }
```

(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.

```
231 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
232 \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”.

```
233 \cs_new_protected:Nn \__enumext_after_hyperref:
234 {
235   \IfPackageLoadedTF { hyperref }
236   {
237     \msg_info:nnn { enumext } { package-load } { hyperref }
238     \bool_set_true:N \l__enumext_hyperref_bool
239     \IfHyperBoolean{hyperfootnotes}
240     {
241       \typeout{hyperfootnotes=true}
242       \bool_set_true:N \l__enumext_footnotes_key_bool
243     }
244     { \typeout{hyperfootnotes=false} }
245   }
246   { }
```

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`.

```
247 \bool_if:NT \l__enumext_footnotes_key_bool
248 {
249   \IfPackageLoadedTF { footnotehyper }
250   {
```

```

251         \msg_info:nnn { enumext } { package-load } { footnotehyper }
252     }
253     {
254         \typeout{No ~ footnotehyper ~ load}
255         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
256         \bool_set_false:N \l__enumext_footnotes_key_bool
257     }
258 }

```

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.

```

259     \bool_if:NTF \l__enumext_hyperref_bool
260     {
261         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
262         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
263     }
264     {
265         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
266         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
267     }
268 }

```

(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.

```

269 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
270 {
271     \protected@write \@auxout { }
272     {
273         \token_to_str:N \newlabel {#1}
274         {
275             {#2}
276             \bool_if:NT \l__enumext_hyperref_bool
277             { { \thepage } {#2} {#1} }
278             { }
279         }
280     }
281     \__enumext_hypertarget:nn {#1} { }
282     \__enumext_phantomsection:
283 }

```

(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.

```

284 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
285 {
286     \cs_if_exist:cTF { c@ #2 }
287     { \msg_fatal:nnn { enumext } { counters } { #2 } }
288     {
289         \tl_set:Nn #1 { #2 }
290         \newcounter { #2 }
291     }
292 }

```

(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   293 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv    294 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi   295 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii  296 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 297 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
          298 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
          299 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
          300 \__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.

```

301 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
302 {
303   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
304   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
305 }
306 \__enumext_register_counter_style:Nn \arabic { 0 }
307 \__enumext_register_counter_style:Nn \Alph { M }
308 \__enumext_register_counter_style:Nn \alph { m }
309 \__enumext_register_counter_style:Nn \Roman { VIII }
310 \__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.

```

311 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
312 {
313   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
314   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
315 }
316 \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.

```

317 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
318 {
319   \tl_clear_new:N #1
320   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
321   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
322   \tl_map_inline:Nn \g__enumext_counter_styles_tl
323   {
324     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
325     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
326     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
327   }
328   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
329   { \tl_use:N \g__enumext_widest_label_tl }
330   \tl_set_eq:cN { the #2 } #1
331 }
332 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

(End of definition for \\_\_enumext\_label\_style:Nnn.)

## 10.10 Setting keys associated with label

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

```

333 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
334 {
335   \keys_define:nn { enumext / #1 }
336   {
337     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
338     font      .value_required:n = true,
339     labelsep   .dim_set:c = { l__enumext_labelsep_#2_dim },
340     labelsep   .initial:n  = {0.3333em},
341     labelsep   .value_required:n = true,
342     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
343     labelwidth .value_required:n = true,
344     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
345     wrap-label .initial:n  = {##1},
346     wrap-label .value_required:n = true,
347     wrap-label* .code:n = {
348       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
349       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
350     },
351     wrap-label* .value_required:n = true,
352   }
353 }
354 \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.

```

355 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
356 {
357   \keys_define:nn { enumext / #1 }
358   {
359     align .choice:,
360     align / left .code:n =
361       {
362         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
363         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
364       },
365     align / right .code:n =
366       {
367         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
368         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
369       },
370     align / center .code:n =
371       {
372         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
373         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
374       },
375     align .initial:n = left,
376     align .value_required:n = true,
377   }
378 }
379 \clist_map_inline:nn
380 {
381   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
382 }
383 { \__enumext_tmp:nn #1 }

```

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

```

384 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
385 {
386   \keys_define:nn { enumext / #1 }
387   {
388     align .choice:,
389     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
390     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },

```

```

391         align / center .code:n = \str_set:cn { \__enumext_align_label_#2_str } { c },
392         align .initial:n = left,
393         align .value_required:n = true,
394     }
395 }
396 \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.

```

397 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
398 {
399     \tl_map_inline:Nn \c__enumext_counter_style_tl
400     {
401         \regex_replace_once:nnN { \c{##1}\* }
402         { \c{##1}\cB{\u{\__enumext_ref_aux_tl}\cE} } \__enumext_ref_key_arg_tl
403     }
404 }

```

(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`.

```

405 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
406 {
407     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
408     \tl_set_eq:Nc \l__enumext_ref_aux_tl { \__enumext_counter_ \__enumext_level: _tl }
409     \__enumext_regex_label_ref_key:
410     \tl_set_eq:Nc \l__enumext_ref_aux_tl { \__enumext_the_counter_ \__enumext_level: _tl }
411     \tl_put_right:ce { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
412     {
413         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
414         { \exp_not:V \l__enumext_ref_key_arg_tl }
415     }
416 }

```

(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`.

```

417 \cs_new_protected:Nn \__enumext_use_key_ref:
418 {
419     \tl_if_empty:cF { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
420     {
421         \tl_use:c { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
422     }
423 }

```

(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`.

```

424 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
425 {

```

```

426 \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
427 \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
428 {
429   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
430   \__enumext_regex_label_ref_key:
431   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
432   \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
433   {
434     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
435     { \exp_not:V \l__enumext_ref_key_arg_tl }
436   }
437 }
438 {
439   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
440   \__enumext_regex_label_ref_key:
441   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
442   \tl_put_right:Ne \l__enumext_counter_style_for_ref_viii_tl
443   {
444     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
445     { \exp_not:V \l__enumext_ref_key_arg_tl }
446   }
447 }
448 }

```

(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*`.

```

449 \cs_new_protected:Nn \__enumext_use_key_ref_h:
450 {
451   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
452   {
453     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
454     {
455       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
456     }
457   }
458   {
459     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
460     {
461       \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
462     }
463   }
464 }

```

(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.

```

\l__enumext_label_i_tl 465 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 466 {
\l__enumext_label_iii_tl 467   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 468   {
469     label .code:n = {
470       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
471       { l__enumext_counter_#2_tl } {##1}
472       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
473       \l__enumext_current_widest_dim
474     },
475     label .initial:n = #3,
476     label .value_required:n = true,
477     ref .code:n = \__enumext_set_label_ref:n {##1},
478     ref .value_required:n = true,
479   }
480 }
481 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
482 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
483 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
484 \__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 485 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 486 {
487   \keys_define:nn { enumext / #1 }
488   {
489     label .code:n = {
490       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
491       { \l__enumext_counter_#2_tl } {##1}
492       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
493       \l__enumext_current_widest_dim
494     },
495     label .initial:n = #3,
496     label .value_required:n = true,
497     ref .code:n = \__enumext_set_label_ref_h:n {##1},
498     ref .value_required:n = true,
499   }
500 }
501 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
502 \__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.

```

503 \keys_define:nn { enumext / keyans }
504 {
505   label .code:n = {
506     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
507     { \l__enumext_counter_v_tl } {##1}
508     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
509     \l__enumext_current_widest_dim
510     \__enumext_label_style:cvn { \l__enumext_label_vii_tl }
511     { \l__enumext_counter_vii_tl } {##1}
512     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
513     \l__enumext_current_widest_dim
514   },
515   label .initial:n = (\Alph*),
516   label .value_required:n = true,
517 }

```

(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.

```

518 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
519 {
520   \__enumext_if_is_int:nTF { #3 }
521   {
522     \int_set:Nn #2 {##3}
523   }
524   {
525     \regex_match:nVT { \c{Alph} | \c{alph} } {##1}
526     { \int_set:Nn #2 { \int_from_alph:n {##3} } }
527     \regex_match:nVT { \c{Roman} | \c{roman} } {##1}
528     { \int_set:Nn #2 { \int_from_roman:n {##3} } }
529   }
530 }
531 \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.

```

532 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
533 {
534   \__enumext_if_is_int:nTF {#4}
535   {
536     \setcounter{enumX#1} { #4 }
537   }
538   {
539     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
540     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
541     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
542     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
543   }
544   \__enumext_label_width_by_box:cv
545   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
546 }
547 \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`

```

548 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
549 {
550   \keys_define:nn { enumext / #1 }
551   {
552     start .code:n = {
553       \__enumext_start_from:ccn
554       { \l__enumext_label_#2_tl }
555       { \l__enumext_start_#2_int } {##1}
556     },
557     start .initial:n = 1,
558     widest .code:n = {
559       \__enumext_widest_from:nccn {#2}
560       { \l__enumext_label_#2_tl }
561       { \l__enumext_labelwidth_#2_dim } {##1}
562     },
563     widest .value_required:n = true,
564     start .value_required:n = true,
565   }
566 }
567 \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.

```

568 \cs_set_protected:Npn \__enumext_tmp:nnnnn #1 #2 #3 #4 #5 #6
569 {
570   \keys_define:nn { enumext / #1 }
571   {
572     topsep .skip_set:c = { \l__enumext_topsep_#2_skip },
573     topsep .initial:n = {#3},
574     topsep .value_required:n = true,
575     partopsep .skip_set:c = { \l__enumext_partopsep_#2_skip },
576     partopsep .initial:n = {#4},
577     partopsep .value_required:n = true,
578     parsep .skip_set:c = { \l__enumext_parsep_#2_skip },
579     parsep .initial:n = {#5},

```

```

580     parsep      .value_required:n = true,
581     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
582     itemsep     .initial:n = {#6},
583     itemsep     .value_required:n = true,
584     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
585     noitemsep   .value_forbidden:n = true,
586     nosepe     .meta:n = {
587                     itemsep = 0pt, parsep= 0pt,
588                     topsep = 0pt, partopsep = 0pt,
589                     },
590     nosepe     .value_forbidden:n = true,
591   }
592 }

```

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

```

593 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
594 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
595 { 4.0pt plus 2.0pt minus 1.0pt }
596 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
597 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
598 { 2.0pt plus 1.0pt minus 1.0pt }
599 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
600 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
601 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
602 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
603 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
604 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
605 { 2.0pt plus 1.0pt minus 1.0pt }
606 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
607 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
608 { 4.0pt plus 2.0pt minus 1.0pt }
609 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
610 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
611 { 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.

```

612 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
613 {
614   \keys_define:nn { enumext / #1 }
615   {
616     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
617     itemindent .value_required:n = true,
618     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
619     rightmargin .value_required:n = true,
620     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
621     listparindent .value_required:n = true,
622     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
623     list-offset .value_required:n = true,
624     list-indent .code:n =
625       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
626       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
627     list-indent .value_required:n = true,
628   }
629 }
630 \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.

```

631 \cs_set_protected:Npn \__enumext_tmp:n #1
632 {
633   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
634 }
635 \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.

```

636 \cs_set_protected:Nn \__enumext_fake_item:
637 {
638   \dim_compare:nNnT
639     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
640     >
641     { \c_zero_dim }
642   {
643     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
644     {
645       \exp_not:N \mode_leave_vertical:
646       \exp_not:n { \skip_horizontal:n }
647       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } } }
648     \ignorespaces
649   }
650 }
651 }
652 \cs_set_protected:Nn \__enumext_keyans_fake_item:
653 {
654   \dim_compare:nNnT
655     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
656   {
657     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
658     {
659       \exp_not:N \mode_leave_vertical:
660       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
661     }
662   }
663 }
664 \cs_set_protected:Nn \__enumext_fake_item_vii:
665 {
666   \dim_compare:nNnT
667     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
668   {
669     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
670     {
671       \exp_not:N \mode_leave_vertical:
672       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
673     }
674   }
675 }
676 \cs_set_protected:Nn \__enumext_fake_item_viii:
677 {
678   \dim_compare:nNnT
679     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
680   {
681     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
682     {
683       \exp_not:N \mode_leave_vertical:
684       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
685     }
686   }
687 }

```

(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.

```

688 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
689 {
690   \keys_define:nn { enumext / #1 }
691   {
692     show-length .bool_set:c = { l__enumext_show_length_#2_bool },

```

```

693         show-length .initial:n = false,
694     }
695 }
696 \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* 697 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    698 {
first    699     \keys_define:nn { enumext / #1 }
        700     {
        701         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
        702         before .value_required:n = true,
        703         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
        704         before* .value_required:n = true,
        705         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
        706         after .value_required:n = true,
        707         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
        708         first .value_required:n = true,
        709     }
        710 }
        711 \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 `{⟨code⟩}` set by the `before*` key “before” the `enumext` environment is started. The `{⟨code⟩}` is executed “without” knowing any definition of the *second argument* of the list.

```

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:
712 \cs_new_protected:Nn \__enumext_before_args_exec:
713 {
714     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
715 }

```

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

```

716 \cs_new_protected:Nn \__enumext_before_keys_exec:
717 {
718     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
719 }

```

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

```

720 \cs_new_protected:Nn \__enumext_after_stop_list:
721 {
722     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
723 }

```

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

```

724 \cs_new_protected:Nn \__enumext_after_args_exec:
725 {
726     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
727 }

```

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

```

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:
728 \cs_new_protected:Nn \__enumext_before_args_exec_v:
729 {
730     \tl_use:N \l__enumext_before_starred_key_v_tl
731 }

```

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⟩`.

```
732 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
733 {
734   \tl_use:N \l__enumext_before_no_starred_key_v_tl
735 }
```

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

```
736 \cs_new_protected:Nn \__enumext_after_stop_list_v:
737 {
738   \tl_use:N \l__enumext_after_stop_list_v_tl
739 }
```

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`.

```
740 \cs_new_protected:Nn \__enumext_after_args_exec_v:
741 {
742   \tl_use:N \l__enumext_after_list_args_v_tl
743 }
```

(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.

```
744 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
745 {
746   \tl_use:N \l__enumext_before_starred_key_vii_tl
747 }
748 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
749 {
750   \tl_use:N \l__enumext_before_starred_key_viii_tl
751 }
```

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⟩`.

```
752 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
753 {
754   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
755 }
756 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
757 {
758   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
759 }
```

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

```
760 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
761 {
762   \tl_use:N \l__enumext_after_stop_list_vii_tl
763 }
764 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
765 {
766   \tl_use:N \l__enumext_after_stop_list_viii_tl
767 }
```

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`.

```
768 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
769 {
770   \tl_use:N \l__enumext_after_list_args_vii_tl
771 }
772 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
773 {
774   \tl_use:N \l__enumext_after_list_args_viii_tl
775 }
```

(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

```

776 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
777 {
778   \keys_define:nn { enumext / #1 }
779   {
780     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
781     mini-env .value_required:n = true,
782     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
783     mini-sep .initial:n = 0.3333em,
784     mini-sep .value_required:n = true,
785     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
786     columns-sep .value_required:n = true,
787     columns .int_set:c = { \__enumext_columns_#2_int },
788     columns .initial:n = 1,
789     columns .value_required:n = true,
790   }
791 }
792 \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`.

```

793 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
794 {
795   \keys_define:nn { enumext / #1 }
796   {
797     columns .initial:n = 2,
798     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
799     miniright .value_required:n = true,
800     miniright* .code:n = {
801       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
802       \keys_set:nn { enumext / #1 } { miniright = {##1} }
803     },
804     miniright* .value_required:n = true,
805   }
806 }
807 \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.

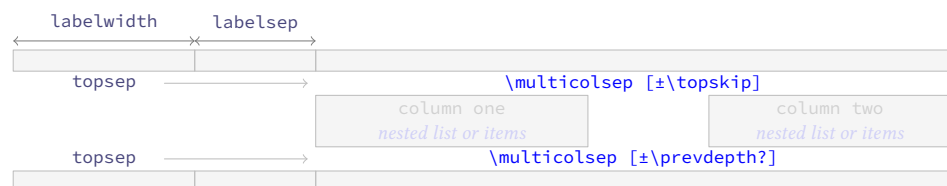


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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  $\text{\TeX}$  is in *horizontal mode*, then we will make the settings for the *vertical mode* in which `\partopsep` comes into play.

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

```

808 \cs_new_protected:Nn \__enumext_multi_set_vskip:
809 {
810   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
811   {
812     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
813   }
814   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
815   {
816     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
817   }
818   \__enumext_add_pre_parsep:
819 }

```

(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*.

```

820 \cs_new_protected:Nn \__enumext_add_pre_parsep:
821 {
822   \int_case:nn { \l__enumext_level_int }
823   {
824     { 2 }{
825       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
826       {
827         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
828       }
829     }
830     { 3 }{
831       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
832       {
833         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
834       }
835     }
836     { 4 }{
837       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
838       {
839         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
840       }
841     }
842   }
843 }

```

(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  $\text{\TeX}$  is in *horizontal mode* or *vertical mode*.

```

844 \cs_new_protected:Nn \__enumext_multi_addvspace:
845 {
846   \__enumext_multi_set_vskip:
847   \mode_if_vertical:T
848   {
849     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
850     {
851       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
852     }
853     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
854     {
855       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
856     }
857   }

```

```

858 \par\nopagebreak
859 \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: \skip } }
860 }

```

(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`.

```

861 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
862 {
863   \skip_set:Nn \l__enumext_multicols_above_v_skip
864   {
865     \l__enumext_topsep_v_skip
866   }
867   \skip_set:Nn \l__enumext_multicols_below_v_skip
868   {
869     \l__enumext_topsep_v_skip
870   }
871 }
872 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
873 {
874   \l__enumext_keyans_multi_set_vskip:
875   \mode_if_vertical:T
876   {
877     \skip_add:Nn \l__enumext_multicols_above_v_skip
878     {
879       \skip_use:N \l__enumext_partopsep_v_skip
880     }
881     \skip_add:Nn \l__enumext_multicols_below_v_skip
882     {
883       \skip_use:N \l__enumext_partopsep_v_skip
884     }
885   }
886   \par\nopagebreak
887   \addvspace{ \l__enumext_multicols_above_v_skip }
888 }

```

(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.

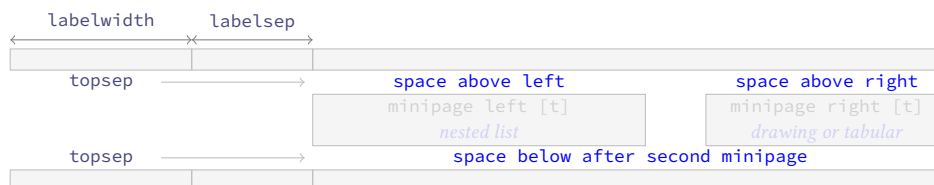


Figure 8: Representation of the `\minipage` spacing adjustment for a nested level.

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.

```

889 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
890 {
891     \__enumext_minipage:w [ t ] { #1 }
892     \legacy_if_gset_false:n { @minipage }
893     \vspace { 0pt }
894 }
895 { \__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`.

```

896 \cs_new_protected:Nn \__enumext_mini_set_vskip:
897 {
898     \int_compare:nNnTF
899     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
900     {

```

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.

```

901     \skip_if_eq:nnTF
902     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
903     {
904         \skip_set:Nn \l__enumext_minipage_left_skip
905         {
906             -0.150\box_dp:N \strutbox
907         }
908         \skip_set:Nn \l__enumext_minipage_right_skip
909         {
910             0.695\box_dp:N \strutbox
911         }
912         \skip_set:Nn \l__enumext_minipage_after_skip
913         {
914             \box_dp:N \strutbox
915         }
916         \__enumext_zero_parsep:
917     }
918     {
919         \skip_set:Nn \l__enumext_minipage_left_skip
920         {
921             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
922         }
923         \skip_set:Nn \l__enumext_minipage_right_skip
924         {
925             0.695\box_dp:N \strutbox
926         }
927         \skip_set:Nn \l__enumext_minipage_after_skip
928         {
929             1.85\box_dp:N \strutbox
930             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
931         }
932     }
933 }
934 {

```

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.

```

935     \skip_if_eq:nnTF
936     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
937     {
938         \skip_set:Nn \l__enumext_minipage_left_skip

```

```

939         {
940             0.5\box_dp:N \strutbox
941             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
942         }
943     \skip_set:Nn \l__enumext_minipage_right_skip
944     {
945         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
946     }
947     \skip_set:Nn \l__enumext_minipage_after_skip
948     {
949         1.6\box_dp:N \strutbox
950     }
951 }
952 {
953     \skip_set:Nn \l__enumext_minipage_left_skip
954     {
955         0.5875\box_dp:N \strutbox
956         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
957     }
958     \skip_set:Nn \l__enumext_minipage_right_skip
959     {
960         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
961         + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
962     }
963     \skip_set:Nn \l__enumext_minipage_after_skip
964     {
965         0.325\box_dp:N \strutbox
966         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
967     }
968 }
969 }
970 }

```

(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.

```

971 \cs_new_protected:Nn \__enumext_zero_parsep:
972 {
973     \int_case:nn { \l__enumext_level_int }
974     {
975         { 2 }{
976             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
977             {
978                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
979             }
980         }
981         { 3 }{
982             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
983             {
984                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
985             }
986         }
987         { 4 }{
988             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
989             {
990                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
991             }
992         }
993     }
994 }

```

(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*.

```

995 \cs_new_protected:Nn \__enumext_mini_addvspace:
996 {

```

```

997 \__enumext_mini_set_vskip:
998 \mode_if_vertical:T
999 {
1000   \skip_add:Nn \l__enumext_minipage_left_skip
1001   {
1002     \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1003   }
1004   \skip_add:Nn \l__enumext_minipage_after_skip
1005   {
1006     \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1007   }
1008 }
1009 \par\nopagebreak
1010 \addvspace { \l__enumext_minipage_left_skip }
1011 }

```

(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`.

```

1012 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1013 {
1014   \skip_zero_new:N \l__enumext_minipage_after_skip
1015   \skip_zero_new:N \l__enumext_minipage_left_skip
1016   \skip_zero_new:N \l__enumext_minipage_right_skip
1017   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1018   {
1019     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1020     {
1021       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1022       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1023       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1024       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1025       {
1026         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1027       }
1028     }
1029     {
1030       \skip_set:Nn \l__enumext_minipage_left_skip
1031       {
1032         \skip_use:N \l__enumext_topsep_v_skip
1033       }
1034       \skip_set:Nn \l__enumext_minipage_right_skip
1035       {
1036         0.705\box_dp:N \strutbox
1037       }
1038       \skip_set:Nn \l__enumext_minipage_after_skip
1039       {
1040         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1041       }
1042     }
1043   }
1044   {
1045     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1046     {
1047       \skip_set:Nn \l__enumext_minipage_left_skip
1048       {
1049         0.5\box_dp:N \strutbox
1050         + \l__enumext_partopsep_v_skip
1051       }
1052       \skip_set:Nn \l__enumext_minipage_right_skip
1053       {
1054         \l__enumext_partopsep_v_skip
1055       }
1056       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1057     }
1058     {
1059       \skip_set:Nn \l__enumext_minipage_left_skip
1060       {

```

```

1061         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1062     }
1063     \skip_set:Nn \l__enumext_minipage_right_skip
1064     {
1065         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1066     }
1067     \skip_set:Nn \l__enumext_minipage_after_skip
1068     {
1069         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1070     }
1071 }
1072 }
1073 }

```

(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`.

```

1074 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1075 {
1076     \__enumext_keyans_mini_set_vskip:
1077     \mode_if_vertical:T
1078     {
1079         \skip_add:Nn \l__enumext_minipage_left_skip
1080         {
1081             \l__enumext_partopsep_v_skip
1082         }
1083         \skip_add:Nn \l__enumext_minipage_after_skip
1084         {
1085             \l__enumext_partopsep_v_skip
1086         }
1087     }
1088     \par\nopagebreak
1089     \addvspace { \l__enumext_minipage_left_skip }
1090 }

```

(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*`.

```

1091 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1092 {
1093     \skip_zero_new:N \l__enumext_minipage_left_skip
1094     \skip_gzero_new:N \g__enumext_minipage_right_skip
1095     \skip_gzero_new:N \g__enumext_minipage_after_skip
1096     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1097     {
1098         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1099         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1100     }
1101     {
1102         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1103         \skip_gset:Nn \g__enumext_minipage_right_skip
1104         {
1105             \l__enumext_topsep_vii_skip
1106         }
1107         \skip_gset:Nn \g__enumext_minipage_after_skip
1108         {
1109             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1110         }
1111     }
1112 }
1113 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1114 {
1115     \skip_zero_new:N \l__enumext_minipage_after_skip

```

```

1116 \skip_zero_new:N \l__enumext_minipage_left_skip
1117 \skip_zero_new:N \l__enumext_minipage_right_skip
1118 \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1119 {
1120   \skip_set:Nn \l__enumext_minipage_left_skip
1121   {
1122     0.5\box_dp:N \strutbox
1123   }
1124   \skip_set:Nn \l__enumext_minipage_right_skip
1125   {
1126     \l__enumext_partopsep_viii_skip
1127   }
1128   \skip_set:Nn \l__enumext_minipage_after_skip
1129   {
1130     1.6\box_dp:N \strutbox
1131   }
1132 }
1133 {
1134   \skip_set:Nn \l__enumext_minipage_left_skip
1135   {
1136     0.5875\box_dp:N \strutbox
1137   }
1138   \skip_set:Nn \l__enumext_minipage_right_skip
1139   {
1140     \l__enumext_topsep_viii_skip
1141   }
1142   \skip_set:Nn \l__enumext_minipage_after_skip
1143   {
1144     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1145   }
1146 }
1147 }

```

(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.

```

1148 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1149 {
1150   \__enumext_mini_set_vskip_vii:
1151   \par\nopagebreak
1152   \addvspace { \l__enumext_minipage_left_skip }
1153 }
1154 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1155 {
1156   \__enumext_mini_set_vskip_viii:
1157   \par\nopagebreak
1158   \addvspace { \l__enumext_minipage_left_skip }
1159 }

```

(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.

```

1160 \NewDocumentCommand \miniright { s }
1161 {
1162   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1163   {
1164     \msg_error:nnn { enumext } { wrong-miniright-place }

```



```

1165     }
1166     \int_compare:nNt { \__enumext_level_int } = { 0 }
1167     {
1168         \msg_error:nnn { enumext } { wrong-miniright-place }
1169     }
1170     \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1171     {
1172         \__enumext_keyans_mini_right_cmd:n {#1}
1173     }
1174     { \__enumext_mini_right_cmd:n {#1} }
1175 }

```

(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.

```

1176 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1177 {
1178     \dim_compare:nNtF
1179     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1180     {
1181         \__enumext_multicols_stop:
1182         \end{\__enumext_mini_env*}
1183         \hfill
1184         \begin{\__enumext_mini_env*}
1185         { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1186         \par\addvspace { \__enumext_minipage_right_skip }
1187         \bool_if:nF {#1}
1188         {
1189             \centering
1190         }
1191         \int_gzero:N \g__enumext_minipage_stat_int
1192     }
1193     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1194 }

```

(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.

```

1195 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1196 {
1197     \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1198     {
1199         \__enumext_keyans_multicols_stop:
1200         \end{\__enumext_mini_env*}
1201         \hfill
1202         \begin{\__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1203         \par\addvspace { \__enumext_minipage_right_skip }
1204         \bool_if:nF {#1}
1205         {
1206             \centering
1207         }
1208         \int_gzero:N \g__enumext_minipage_stat_int
1209     }
1210     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1211 }

```

(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* 1212 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1213 {
below* 1214 \keys_define:nn { enumext / #1 }
1215 {
1216     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1217     above .value_required:n = true,
1218     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1219             \keys_set:nn { enumext / #1 } { above = {##1} },
1220     above* .value_required:n = true,
1221     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1222     below .value_required:n = true,
1223     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1224             \keys_set:nn { enumext / #1 } { below = {##1} },
1225     below* .value_required:n = true,
1226 }
1227 }
1228 \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.

```

1229 \cs_new_protected:Nn \__enumext_vspace_above:
1230 {
1231     \skip_if_eq:nnF
1232     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1233     {
1234         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1235         {
1236             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1237         }
1238         {
1239             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1240         }
1241     }
1242 }

```

(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.

```

1243 \cs_new_protected:Nn \__enumext_vspace_below:
1244 {
1245     \skip_if_eq:nnF
1246     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1247     {
1248         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1249         {
1250             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1251         }
1252         {
1253             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1254         }
1255     }
1256 }

```

(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.

```

1257 \cs_new_protected:Nn \__enumext_vspace_above_v:
1258 {
1259     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1260     {
1261         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1262         {
1263             \vspace*{ \l__enumext_vspace_above_v_skip }
1264         }
1265         { \vspace { \l__enumext_vspace_above_v_skip } }
1266     }
1267 }
```

(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.

```

1268 \cs_new_protected:Nn \__enumext_vspace_below_v:
1269 {
1270     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1271     {
1272         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1273         {
1274             \vspace*{ \l__enumext_vspace_below_v_skip }
1275         }
1276         { \vspace { \l__enumext_vspace_below_v_skip } }
1277     }
1278 }
```

(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:`

```

1279 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1280 {
1281     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1282     {
1283         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1284         {
1285             \vspace*{ \l__enumext_vspace_above_vii_skip }
1286         }
1287         { \vspace { \l__enumext_vspace_above_vii_skip } }
1288     }
1289 }
1290 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1291 {
1292     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1293     {
1294         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1295         {
1296             \vspace*{ \l__enumext_vspace_above_viii_skip }
1297         }
1298         { \vspace { \l__enumext_vspace_above_viii_skip } }
1299     }
1300 }
```

(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:`

```

1301 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1302 {
1303     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1304     {
1305         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1306         {
1307             \vspace*{ \l__enumext_vspace_below_vii_skip }

```

```

1308     }
1309     { \vspace { \l__enumext_vspace_below_vii_skip } }
1310   }
1311 }
1312 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1313 {
1314   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1315   {
1316     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1317     {
1318       \vspace*{ \l__enumext_vspace_below_viii_skip }
1319     }
1320     { \vspace { \l__enumext_vspace_below_viii_skip } }
1321   }
1322 }

```

(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.

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

```

resume
resume*
1323 \keys_define:nn { enumext / level-1 }
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:,
1328   resume .value_forbidden:n = true,
1329   resume* .code:n = \__enumext_resume_counter_star:,
1330   resume* .value_forbidden:n = true,
1331 }
1332 \keys_define:nn { enumext / enumext* }
1333 {
1334   save-ans .code:n = \__enumext_storing_set:n {#1},
1335   save-ans .value_required:n = true,
1336   resume .code:n = \__enumext_resume_counter_vii:,
1337   resume .value_forbidden:n = true,
1338 }

```

(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, if it does not exist it will create it globally.

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.

```

1339 \cs_new_protected:Npn \__enumext_storing_set:n #1
1340 {
1341   \tl_set:Nx \l__enumext_store_name_tl {#1}
1342   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1343   {
1344     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1345   }
1346   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1347   {
1348     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1349   }
1350   \bool_set_true:N \l__enumext_store_active_bool
1351   \int_if_exist:cF { g__enumext_resume_#1_int }
1352   {
1353     \int_new:c { g__enumext_resume_#1_int }
1354   }
1355   \bool_if:NT \l__enumext_check_ans_bool
1356   {
1357     \__enumext_check_ans_int:n {#1}
1358   }
1359 }

```

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

`\__enumext_resume_counter:` 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`.

```

1360 \cs_new_protected:Nn \__enumext_resume_counter:
1361 {
1362   \bool_if:NT \l__enumext_store_active_bool
1363   {
1364     \int_gset:Nn \g__enumext_resume_int
1365     {
1366       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1367     }
1368   }
1369   \bool_set_true:N \l__enumext_resume_bool
1370 }
1371 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1372 {
1373   \bool_if:NT \l__enumext_store_active_bool
1374   {
1375     \int_gset:Nn \g__enumext_resume_int
1376     {
1377       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1378     }
1379   }
1380   \bool_set_true:N \l__enumext_resume_vii_bool
1381 }

```

(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 1382 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1383 {
1384   \keys_define:nn { enumext / #1 }
1385   {
1386     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1387     check-ans .initial:n = false,
1388     no-store .code:n = {
1389       \bool_set_false:N \l__enumext_store_ans_bool
1390       \bool_set_false:N \l__enumext_check_ans_bool
1391     },
1392     no-store .value_forbidden:n = true,
1393   }
1394 }
1395 \clist_map_inline:nn
1396 {
1397   level-1, level-2, level-3, level-4, enumext*
1398 }
1399 { \__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`

```

1400 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1401 {
1402   \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1403   { \int_new:c { g__enumext_count_#1_item_ans_int } }
1404   \int_if_exist:cF { g__enumext_count_#1_i_int }
1405   { \int_new:c { g__enumext_count_#1_i_int } }
1406   \int_if_exist:cF { g__enumext_count_#1_ii_int }
1407   { \int_new:c { g__enumext_count_#1_ii_int } }
1408   \int_if_exist:cF { g__enumext_count_#1_iii_int }
1409   { \int_new:c { g__enumext_count_#1_iii_int } }
1410   \int_if_exist:cF { g__enumext_count_#1_iv_int }
1411   { \int_new:c { g__enumext_count_#1_iv_int } }
1412   \int_if_exist:cF { g__enumext_count_#1_vii_int }
1413   { \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.

```

1414   \bool_lazy_all:nTF
1415   {
1416     { \g__enumext_starred_bool }
1417     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1418   }
1419   {
1420     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1421   }
1422   {
1423     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1424   }
1425   \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1426   \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1427   \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1428   \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1429   \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1430   \int_gset_eq:Nc \g__enumext_count_item_with_ans_int { g__enumext_count_#1_item_ans_int }
1431 }

```

(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.

```

1432 \cs_new_protected:Nn \__enumext_check_ans_count:
1433 {
1434   \bool_if:NT \l__enumext_check_ans_bool
1435   {
1436     \bool_if:NTF \l__enumext_store_ans_bool
1437     {
1438       \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1439       { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1440     }
1441     { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1442   }
1443 }

```

(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:`

```

1444 \cs_new_protected:Nn \__enumext_check_ans_active:
1445 {
1446   \int_set:Nn \l__enumext_compare_items_ans_int
1447   {
1448     \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1449     - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1450   }
1451   \int_compare:nNnTF
1452   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1453   {
1454     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1455   }
1456   {
1457     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1458   }

```

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

```

1459   \__enumext_zero_count_level:
1460 }

1461 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1462 {
1463   \int_set:Nn \l__enumext_compare_items_ans_int
1464   {
1465     \g__enumext_count_item_all_int - \g__enumext_count_item_i_int
1466     - \g__enumext_count_item_ii_int - \g__enumext_count_item_iii_int
1467     - \g__enumext_count_item_iv_int
1468   }
1469   \int_compare:nNnTF
1470   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1471   {
1472     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1473   }
1474   {
1475     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1476   }
1477   \__enumext_zero_count_level:
1478 }

```

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

### 10.23 Keys and functions associated with storage

We add the keys `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*`.

```

1479 \cs_set_protected:Npn \__enumext_tmp:n #1
1480 {
1481   \keys_define:nn { enumext / #1 }

```



```

1482     {
1483       wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,
1484       wrap-ans .initial:n = \fbox{##1},
1485       wrap-ans .value_required:n = true,
1486       mark-ans .code:n = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1487       mark-ans .initial:n = \textasteriskcentered,
1488       mark-ans .value_required:n = true,
1489       mark-pos .choice:,
1490       mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1491       mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1492       mark-pos .initial:n = right,
1493       mark-pos .value_required:n = true,
1494       show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1495         \bool_set_false:N \l__enumext_show_position_bool,
1496       show-ans .value_forbidden:n = true,
1497       show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1498         \bool_set_false:N \l__enumext_show_answer_bool,
1499       show-pos .value_forbidden:n = true,
1500       mark-ref .code:n = \tl_set:Nn \l__enumext_mark_ref_sym_tl {##1},
1501       mark-ref .initial:n = \textasteriskcentered,
1502       mark-ref .value_required:n = true,
1503       store-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1504       store-ref .initial:n = false,
1505     }
1506   }
1507 \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-  
show-ans pos.

```

1508 \cs_set_protected:Npn \__enumext_tmp:n #1
1509 {
1510   \keys_define:nn { enumext / #1 }
1511   {
1512     mark-pos .choice:,
1513     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1514     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1515     mark-pos .initial:n = right,
1516     mark-pos .value_required:n = true,
1517     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1518       \bool_set_false:N \l__enumext_show_position_bool,
1519     show-ans .value_forbidden:n = true,
1520     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1521       \bool_set_false:N \l__enumext_show_answer_bool,
1522     show-pos .value_forbidden:n = true,
1523   }
1524 }
1525 \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.

```

1526 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1527 {
1528   \keys_define:nn { enumext / #1 }
1529   {
1530     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1531       \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1532       \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1533       {
1534         columns = \exp_not:v { l__enumext_store_columns_#2_int },
1535       },,
1536     columns* .value_required:n = true,
1537     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1538       \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1539       \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1540       {

```

```

1541             columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dir
1542             },
1543             columns-sep* .value_required:n = true,
1544         }
1545     }
1546     \clist_map_inline:nn
1547     {
1548         {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1549     }
1550     { \__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 *prop list* defined by `save-ans` key. The “*stored content*” is retrieved by means of the `\getkeyans` command.

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

```

1551 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1552 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1553 {
1554     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1555     {
1556         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \c_one_int }
1557     }
1558     { #1 }
1559 }
1560 \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 *sequence* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

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

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

```

1561 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1562 {
1563     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1564 }
1565 \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`.

```

1566 \cs_new_protected:Nn \__enumext_store_level_open:
1567 {
1568     \bool_if:NF \l__enumext_store_ans_bool
1569     {
1570         \tl_if_empty:cTF { l__enumext_store_opt_ \l__enumext_level: _tl }
1571         {
1572             \__enumext_store_addto_seq:n
1573             {
1574                 \item \begin{enumext}
1575             }
1576         }
1577         {
1578             \tl_put_left:cn { l__enumext_store_opt_ \l__enumext_level: _tl }
1579             {
1580                 \item \begin{enumext} [

```

```

1581         }
1582         \tl_put_right:cn { \l__enumext_store_opt_ \l__enumext_level: _tl }
1583         {
1584             ]
1585         }
1586         \l__enumext_store_addto_seq:v { \l__enumext_store_opt_ \l__enumext_level: _tl }
1587     }
1588 }
1589 }
1590 \cs_new_protected:Nn \l__enumext_store_level_close:
1591 {
1592     \bool_if:NF \l__enumext_store_ans_bool
1593     {
1594         \l__enumext_store_addto_seq:n { \end{enumext} }
1595     }
1596 }

```

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

\l\_\_enumext\_store\_level\_open\_vii:  
\l\_\_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.

```

1597 \cs_new_protected:Nn \l__enumext_store_level_open_vii:
1598 {
1599     \bool_if:NF \l__enumext_store_ans_bool
1600     {
1601         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1602         {
1603             \l__enumext_store_addto_seq:n
1604             {
1605                 \item \mode_leave_vertical:
1606                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1607                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1608             }
1609         }
1610         {
1611             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1612             {
1613                 \item \mode_leave_vertical:
1614                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1615                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1616                 ]
1617                 \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1618                 {
1619                     ]
1620                 }
1621                 \l__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1622             }
1623         }
1624     }
1625 \cs_new_protected:Nn \l__enumext_store_level_close_vii:
1626 {
1627     \bool_if:NF \l__enumext_store_ans_bool
1628     {
1629         \l__enumext_store_addto_seq:n { \end{enumext*} }
1630     }
1631 }

```

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

#### 10.23.4 Function for show marks and position

\l\_\_enumext\_print\_keyans\_box:NN  
\l\_\_enumext\_print\_keyans\_box:cc

The function `\l__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`

```

1632 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1633 {
1634   \mode_leave_vertical:
1635   \skip_horizontal:n { -\dim_use:N #2 }
1636   \makebox[0pt][ r ]
1637   {
1638     \makebox[ \dim_use:N #1 ][ \__enumext_mark_position_str ]
1639     {
1640       \tl_use:N \__enumext_mark_answer_sym_tl
1641     }
1642   }
1643   \skip_horizontal:n { \dim_use:N #2 }
1644 }
1645 \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`.

```

1646 \keys_define:nn { enumext / anskey }
1647 {
1648   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
1649   item-sym* .value_required:n = true,
1650   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
1651   item-pos* .value_required:n = true,
1652   store-brk .bool_set:N = \__enumext_store_columns_break_bool,
1653   store-brk .default:n = true,
1654   store-brk .value_forbidden:n = true,
1655 }

```

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)*.

```

1656 \NewDocumentCommand \anskey { d() s o +m }
1657 {
1658   \bool_if:NF \__enumext_store_active_bool
1659   {
1660     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1661   }
1662   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
1663   {
1664     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1665   }
1666   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1667   {
1668     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1669   }
1670   \group_begin:
1671     \bool_if:NF \__enumext_store_ans_bool
1672     {
1673       \bool_if:NT \__enumext_check_ans_bool
1674       {
1675         \int_gincr:N \g__enumext_count_item_with_ans_int
1676       }
1677       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1678     }
1679   \group_end:
1680 }

```

(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 `\__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.

```

1681 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1682 {
1683   \__enumext_store_addto_prop:n {#4}
1684   \bool_if:NT \l__enumext_store_ref_key_bool
1685   {
1686     \__enumext_store_internal_ref:
1687   }
1688   \__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`.

```

1689   \tl_clear:N \l__enumext_store_anskey_arg_tl
1690   \tl_if_novalue:nF {#3}
1691   {
1692     \keys_set:nn { enumext / anskey } {#3}
1693   }
1694   \bool_lazy_and:nnT
1695   { \l__enumext_store_columns_break_bool }
1696   { \bool_not_p:n { \l__enumext_starred_bool } }
1697   {
1698     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1699   }
1700   \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).

```

1701   \tl_if_novalue:nF {#1}
1702   {
1703     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1704     \bool_if:NT \l__enumext_starred_bool
1705     {
1706       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1707       {
1708         ( \exp_not:V \l__enumext_store_columns_join_int )
1709       }
1710     }
1711   }

```

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`.

```

1712   \bool_if:nTF {#2}
1713   {
1714     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1715     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1716     {
1717       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1718       {
1719         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1720       }
1721     }
1722     \dim_compare:nT
1723     {
1724       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1725     }
1726     {
1727       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1728       {
1729         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1730       }
1731     }
1732     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1733   }
1734   {
1735     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1736   }

```

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`.

```

1737 \bool_lazy_and:nnT
1738 { \l__enumext_store_ref_key_bool }
1739 { \l__enumext_hyperref_bool }
1740 {
1741   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1742   {
1743     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1744     { \exp_not:V \l__enumext_mark_ref_sym_tl }
1745   }
1746 }
1747 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1748 }

```

(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 `\l__enumext_newlabel_arg_two_tl`.

```

1749 \cs_new_protected:Nn \__enumext_store_internal_ref:
1750 {
1751   \cs_set_protected:Npn \__enumext_tmp:n ##1
1752   {
1753     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1754     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1755     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1756     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1757   }
1758   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1759   \cs_set:Npn \__enumext_tmp:n ##1
1760   { . \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

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

```

1761 \bool_lazy_all:nT
1762 {
1763   { \g__enumext_starred_bool }
1764   { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1765 }
1766 {
1767   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1768   { \tl_use:N \l__enumext_label_copy_vii_tl }
1769 }
1770 \bool_lazy_all:nT
1771 {
1772   { \l__enumext_standar_bool }
1773   { \g__enumext_starred_bool }
1774   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1775 }
1776 {
1777   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1778   {
1779     \tl_use:N \l__enumext_label_copy_vii_tl
1780     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1781     }
1782 }

```

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.

```

1783 \bool_lazy_all:nT
1784 {
1785   { \l__enumext_standar_bool }
1786   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1787   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1788   { \bool_not_p:n { \l__enumext_starred_bool } }
1789 }
1790 {
1791   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

1792         {
1793             \tl_use:N \l__enumext_label_copy_i_tl
1794             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1795         }
1796     }
1797     \cs_set:Npn \__enumext_tmp:n ##1
1798     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1799     \bool_lazy_all:nT
1800     {
1801         { \l__enumext_standar_bool }
1802         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1803         { \bool_not_p:n { \g__enumext_starred_bool } }
1804         { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1805     }
1806     {
1807         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1808         {
1809             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1810             . \tl_use:N \l__enumext_label_copy_vii_tl
1811         }
1812     }

```

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

```

1813     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1814     {
1815         \l__enumext_store_name_tl \c_colon_str
1816         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1817     }

```

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.

```

1818     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1819     {
1820         \__enumext_newlabel:nn
1821         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1822         { \l__enumext_newlabel_arg_two_tl }
1823     }
1824     \l__enumext_store_write_aux_file_tl
1825 }

```

(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.

```

1826 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1827 {
1828     \par
1829     \bool_if:NT \l__enumext_starred_bool
1830     {
1831         \cs_set:Nn \__enumext_level: { vii }
1832     }
1833     \__enumext_print_keyans_box:cc
1834     { l__enumext_labelwidth_ \__enumext_level: _dim }
1835     { l__enumext_labelsep_ \__enumext_level: _dim }
1836     \__enumext_anskey_wrapper:n { #1 }
1837 }

```

(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.

```

1838 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1839 {
1840     \bool_if:NT \l__enumext_show_answer_bool
1841     {
1842         \__enumext_store_anskey_show_wrap:n { #1 }
1843     }
1844     \bool_if:NT \l__enumext_show_position_bool

```



```

1845     {
1846       \tl_set:Nc \l__enumext_mark_answer_sym_tl
1847       {
1848         \group_begin:
1849         \exp_not:N \normalfont
1850         \exp_not:N \footnotesize [ \int_eval:n
1851         {
1852           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1853         }
1854       ]
1855       \group_end:
1856     }
1857     \__enumext_store_anskey_show_wrap:n { #1 }
1858   }
1859 }

```

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

## 10.25 Common functions for keyans, 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 current *⟨label⟩* \l\_\_enumext\_label\_v\_tl for the **keyans** environment and the current *⟨label⟩* \l\_\_enumext\_label\_vi\_tl for the **keyanspic** environment when using \item\* and \anspic\*, followed by the *contents* of the optional argument of both commands to the \l\_\_enumext\_store\_keyans\_label\_tl variable, which will be passed to the *⟨prop list⟩* defined by the **save-ans** key using the \\_\_enumext\_store\_addto\_prop:V.

```

1860 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1861 {
1862   \tl_clear:N \l__enumext_store_keyans_label_tl
1863   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1864   {
1865     \tl_put_right:Nc \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1866   }
1867   {
1868     \tl_put_right:Nc \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1869   }
1870   \tl_if_novalue:nF { #1 }
1871   {
1872     \tl_put_right:Nc \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1873   }
1874   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1875 }

```

(End of definition for \\_\_enumext\_keyans\_addto\_prop:n.)

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

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

\\_\_enumext\_keyans\_store\_ref:  
 \\_\_enumext\_keyans\_store\_ref\_aux\_i:  
 \\_\_enumext\_keyans\_store\_ref\_aux\_ii:

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

```

1876 \cs_new_protected:Nn \__enumext_keyans_store_ref:
1877 {
1878   \bool_if:NT \l__enumext_store_ref_key_bool
1879   {
1880     \cs_set_protected:Npn \__enumext_tmp:n ##1
1881     {
1882       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1883       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1884       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1885       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1886     }
1887     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1888     \__enumext_keyans_store_ref_aux_i:
1889   }
1890 }

```

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

```

1891 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
1892 {
1893   \bool_if:NT \g__enumext_starred_bool
1894   {
1895     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
1896   }
1897   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1898   {
1899     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1900     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
1901   }
1902   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1903   {
1904     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1905     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
1906   }
1907   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
1908   {
1909     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1910     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
1911   }
1912   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1913   {
1914     \l__enumext_store_name_tl \c_colon_str
1915     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1916   }
1917   \__enumext_keyans_store_ref_aux_ii:
1918 }

```

Now auxiliary function `\__enumext_keyans_store_ref_aux_ii:` save the result in the variable `\l__enumext_store_write_aux_file_tl` and finally we write in the `.aux` file.

```

1919 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
1920 {
1921   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1922   {
1923     \__enumext_newlabel:nn
1924     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1925     { \l__enumext_newlabel_arg_two_tl }
1926   }
1927   \l__enumext_store_write_aux_file_tl
1928 }

```

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

### 10.25.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n`  
`\__enumext_keyans_addto_seq_link:`

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the current  $\langle \textit{label} \rangle$  `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the  $\langle \textit{contents} \rangle$  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.

```

1929 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1930 {
1931   \tl_clear:N \l__enumext_store_keyans_label_tl
1932   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1933   {
1934     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1935   }
1936   {
1937     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1938   }
1939   \tl_if_novalue:nF { #1 }
1940   {
1941     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1942   }
1943   \__enumext_keyans_addto_seq_link:
1944 }

```

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. 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.

```

1945 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
1946 {
1947   \bool_lazy_and:nnT
1948     { \l__enumext_store_ref_key_bool }
1949     { \l__enumext_hyperref_bool }
1950   {
1951     \tl_put_right:Nx \l__enumext_store_keyans_label_tl
1952       {
1953         \hfill \exp_not:N \hyperlink
1954           {
1955             \exp_not:V \l__enumext_newlabel_arg_one_tl
1956           }
1957           { \exp_not:V \l__enumext_mark_ref_sym_tl }
1958       }
1959   }
1960   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
1961   \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1962   \bool_if:NT \l__enumext_check_ans_bool
1963   {
1964     \int_gincr:N \g__enumext_count_item_with_ans_int
1965   }
1966 }

```

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

#### 10.25.4 Check for starred commands

`\__enumext_keyans_check_ans:nn`

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

```

1967 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1968 {
1969   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1970   {
1971     \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1972   }
1973   { \tl_gclear:N \g__enumext_check_ans_item_tl }
1974 }

```

(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* `\item*` and `\position` of stored content in `\prop list` for `keyans` and `keyanspic`. Need add 1 to `\l__enumext_show_position_bool` for `keyans` environment.

```

1975 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1976 {
1977   \bool_if:NT \l__enumext_show_answer_bool
1978   {
1979     \tl_put_left:Nn \l__enumext_label_v_tl
1980       {
1981         \__enumext_print_keyans_box:NN
1982           \l__enumext_labelwidth_i_dim
1983           \l__enumext_labelsep_i_dim
1984       }
1985     \tl_if_novalue:nF { #1 }
1986     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1987   }
1988   \bool_if:NT \l__enumext_show_position_bool
1989   {
1990     \tl_set:Nx \l__enumext_mark_answer_sym_tl
1991       {
1992         \group_begin:

```

```

1993         \exp_not:N \normalfont
1994         \exp_not:N \footnotesize [ \int_eval:n
1995         {
1996             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \c_one_int
1997         }
1998         ]
1999         \group_end:
2000     }
2001     \tl_put_left:Nn \l__enumext_label_v_tl
2002     {
2003         \__enumext_print_keyans_box:NN
2004         \l__enumext_labelwidth_i_dim
2005         \l__enumext_labelsep_i_dim
2006     }
2007     \tl_if_novalue:nF { #1 }
2008     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
2009 }
2010 }

```

(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*`.

```

2011 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2012 {
2013     \keys_define:nn { enumext / #1 }
2014     {
2015         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2016         item-sym* .value_required:n = true,
2017         item-sym* .initial:n = { $\star$ },
2018         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2019         item-pos* .value_required:n = true,
2020     }
2021 }
2022 \clist_map_inline:nn
2023 {
2024     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2025 }
2026 { \__enumext_tmp:nn #1 }

```

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

## 10.27 Redefining `\footnote` command

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

```

2027 \cs_new_protected:Nn \__enumext_footnotetext:nn
2028 {
2029     \footnotetext[#1]{#2}
2030 }
2031 \cs_new_protected:Nn \__enumext_renew_footnote:
2032 {
2033     \seq_gclear:N \g__enumext_footnote_arg_seq
2034     \seq_gclear:N \g__enumext_footnote_int_seq
2035     \RenewDocumentCommand \footnote { o +m }
2036     {
2037         \tl_if_novalue:nTF {##1}
2038         {
2039             \stepcounter{footnote}
2040             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2041         }
2042         {
2043             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2044         }
2045         \footnotemark [ \g__enumext_footnote_int ]
2046         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2047         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int

```

```

2048     }
2049   }
2050   \cs_new_protected:Nn \__enumext_print_footnote:
2051   {
2052     \seq_if_empty:NF \g__enumext_footnote_int_seq
2053     {
2054       \seq_map_pairwise_function:NNN
2055       \g__enumext_footnote_int_seq
2056       \g__enumext_footnote_arg_seq
2057       \__enumext_footnotetext:nn
2058     }
2059   }

```

(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).

```

2060 \cs_new_protected:Npn \__enumext_default_item:n #1
2061 {
2062   \tl_if_novalue:nTF {#1}
2063   {
2064     \bool_if:NT \l__enumext_check_ans_bool
2065     {
2066       \int_gincr:N \g__enumext_count_item_all_int
2067       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2068     }
2069     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2070     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2071   }
2072   {
2073     \bool_set_eq:cc
2074     { l__enumext_wrap_label_ \__enumext_level: _bool }
2075     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2076     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2077   }
2078 }

```

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

`\__enumext_starred_item:nn`

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

`#1: \l__enumext_item_symbol_X_tl`

`#2: \l__enumext_item_symbol_sep_X_dim`

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `\__enumext_make_label:` (§10.29).

```

2079 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2080 {
2081   \tl_if_novalue:nF {#1}
2082   {
2083     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2084   }
2085   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2086   \tl_if_novalue:nTF {#2}
2087   {
2088     \dim_set_eq:cc
2089     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2090     { l__enumext_labelsep_ \__enumext_level: _dim }
2091   }
2092   {
2093     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2094   }
2095   \bool_if:NT \l__enumext_check_ans_bool
2096   {
2097     \int_gincr:N \g__enumext_count_item_all_int
2098     \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2099   }
2100   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2101   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2102 }

```

(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).

```

2103 \cs_new_protected:Nn \__enumext_redefine_item:
2104 {
2105   \RenewDocumentCommand \item { s o o }
2106   {
2107     \bool_if:nTF {##1}
2108     {
2109       \__enumext_starred_item:nn {##2} {##3}
2110     }
2111     { \__enumext_default_item:n {##2} }
2112   }
2113 }

```

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

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

```

2114 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2115 {
2116   \tl_if_novalue:nTF { #1 }
2117   {
2118     \bool_set_true:N \l__enumext_wrap_label_v_bool
2119     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2120   }
2121   {
2122     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2123     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2124   }
2125 }

```

(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>`.

```
2126 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2127 {
2128   \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2129   \__enumext_keyans_show_left:n { #1 }
2130   \bool_set_true:N \l__enumext_wrap_label_v_bool
2131   \__enumext_item_std:w \tl_use:N \l__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>`.

```
2132   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2133   \__enumext_keyans_addto_prop:n { #1 }
2134   \__enumext_keyans_store_ref:
2135   \__enumext_keyans_addto_seq:n { #1 }
2136 }
```

(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).

```
2137 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2138 {
2139   \RenewDocumentCommand \item { s o }
2140   {
2141     \bool_if:nTF {##1}
2142     {
2143       \peek_remove_spaces:n
2144       {
2145         \__enumext_keyans_starred_item:n {##2}
2146       }
2147     }
2148     {
2149       \__enumext_keyans_default_item:n {##2}
2150     }
2151   }
2152 }
```

(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.

```
2153 \cs_new_protected:Nn \__enumext_item_starred:
2154 {
2155   \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2156   {
2157     \mode_leave_vertical:
2158     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2159     \makebox[ 0pt ][ r ]{ \tl_use:N \g__enumext_item_symbol_tl }
2160     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2161   }
2162 }
```

(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).

```

2163 \cs_new_protected:Nn \__enumext_make_label:
2164 {
2165   \RenewDocumentCommand \makeLabel { m }
2166   {
2167     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2168     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2169     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2170     {
2171       \__enumext_item_starred:
2172       \use:c { \__enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2173     }
2174     { ##1 }
2175     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2176     \tl_gclear:N \g__enumext_item_symbol_tl
2177   }
2178 }

```

(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).

```

2179 \cs_new_protected:Nn \__enumext_keyans_make_label:
2180 {
2181   \RenewDocumentCommand \makeLabel { m }
2182   {
2183     \tl_use:N \l__enumext_label_fill_left_v_tl
2184     \tl_use:N \l__enumext_label_font_style_v_tl
2185     \bool_if:NTF { \l__enumext_wrap_label_v_bool }
2186     {
2187       \__enumext_wrapper_label_v:n { ##1 }
2188     }
2189     { ##1 }
2190     \tl_use:N \l__enumext_label_fill_right_v_tl
2191   }
2192 }

```

(End of definition for `\__enumext_keyans_make_label:`)

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

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

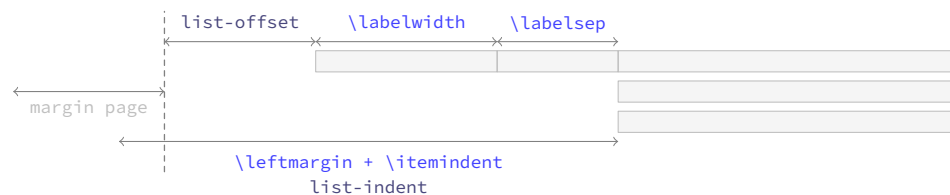


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

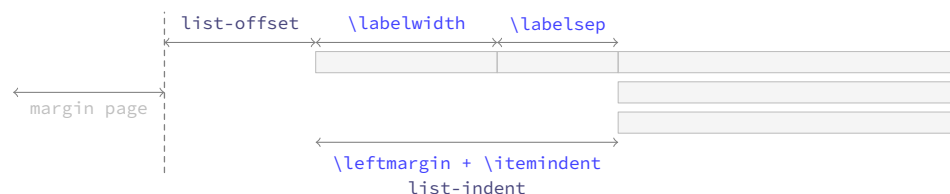


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

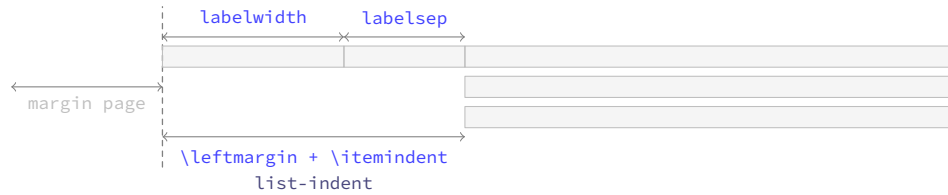
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

Figure 11: Default horizontal lengths in `enumext`.

```

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

```

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

This function is passed to `\__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§10.31).

```

2193 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2194 {
2195   \dim_compare:nNtT { #1 } < { \c_zero_dim }
2196   {
2197     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2198     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2199   }
2200   \dim_compare:nNtT { #2 } < { \c_zero_dim }
2201   {
2202     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2203     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2204   }

```

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

```

2205   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2206   \dim_compare:nNtTF { #4 } < { \c_zero_dim }
2207   {
2208     \dim_set:Nn #6 { #1 + #2 - #4 }
2209     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2210   }
2211   {
2212     \dim_compare:nNtT { #4 } = { #1 + #2 }
2213     { \dim_set:Nn #6 { \c_zero_dim } }
2214     \dim_compare:nNtT { #4 } < { #1 + #2 }
2215     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2216     \dim_compare:nNtT { #4 } > { #1 + #2 }
2217     {
2218       \dim_set:Nn #6 { -#1 - #2 + #4 }
2219       \dim_set:Nn #6 { #6*-1 }
2220     }
2221     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2222   }
2223 }
2224 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

### 10.31 Setting second argument of the lists

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

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

```

2225 \cs_set_protected:Npn \__enumext_tmp:n #1
2226 {

```

```

2227 \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2228 {
2229   \__enumext_calc_hspace:ccccc
2230   { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2231   { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2232   { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2233   { \__enumext_leftmargin_tmp_#1_bool }
2234 \clist_map_inline:nn
2235   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2236   { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2237 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2238   { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2239 \usecounter { enumX#1 }
2240 \bool_lazy_and:nnTF
2241   { \str_if_eq_p:nn {#1} { i } }
2242   { \bool_if_p:N \__enumext_resume_bool }
2243   { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2244   {
2245     \setcounter { enumX#1 }
2246     { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2247   }
2248 \str_if_eq:nnTF {#1} { v }
2249 {
2250   \__enumext_keyans_redefine_item:
2251   \__enumext_keyans_make_label:
2252   \__enumext_keyans_fake_item:
2253   \bool_if:cT { \__enumext_show_length_#1_bool }
2254   {
2255     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2256   }
2257 }
2258 {
2259   \__enumext_redefine_item:
2260   \__enumext_make_label:
2261   \__enumext_use_key_ref:
2262   \__enumext_fake_item:
2263   \bool_if:cT { \__enumext_show_length_#1_bool }
2264   {
2265     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2266   }
2267 }
2268 }
2269 }
2270 \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.

```

2271 \cs_set_protected:Npn \__enumext_tmp:n #1
2272 {
2273   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2274   {
2275     \__enumext_calc_hspace:ccccc
2276     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2277     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2278     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2279     { \__enumext_leftmargin_tmp_#1_bool }
2280 \clist_map_inline:nn
2281   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2282   { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2283 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2284   { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2285 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2286 \skip_zero:N \partopsep
2287 \usecounter { enumX#1 }
2288 \bool_lazy_and:nnTF
2289   { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2290   { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }

```

```

2291     {
2292         \setcounter { enumX#1 }
2293         { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2294     }
2295     \__enumext_use_key_ref_h:
2296     \str_if_eq:nnTF {#1} { vii }
2297     {
2298         \__enumext_fake_item_vii:
2299         \bool_if:cT { l__enumext_show_length_vii_bool }
2300         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2301     }
2302     {
2303         \__enumext_fake_item_viii:
2304         \bool_if:cT { l__enumext_show_length_#1_bool }
2305         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2306     }
2307 }
2308 }
2309 \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.

```

2310 \NewDocumentEnvironment{enumext}{ 0{ } }
2311 {
2312     \__enumext_safe_exec:
2313     \__enumext_parse_keys:n {#1}
2314     \__enumext_before_list:
2315     \__enumext_start_store_level:
2316     \__enumext_start_list:nn
2317     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2318     {
2319         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2320         \__enumext_before_keys_exec:
2321     }
2322     \__enumext_after_args_exec:
2323 }
2324 {
2325     \__enumext_stop_list:
2326     \__enumext_stop_store_level:
2327     \__enumext_after_list:
2328 }

```

(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 vars **\l\_\_enumext\_standar\_bool** and **\g\_\_enumext\_standar\_bool** to “true”, the latter only if the environment is NOT nested in the **enumext\*** environment.

```

2329 \cs_new_protected:Nn \__enumext_safe_exec:
2330 {
2331     \int_incr:N \l__enumext_level_int
2332     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2333     { \msg_fatal:nn { enumext } { list-too-deep } }
2334     \bool_set_true:N \l__enumext_standar_bool
2335     \bool_lazy_all:nT
2336     {
2337         { \bool_not_p:n { \l__enumext_starred_bool } }
2338         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2339     }
2340     {
2341         \bool_gset_true:N \g__enumext_standar_bool
2342     }
2343 }

```

(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.

```

2344 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2345 {
2346   \exp_args:Ne \keys_set:nn
2347   { enumext / level-\int_use:N \__enumext_level_int } {#1}
2348   \bool_if:NT \l__enumext_store_active_bool
2349   {
2350     \__enumext_parse_store_keys:n {#1}
2351   }
2352 }

```

(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:`.

```

2353 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2354 {
2355   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2356   {
2357     \regex_match:nnT { \b columns\b } {#1}
2358     {
2359       \int_set_eq:cc
2360       { \l__enumext_store_columns_ \__enumext_level: _int }
2361       { \l__enumext_columns_ \__enumext_level: _int }
2362       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2363       {
2364         columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2365       }
2366     }
2367   }
2368   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2369   {
2370     \regex_match:nnT { \b columns-sep\b } {#1}
2371     {
2372       \dim_set_eq:cc
2373       { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2374       { \l__enumext_columns_sep_ \__enumext_level: _dim }
2375       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2376       {
2377         columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2378       }
2379     }
2380   }
2381 }

```

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

\\_\_enumext\_start\_store\_level:

\\_\_enumext\_stop\_store\_level:

The `\__enumext_start_store_level:` and `\__enumext_stop_store_level:` functions activate the level saving mechanism for storage in *sequence* of the `\anskey` command. If `enumext` are nested in `enumext*` add `\__enumext_store_level_open:` to preserve the stored structure.

```

2382 \cs_new_protected:Nn \__enumext_start_store_level:
2383 {
2384   %\bool_lazy_and:nnT
2385   %{ \l__enumext_store_active_bool }
2386   %{ \bool_not_p:n { \l__enumext_keyans_env_bool } }
2387   %{
2388     %\int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2389     %{
2390       %\bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2391       %\__enumext_store_level_open:
2392     }
2393     %\int_compare:nNnT { \l__enumext_level_int } > { 1 }
2394     %{
2395       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2396       \__enumext_store_level_open:
2397     }
2398   }%}
2399 }
2400 \cs_new_protected:Nn \__enumext_stop_store_level:

```

```

2401 {
2402   \bool_if:cT { \__enumext_store_upper_level_ \__enumext_level: _bool }
2403   {
2404     \__enumext_store_level_close:
2405   }
2406 }

```

(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.

```

2407 \cs_new_protected:Nn \__enumext_before_list:
2408 {
2409   \__enumext_vspace_above:
2410   \__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.

```

2411 \__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.

```

2412 \dim_compare:nNtT
2413 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2414 {
2415   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2416   {
2417     \linewidth
2418     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2419     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2420   }

```

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

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

```

2421 \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
2422 \int_gincr:N \g__enumext_minipage_stat_int
2423 \__enumext_mini_addvspace:
2424 \nointerlineskip\noindent
2425 \begin{\__enumext_mini_env*}
2426 { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
2427 }
2428 \__enumext_multicols_start:
2429 }

```

(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.

```

2430 \cs_new_protected:Nn \__enumext_multicols_start:
2431 {
2432   \int_compare:nNtT
2433   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2434   {
2435     \dim_compare:nNtT
2436     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2437     {
2438       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
2439       {
2440         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
2441         + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }

```

```

2442         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2443         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2444     }
2445 }
2446 \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2447 \skip_zero:N \multicolsep
2448 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2449 {
2450     \dim_zero:N \columnseprule
2451 }

```

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.

```

2452     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2453     {
2454         \__enumext_multi_addvspace:
2455     }
2456     \raggedcolumns
2457     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2458 }
2459 }

```

(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.

```

2460 \cs_new_protected:Nn \__enumext_multicols_stop:
2461 {
2462     \int_compare:nNnT
2463     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2464     {
2465         \end{multicols}
2466         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2467         {
2468             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2469         }
2470     }

```

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.

```

2471     \bool_lazy_and:nnT
2472     { \l__enumext_check_ans_bool }
2473     { \bool_not_p:n { \g__enumext_starred_bool } }
2474     {
2475         \bool_gset_true:N \g__enumext_check_ans_show_bool
2476         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2477     }
2478 }

```

(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.

```

2479 \cs_new_protected:Nn \__enumext_after_list:
2480 {
2481     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2482     {
2483         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2484         {
2485             \msg_warning:nn { enumext } { missing-miniright }
2486             \miniright
2487         }
2488         \int_gzero:N \g__enumext_minipage_stat_int
2489         \end{__enumext_mini_env*}

```



```

2490         \par\addvspace { \l__enumext_minipage_after_skip }
2491     }
2492     { \__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.

```

2493     \__enumext_after_stop_list:
2494     \__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.

```

2495     \bool_set_false:N \l__enumext_standar_bool
2496     \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2497     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2498     {
2499         \int_gset_eq:cN
2500         { g__enumext_resume_ \l__enumext_store_name_tl _int }
2501         { \value{enumXi} }
2502     }
2503 }

```

(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`.

```

2504 \__enumext_after_env:nn {enumext}
2505 {
2506     \bool_if:NT \g__enumext_check_ans_show_bool
2507     {
2508         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2509         {
2510             \__enumext_check_ans_active:
2511         }
2512     }
2513     \bool_gset_false:N \g__enumext_check_ans_show_bool
2514     \tl_gclear:N \g__enumext_store_name_tl
2515 }

```

### 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.

```

2516 \NewDocumentEnvironment{keyans}{ 0{} }
2517 {
2518     \__enumext_keyans_safe_exec:
2519     \__enumext_keyans_parse_keys:n {#1}
2520     \__enumext_before_list_v:
2521     \__enumext_start_list:nn
2522     { \tl_use:N \l__enumext_label_v_tl }
2523     {
2524         \__enumext_list_arg_two_v:
2525         \__enumext_before_keys_exec_v:
2526     }
2527     \__enumext_after_args_exec_v:
2528 }
2529 {
2530     \__enumext_keyans_check_ans:nn { item }{ keyans }
2531     \__enumext_stop_list:
2532     \__enumext_after_list_v:
2533 }

```

(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.

```

2534 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2535 {

```

```

2536 \bool_if:NF \l__enumext_store_active_bool
2537 {
2538   \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2539 }
2540 \int_incr:N \l__enumext_keyans_level_int
2541 \bool_set_true:N \l__enumext_keyans_env_bool
2542 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2543 \bool_set_false:N \l__enumext_store_active_bool
2544 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2545 {
2546   \msg_error:nn { enumext } { keyans-nested }
2547 }
2548 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2549 {
2550   \msg_error:nn { enumext } { keyans-wrong-level }
2551 }
2552 }

```

(End of definition for `\__enumext_keyans_safe_exec:`)

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

```

2553 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2554 {
2555   \keys_set:nn { enumext / keyans } {#1}
2556 }

```

(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.

```

2557 \cs_new_protected:Nn \__enumext_before_list_v:
2558 {
2559   \__enumext_vspace_above_v:
2560   \__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.

```

2561 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2562 {
2563   \dim_set:Nn \l__enumext_minipage_left_v_dim
2564   {
2565     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2566   }

```

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*.

```

2567 \bool_set_true:N \l__enumext_minipage_active_v_bool
2568 \int_gincr:N \g__enumext_minipage_stat_int
2569 \__enumext_keyans_mini_addvspace:
2570 \nointerlineskip\noindent
2571 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2572 }

```

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

```

2573 \__enumext_keyans_multicols_start:
2574 }

```

(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.

```

2575 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2576 {
2577   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2578   {

```

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

```

2579 \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2580 {
2581   \dim_set:Nn \l__enumext_columns_sep_v_dim
2582   {
2583     (
2584       \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2585     ) / \l__enumext_columns_v_int
2586     - \l__enumext_listoffset_v_dim
2587   }
2588 }
2589 \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).

```

2590 \skip_zero:N \multicolsep
2591 \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.

```

2592 \bool_if:NF \l__enumext_minipage_active_v_bool
2593 {
2594   \__enumext_keyans_multi_addvspace:
2595 }
2596 \raggedcolumns
2597 \begin{multicols}{ \l__enumext_columns_v_int }
2598 }
2599 }

```

(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.

```

2600 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2601 {
2602   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2603   {
2604     \end{multicols}
2605     \bool_if:NF \l__enumext_minipage_active_v_bool
2606     {
2607       \par\addvspace{ \l__enumext_multicols_below_v_skip }
2608     }
2609   }
2610 }

```

(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.

```

2611 \cs_new_protected:Nn \__enumext_after_list_v:
2612 {
2613   \bool_if:NTF \l__enumext_minipage_active_v_bool
2614   {
2615     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2616     {
2617       \msg_warning:nn { enumext } { missing-miniright }
2618       \miniright
2619     }
2620     \int_gzero:N \g__enumext_minipage_stat_int
2621     \end{\__enumext_mini-env*}
2622     \par\addvspace{ \l__enumext_minipage_after_skip }
2623   }
2624   { \__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.

```
2625 \bool_set_false:N \__enumext_keyans_env_bool
2626 \__enumext_after_stop_list_v:
2627 \__enumext_vspace_below_v:
2628 }
```

(End of definition for `\__enumext_after_list_v:`)

### 10.34 The environment `keyanspic` and `\anspic`

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

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

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

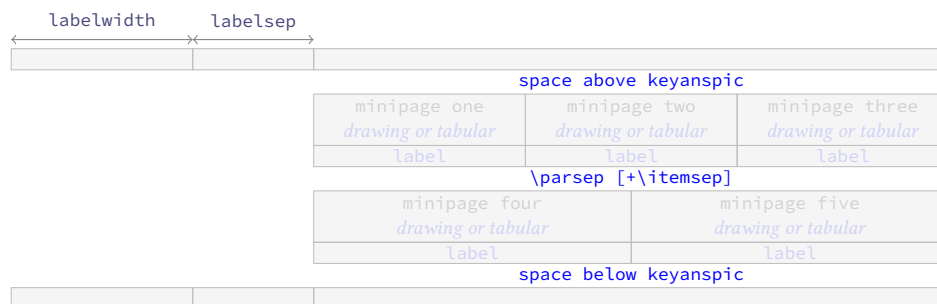


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

This implementation is adapted from the answer given by Enrico Gregorio in [How to process the body of an environment and divide it by a \macro?](#).

#### 10.34.1 The command `\anspic`

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

```
2629 \NewDocumentCommand \anspic { s o +m }
2630 {
```

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

```
2631 \bool_if:NF \__enumext_store_active_bool
2632 {
2633 \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2634 }
2635 \int_compare:nNt { \__enumext_level_int } > { 1 }
2636 {
2637 \msg_error:nn { enumext } { keyanspic-wrong-level }
2638 }
2639 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2640 {
2641 \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2642 }
```

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.

```
2643 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
2644 {
2645 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2646 }
2647 }
```

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

`\__enumext_keyans_anspic_code:nnn` The function `\__enumext_keyans_anspic_code:nnn` will be in charge of handling the “*counter*” and `\label`, which will have the same configuration as the `keyans` environment.

```
2648 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2649 {
2650 \stepcounter { enumXvi }
```

```

2651 #3 \\\
2652 \bool_if:nT { #1 }
2653 {
2654   \__enumext_keyans_addto_prop:n { #2 }
2655   \__enumext_keyans_store_ref:
2656   \__enumext_keyans_addto_seq:n { #2 }
2657   \bool_lazy_or:nnT
2658   { \__enumext_show_answer_bool }
2659   { \__enumext_show_position_bool }
2660   {
2661     \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
2662     \__enumext_keyans_show_left:n { #2 }
2663     \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
2664   }
2665 }
2666 \tl_use:N \__enumext_label_font_style_v_tl
2667 \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
2668 }

```

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

### 10.34.2 The environment `keyanspic`

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

```

2669 \NewDocumentEnvironment{keyanspic}{o}
2670 {
2671   \__enumext_keyans_pic_safe_exec:
2672   \__enumext_start_list:nn
2673   { }
2674   {
2675     \__enumext_keyans_pic_arg_two:
2676   }

```

We apply the “adjusted” vertical spacing above the environment

```

2677   \vspace { \__enumext_keyans_pic_above_skip }
2678 }

```

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

```

2679 {
2680   \tl_if_novalue:nTF { #1 }
2681   {
2682     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
2683   }
2684   { \__enumext_keyans_pic_do:n { #1 } }
2685   \__enumext_stop_list:
2686   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2687   \setcounter { enumXvi } { 0 }
2688   \vspace { \__enumext_topsep_v_skip }
2689   %\bool_set_false:N \__enumext_store_active_bool
2690 }

```

(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.

```

2691 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2692 {
2693   \int_incr:N \__enumext_keyans_pic_level_int
2694   \int_compare:nNnT { \__enumext_keyans_pic_level_int } > { 1 }
2695   {
2696     \msg_error:nn { enumext } { keyanspic-nested }
2697   }
2698 }

```

(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`.

```
2699 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2700 {
2701   \dim_compare:nNtT { #1 } < { 0pt }
2702   { \skip_set:Nn #1 { -#1 } }
2703 }
```

(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.

```
2704 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2705 {
```

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.

```
2706   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2707   \__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.

```
2708   \skip_add:Nn \parsep { \itemsep }
2709   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2710   \dim_zero:N \labelwidth
2711   \dim_zero:N \listparindent
2712   \dim_zero:N \labelsep
2713   \skip_zero:N \partopsep
2714   \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.

```
2715   \__enumext_keyans_pic_skip_abs:N \parsep
2716   \skip_set:Nn \l__enumext_keyans_pic_above_skip
2717   {
2718     \box_dp:N \strutbox
2719     + \l__enumext_topsep_v_skip
2720     - \parsep
2721   }
2722   \__enumext_item_std:w \scan_stop:
2723 }
```

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

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

```
2724 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2725 {
2726   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2727 }
2728 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }
```

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

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

```
2729 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2730 {
2731   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2732   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2733   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2734   \int_step_inline:nnn
2735   { \l__enumext_keyans_pic_above_int + 1 }
2736   { \l__enumext_keyans_pic_below_int }
2737   {
2738     \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2739     \centering
```

```

2740         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2741     \__enumext_endminipage:
2742 }
2743 \par
2744 }

```

(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.

```

2745 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2746 {
2747     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2748     {
2749         \dim_set:Nn \l__enumext_columns_sep_vii_dim
2750         {
2751             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2752             / \l__enumext_columns_vii_int
2753         }
2754     }
2755     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2756     \dim_set:Nn \l__enumext_item_width_vii_dim
2757     {
2758         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2759         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2760         - \l__enumext_labelsep_vii_dim
2761     }
2762     \dim_zero_new:N \itemwidth
2763 }

```

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

The function `\__enumext_starred_joined_item_vii:n` will set the *width* of the box in which the content passed to `\item(\langle number \rangle)` will be stored together with the value of `\itemwidth`.

```

2764 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2765 {
2766     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2767     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2768     {
2769         \msg_warning:nnee { enumext } { item-joined }
2770         { \int_use:N \l__enumext_joined_item_vii_int }
2771         { \int_use:N \l__enumext_columns_vii_int }
2772         \int_set:Nn \l__enumext_joined_item_vii_int
2773         {
2774             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2775         }
2776     }
2777     \int_compare:nNnT
2778     { \l__enumext_joined_item_vii_int }
2779     >
2780     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2781     {
2782         \msg_warning:nnee { enumext } { item-joined-columns }
2783         { \int_use:N \l__enumext_joined_item_vii_int }
2784         {
2785             \int_eval:n
2786             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```



```

2787     }
2788     \int_set:Nn \l__enumext_joined_item_vii_int
2789     {
2790         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2791     }
2792 }

```

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

```

2793 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2794 {
2795     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2796     \int_decr:N \l__enumext_joined_item_aux_vii_int
2797     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2798     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2799     \dim_set:Nn \l__enumext_joined_width_vii_dim
2800     {
2801         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2802         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2803             + \l__enumext_columns_sep_vii_dim
2804             ) * \l__enumext_joined_item_aux_vii_int
2805     }
2806     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2807 }
2808 {
2809     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2810     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2811 }
2812 }

```

(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`.

```

2813 \cs_new_protected:Nn \__enumext_start_mini_vii:
2814 {
2815     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2816     {
2817         \dim_set:Nn \l__enumext_minipage_left_vii_dim
2818         {
2819             \linewidth
2820             - \l__enumext_minipage_right_vii_dim
2821             - \l__enumext_minipage_hsep_vii_dim
2822         }
2823         \bool_set_true:N \l__enumext_minipage_active_vii_bool
2824         \dim_gset_eq:NN
2825             \g__enumext_minipage_right_vii_dim
2826             \l__enumext_minipage_right_vii_dim
2827         \__enumext_mini_addvspace_vii:
2828         \nointerlineskip\noindent
2829         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2830     }
2831 }

```

(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”.

```

2832 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2833 {
2834     \bool_if:NT \l__enumext_minipage_active_vii_bool
2835     {
2836         \end{__enumext_mini_env*}
2837         \hfill
2838         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2839     }
2840 }

```

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”.

```

2841 \__enumext_after_env:nn {enumext*}
2842 {
2843   \bool_if:NT \g__enumext_minipage_active_vii_bool
2844   {
2845     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2846     \par\addvspace { \g__enumext_minipage_right_skip }
2847     \bool_if:NF \g__enumext_minipage_center_vii_bool
2848     {
2849       \centering
2850     }
2851     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2852     \end{__enumext_mini_env*}
2853     \par\addvspace{ \g__enumext_minipage_after_skip }
2854   }
2855   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2856   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2857   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2858   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2859 }

```

(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.

```

2860 \NewDocumentEnvironment{enumext*}{ o }
2861 {
2862   \__enumext_safe_exec_vii:
2863   \__enumext_parse_keys_vii:n {#1}
2864   \__enumext_before_list_vii:
2865   \__enumext_start_store_level_vii:
2866   \__enumext_start_list:nn { }
2867   {
2868     \__enumext_list_arg_two_vii:
2869     \__enumext_before_keys_exec_vii:
2870   }
2871   \__enumext_starred_columns_set_vii:
2872   \item[] \scan_stop:
2873   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2874   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2875 }
2876 {
2877   \__enumext_stop_item_tmp_vii:
2878   \__enumext_remove_extra_parsep_vii:
2879   \__enumext_stop_list:
2880   \__enumext_stop_store_level_vii:
2881   \__enumext_after_list_vii:
2882 }

```

(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`.

```

2883 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2884 {
2885   \int_incr:N \l__enumext_level_h_int
2886   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2887   {
2888     \msg_error:nn { enumext } { nested }
2889   }
2890   \bool_set_true:N \l__enumext_starred_bool
2891   \bool_lazy_all:nT
2892   {
2893     { \bool_not_p:n { \l__enumext_standar_bool } }
2894     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2895   }
2896   {
2897     \bool_gset_true:N \g__enumext_starred_bool
2898   }

```

```
2899 }
```

(End of definition for `\__enumext_safe_exec_vii:`)

```
\__enumext_parse_keys_vii:n
```

Parse [`<key = val>`] for `enumext*`. If the variable `\__enumext_store_active_bool` is true it will call the function `\__enumext_parse_store_keys_vii:n` and reprocess the keys to pass them to the storage sequence.

```
2900 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2901 {
2902   \tl_if_novalue:NF {#1}
2903   {
2904     \keys_set:nn { enumext / enumext* } {#1}
2905     \bool_if:NT \__enumext_store_active_bool
2906     {
2907       \__enumext_parse_store_keys_vii:n {#1}
2908     }
2909   }
2910 }
```

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

```
\__enumext_parse_store_keys_vii:n
```

The function `\__enumext_parse_store_keys_vii:n` searches for the values of the `columns` and `columns-sep` keys in the optional argument in `enumext*` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\__enumext_store_opt_vii_tl` which is used by the function `\__enumext_store_level_open_vii:`.

```
2911 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2912 {
2913   \bool_if:NF \__enumext_store_columns_vii_bool
2914   {
2915     \regex_match:nnT { \b columns\b } {#1}
2916     {
2917       \int_set_eq:NN
2918       \__enumext_store_columns_vii_int
2919       \__enumext_columns_vii_int
2920       \tl_put_right:Ne \__enumext_store_opt_vii_tl
2921       {
2922         columns = \exp_not:V \__enumext_store_columns_vii_int ,
2923       }
2924     }
2925   }
2926   \bool_if:NF \__enumext_store_columns_sep_vii_bool
2927   {
2928     \regex_match:nnT { \b columns-sep\b } {#1}
2929     {
2930       \dim_set_eq:NN
2931       \__enumext_store_columns_sep_vii_dim
2932       \__enumext_columns_sep_vii_dim
2933       \tl_put_right:Ne \__enumext_store_opt_vii_tl
2934       {
2935         columns-sep = \exp_not:V \__enumext_store_columns_sep_vii_dim,
2936       }
2937     }
2938   }
2939 }
```

(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`.

```
2940 \cs_new_protected:Nn \__enumext_before_list_vii:
2941 {
2942   \__enumext_vspace_above_vii:
2943   \__enumext_before_args_exec_vii:
2944   \__enumext_start_mini_vii:
2945 }
```

(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.

```

2946 \cs_new_protected:Nn \__enumext_after_list_vii:
2947 {
2948   \__enumext_stop_mini_vii:
2949   \__enumext_after_stop_list_vii:
2950   \__enumext_vspace_below_vii:
2951   \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2952   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2953   {
2954     \int_gset_eq:cN
2955       { g__enumext_resume_ \l__enumext_store_name_tl _int }
2956       { \value{enumXvii} }
2957   }
2958   \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2959   {
2960     \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2961     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2962   }
2963   \bool_gset_false:N \g__enumext_starred_bool
2964   \bool_set_false:N \l__enumext_starred_bool
2965 }

```

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

`\__enumext_start_store_level_vii:` The `\__enumext_start_store_level_vii:` and `\__enumext_stop_store_level_vii:` functions activate the level saving mechanism for storage in *(sequence)* of the `\anskey` command if `enumext*` are nested in `enumext`.

`\__enumext_stop_store_level_vii:`

```

2966 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2967 {
2968   \bool_if:NT \l__enumext_store_active_bool
2969   {
2970     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2971     {
2972       \__enumext_store_level_open_vii:
2973     }
2974   }
2975 }
2976 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2977 {
2978   \bool_if:NT \l__enumext_store_active_bool
2979   {
2980     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2981     {
2982       \__enumext_store_level_close_vii:
2983     }
2984   }
2985 }

```

(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`.

```

2986 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2987 {
2988   \__enumext_stop_item_tmp_vii:
2989   \int_incr:N \l__enumext_item_column_pos_vii_int
2990   \int_gincr:N \g__enumext_item_count_all_vii_int
2991   \__enumext_item_peek_args_vii:
2992 }

```

(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).

```

2993 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2994 {
2995   \peek_meaning:NTF (
2996     { \__enumext_joined_item_vii:w }
2997     { \__enumext_joined_item_vii:w (1) }
2998   }

```

(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`.

```

2999 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3000 {
3001   \__enumext_starred_joined_item_vii:n {#1}
3002   \peek_meaning_remove:NTF *
3003     { \__enumext_starred_item_vii:w }
3004     { \__enumext_standard_item_vii:w }
3005 }

```

(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 ]`.

```

3006 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3007 {
3008   \bool_set_false:N \l__enumext_item_starred_vii_bool
3009   \peek_meaning:NTF [
3010     {
3011       \bool_set_eq:NN
3012         \l__enumext_wrap_label_vii_bool
3013         \l__enumext_wrap_label_opt_vii_bool
3014       \__enumext_start_item_vii:w
3015     }
3016     {
3017       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3018       \legacy_if_set_true:n { @noitemarg }
3019       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3020     }
3021   }

```

(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>]`.

```

3022 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3023 {
3024   \bool_set_true:N \l__enumext_item_starred_vii_bool
3025   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3026   \peek_meaning:NTF [
3027     { \__enumext_starred_item_vii_aux_i:w }
3028     { \__enumext_starred_item_vii_aux_ii:w }
3029   }
3030 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3031 {
3032   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3033   \__enumext_starred_item_vii_aux_ii:w
3034 }
3035 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w

```

```

3036 {
3037   \peek_meaning:NTF [
3038     { \__enumext_starred_item_vii_aux_iii:w }
3039     {
3040       \dim_set_eq:NN
3041         \l__enumext_item_symbol_sep_vii_dim
3042         \l__enumext_labelsep_vii_dim
3043       \legacy_if_set_true:n { @noitemarg }
3044       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3045     }
3046   }
3047   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3048   {
3049     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3050     \legacy_if_set_true:n { @noitemarg }
3051     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3052   }

```

(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.

```

3053 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3054 {
3055   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3056   \legacy_if:nT { @noitemarg }
3057   {
3058     \legacy_if_set_false:n { @noitemarg }
3059     \legacy_if:nT { @nmbrlist }
3060     {
3061       \bool_if:NT \l__enumext_hyperref_bool
3062       {
3063         \legacy_if_set_true:n { @hyper@item }
3064       }
3065       \refstepcounter{enumXvii}
3066       % code for check-ans
3067       \bool_if:NT \l__enumext_check_ans_bool
3068       {
3069         % If true |no-store| key => nested in |enumext|
3070         \bool_if:NTF \l__enumext_store_ans_bool
3071         {
3072           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3073           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3074         }
3075         {
3076           \int_gincr:N \g__enumext_count_item_all_int
3077           \int_gincr:N \g__enumext_count_level_vii_int
3078         }
3079       }
3080     }
3081   }

```

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

```

3082   \group_begin:
3083   \lrbox{ \l__enumext_item_text_vii_box }
3084   \bool_if:NF \l__enumext_footnotes_key_bool
3085   {

```

```

3086         \_enumext_renew_footnote:
3087     }
3088     \bool_if:NT \l__enumext_item_starred_vii_bool
3089     {
3090         \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3091         {
3092             \tl_gset_eq:NN
3093             \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3094         }
3095         \mode_leave_vertical:
3096         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3097         \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3098         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3099         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3100     }
3101     \group_begin:
3102     \tl_use:N \l__enumext_label_font_style_vii_tl
3103     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3104     {
3105         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3106         { \_enumext_wrapper_label_vii:n {#1} }
3107     }
3108     {
3109         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3110     }
3111     \group_end:
3112     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3113     \tl_use:N \l__enumext_after_list_args_vii_tl
3114     \_enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3115     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3116     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3117     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3118 }

```

(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.

```

3119 \cs_new_protected_nopar:Nn \_enumext_stop_item_vii:
3120 {
3121     \_enumext_endminipage:
3122     \endlrbox
3123     \group_end:
3124     \box_set_wd:Nn \l__enumext_item_text_vii_box
3125     {
3126         \l__enumext_joined_width_vii_dim
3127         + \l__enumext_labelwidth_vii_dim
3128         + \l__enumext_labelsep_vii_dim
3129     }
3130     \int_set:Nn \hbadness { 10000 }
3131     \box_use:N \l__enumext_item_text_vii_box
3132     \bool_if:NF \l__enumext_footnotes_key_bool
3133     {
3134         \_enumext_print_footnote:
3135     }
3136     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3137     {
3138         \par\noindent
3139         \int_zero:N \l__enumext_item_column_pos_vii_int
3140     }
3141     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3142 }

```

(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.

```

3143 \cs_new_protected:Nn \_enumext_remove_extra_parsep_vii:
3144 {
3145     \int_compare:nNnT

```



```

3146     {
3147         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3148     }
3149     =
3150     { \c_zero_int }
3151     {
3152         \par
3153         \vspace{ -\l__enumext_itemsep_vii_skip }
3154         \int_gzero:N \g__enumext_item_count_all_vii_int
3155     }
3156 }

```

(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`.

```

3157 \__enumext_after_env:nn {enumext*}
3158 {
3159     \bool_if:NT \g__enumext_check_ans_show_h_bool
3160     {
3161         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3162         {
3163             \__enumext_check_ans_active_vii:
3164         }
3165     }
3166     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3167     \tl_gclear:N \g__enumext_store_name_tl
3168 }

```

## 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.

```

3169 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3170 {
3171     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3172     {
3173         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3174         {
3175             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3176             / \l__enumext_columns_viii_int
3177         }
3178     }
3179     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3180     \dim_set:Nn \l__enumext_item_width_viii_dim
3181     {
3182         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3183         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3184         - \l__enumext_labelsep_viii_dim
3185     }
3186     \dim_zero_new:N \itemwidth
3187 }

```

(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`.

```

3188 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3189 {
3190     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3191     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3192     {
3193         \msg_warning:nnee { enumext } { item-joined }
3194         { \int_use:N \l__enumext_joined_item_viii_int }
3195         { \int_use:N \l__enumext_columns_viii_int }
3196         \int_set:Nn \l__enumext_joined_item_viii_int
3197         {
3198             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3199         }
3200     }

```

```

3201 \int_compare:nNnT
3202 { \l__enumext_joined_item_viii_int }
3203 >
3204 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3205 {
3206   \msg_warning:nnee { enumext } { item-joined-columns }
3207   { \int_use:N \l__enumext_joined_item_viii_int }
3208   {
3209     \int_eval:n
3210     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3211   }
3212   \int_set:Nn \l__enumext_joined_item_viii_int
3213   {
3214     \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3215   }
3216 }
3217 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3218 {
3219   \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3220   \int_decr:N \l__enumext_joined_item_aux_viii_int
3221   \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3222   \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3223   \dim_set:Nn \l__enumext_joined_width_viii_dim
3224   {
3225     \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3226     + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3227       + \l__enumext_columns_sep_viii_dim
3228       )*\l__enumext_joined_item_aux_viii_int
3229   }
3230   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3231 }
3232 {
3233   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3234   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3235 }
3236 }

```

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

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:

```

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

```

3237 \cs_new_protected:Nn \__enumext_start_mini_viii:
3238 {
3239   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3240   {
3241     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3242     {
3243       \linewidth
3244       - \l__enumext_minipage_right_viii_dim
3245       - \l__enumext_minipage_hsep_viii_dim
3246     }
3247     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3248     \dim_gset_eq:NN
3249     \g__enumext_minipage_right_viii_dim
3250     \l__enumext_minipage_right_viii_dim
3251     \__enumext_mini_addvspace_viii:
3252     \nointerlineskip\noindent
3253     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3254   }
3255 }
3256 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3257 {
3258   \bool_if:NT \l__enumext_minipage_active_viii_bool
3259   {
3260     \end{__enumext_mini_env*}
3261     \hfill
3262     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3263   }
3264 }
3265 \__enumext_after_env:nn {keyans*}
3266 {
3267   \bool_if:NT \g__enumext_minipage_active_viii_bool

```

```

3268     {
3269         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3270         \par\addvspace { \g__enumext_minipage_right_skip }
3271         \bool_if:NF \g__enumext_minipage_center_viii_bool
3272         {
3273             \centering
3274         }
3275         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3276         \end{__enumext_mini_env*}
3277         \par\addvspace{ \g__enumext_minipage_after_skip }
3278     }
3279     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3280     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3281     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3282     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3283 }

```

(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.

```

3284 \NewDocumentEnvironment{keyans*}{ o }
3285 {
3286     \__enumext_safe_exec_viii:
3287     \__enumext_parse_keys_viii:n {#1}
3288     \__enumext_before_list_viii:
3289     \__enumext_start_list:nn { }
3290     {
3291         \__enumext_list_arg_two_viii:
3292         \__enumext_before_keys_exec_viii:
3293     }
3294     \__enumext_starred_columns_set_viii:
3295     \item[] \scan_stop:
3296     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3297     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3298 }
3299 {
3300     \__enumext_stop_item_tmp_viii:
3301     \__enumext_remove_extra_parsep_viii:
3302     \__enumext_stop_list:
3303     \__enumext_after_list_viii:
3304 }

```

(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.

```

3305 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3306 {
3307     \int_incr:N \l__enumext_keyans_level_h_int
3308     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3309     {
3310         \msg_error:nn { enumext } { nested }
3311     }
3312     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3313     \bool_set_false:N \l__enumext_store_active_bool
3314     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3315     {
3316         \msg_error:nn { enumext } { keyans-wrong-level }
3317     }
3318 }

```

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

\\_\_enumext\_parse\_keys\_viii:n Parse [*key = val*] for **keyans\***.

```

3319 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3320 {
3321     \tl_if_novalue:nF {#1}
3322     {
3323         \keys_set:nn { enumext / keyans* } {#1}
3324     }
3325 }

```

(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`.

```
3326 \cs_new_protected:Nn \__enumext_before_list_viii:
3327 {
3328     \__enumext_vspace_above_viii:
3329     \__enumext_before_args_exec_viii:
3330     \__enumext_start_mini_viii:
3331 }
```

(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.

```
3332 \cs_new_protected:Nn \__enumext_after_list_viii:
3333 {
3334     \__enumext_stop_mini_viii:
3335     \__enumext_after_stop_list_viii:
3336     \__enumext_vspace_below_viii:
3337 }
```

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

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

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

`\__enumext_start_item_tmp_viii:` First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
3338 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3339 {
3340     \__enumext_stop_item_tmp_viii:
3341     \int_incr:N \__enumext_item_column_pos_viii_int
3342     \int_gincr:N \g__enumext_item_count_all_viii_int
3343     \__enumext_item_peek_args_viii:
3344 }
```

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

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

```
3345 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3346 {
3347     \peek_meaning:NTF (
3348         { \__enumext_joined_item_viii:w }
3349         { \__enumext_joined_item_viii:w (1) }
3350 }
```

(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`.

```
3351 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3352 {
3353     \__enumext_starred_joined_item_viii:n {#1}
3354     \peek_meaning_remove:NTF *
3355         { \__enumext_starred_item_viii:w }
3356         { \__enumext_standard_item_viii:w }
3357 }
```

(End of definition for `\__enumext_joined_item_viii:w`)

`\__enumext_standard_item_viii:w`

The function `\__enumext_standard_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[⟨custom⟩]` by means of the function `\__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]`.

```

3358 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3359 {
3360   \bool_set_false:N \l__enumext_item_starred_viii_bool
3361   \peek_meaning:NTF [
3362     {
3363       \bool_set_eq:NN
3364         \l__enumext_wrap_label_viii_bool
3365         \l__enumext_wrap_label_opt_viii_bool
3366       \__enumext_start_item_viii:w
3367     }
3368     {
3369       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3370       \legacy_if_set_true:n { @noitemarg }
3371       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3372     }
3373   }

```

(End of definition for `\__enumext_standard_item_viii:w`)

`\__enumext_starred_item_viii:w`

The function `\__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[⟨content⟩]`.

`\__enumext_starred_item_viii_aux_i:w`

`\__enumext_starred_item_viii_aux_ii:w`

```

3374 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3375 {
3376   \bool_set_true:N \l__enumext_item_starred_viii_bool
3377   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3378   \peek_meaning:NTF [
3379     { \__enumext_starred_item_viii_aux_i:w }
3380     { \__enumext_starred_item_viii_aux_ii:w }
3381   }
3382   \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3383   {
3384     \tl_clear:N \l__enumext_store_keyans_label_tl
3385     \tl_if_novalue:nF { #1 }
3386     {
3387       \tl_set:Nx \l__enumext_keyans_tmpa_tl { \c_space_tl [#1] }
3388       \tl_set:Nx \l__enumext_keyans_tmpb_tl { \c_space_tl #1 }
3389     }
3390     \__enumext_starred_item_viii_aux_ii:w
3391   }
3392   \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3393   {
3394     \legacy_if_set_true:n { @noitemarg }
3395     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3396   }

```

(End of definition for `\__enumext_starred_item_viii:w`, `\__enumext_starred_item_viii_aux_i:w`, and `\__enumext_starred_item_viii_aux_ii:w`)

### Pass content to prop list and more

`\__enumext_starred_item_exec_viii:`

The function `\@starred_item_exec_viii:` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[⟨content⟩]`.

```

3397 \cs_new_protected:Nn \__enumext_starred_item_exec:
3398 {
3399   \tl_put_left:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3400   \tl_if_blank:VF \l__enumext_keyans_tmpb_tl
3401   {
3402     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_keyans_tmpb_tl }
3403   }
3404   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3405   \__enumext_keyans_store_ref:

```

```

3406 \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3407 \__enumext_keyans_addto_seq_link:
3408 \bool_if:NT \l__enumext_show_answer_bool
3409 {
3410   \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3411   {
3412     \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3413     \__enumext_label_width_by_box:Nn \l_tmpa_dim { \tl_use:N \l__enumext_keyans_tmpa_tl }
3414     \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l_tmpa_dim }
3415   }
3416   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3417 }
3418 \bool_if:NT \l__enumext_show_position_bool
3419 {
3420   \tl_set:Ne \l__enumext_mark_answer_sym_tl
3421   {
3422     \group_begin:
3423     \exp_not:N \normalfont
3424     \exp_not:N \footnotesize [ \int_eval:n
3425       {
3426         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3427       }
3428     ]
3429     \group_end:
3430   }
3431   \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3432   {
3433     \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3434     \__enumext_label_width_by_box:Nn \l_tmpa_dim { \tl_use:N \l__enumext_keyans_tmpa_tl }
3435     \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l_tmpa_dim }
3436   }
3437   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3438 }
3439 }

```

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

### 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.

```

3440 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3441 {
3442   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3443   \legacy_if:nT { @noitemarg }
3444   {
3445     \legacy_if_set_false:n { @noitemarg }
3446     \legacy_if:nT { @nmbrrlist }
3447     {
3448       \bool_if:NT \l__enumext_hyperref_bool
3449       {
3450         \legacy_if_set_true:n { @hyper@item }
3451       }
3452       \refstepcounter{enumXviii}
3453     }
3454   }

```

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

```

3455 \group_begin:
3456 \lrbox{ \l__enumext_item_text_viii_box }
3457 \bool_if:NF \l__enumext_footnotes_key_bool
3458 {
3459   \__enumext_renew_footnote:
3460 }
3461 \bool_if:NT \l__enumext_item_starred_viii_bool
3462 {

```

```

3463         \__enumext_starred_item_exec:
3464     }
3465     \group_begin:
3466         \tl_use:N \l__enumext_label_font_style_viii_tl
3467         \bool_if:NTF \l__enumext_wrap_label_viii_bool
3468         {
3469             \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3470                 { \__enumext_wrapper_label_viii:n {#1} }
3471         }
3472         {
3473             \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3474         }
3475     \group_end:
3476     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3477     \tl_use:N \l__enumext_after_list_args_viii_tl
3478     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3479     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3480     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3481     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3482 }

```

(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.

```

3483 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3484 {
3485     \__enumext_endminipage:
3486     \endlrbox
3487     \group_end:
3488     \box_set_wd:Nn \l__enumext_item_text_viii_box
3489     {
3490         \l__enumext_joined_width_viii_dim
3491         + \l__enumext_labelwidth_viii_dim
3492         + \l__enumext_labelsep_viii_dim
3493     }
3494     \int_set:Nn \hbadness { 10000 }
3495     \box_use:N \l__enumext_item_text_viii_box
3496     \bool_if:NF \l__enumext_footnotes_key_bool
3497     {
3498         \__enumext_print_footnote:
3499     }
3500     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3501     {
3502         \par\noindent
3503         \int_zero:N \l__enumext_item_column_pos_viii_int
3504     }
3505     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3506 }

```

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

\\_\_enumext\_remove\_extra\_parsep\_viii: Finally we will remove the vertical space equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```

3507 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3508 {
3509     \int_compare:nNnT
3510     {
3511         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3512     }
3513     =
3514     { \c_zero_int }
3515     {
3516         \par
3517         \vspace{ -\l__enumext_itemsep_viii_skip }
3518         \int_gzero:N \g__enumext_item_count_all_viii_int
3519     }
3520 }

```

(End of definition for \\_\_enumext\_remove\_extra\_parsep\_viii:.)

### 10.37 The command \getkeyans

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

```
3521 \NewDocumentCommand \getkeyans { m }
3522 {
3523   \exp_args:Ne \__enumext_getkeyans_aux:n
3524   { \tl_to_str:e { \text_expand:n {#1} } }
3525 }
```

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

`\__enumext_getkeyans_aux:n` The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the  $\langle argument \rangle$  using “.”. If “.” is omitted it will return an error.

```
3526 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3527 {
3528   \str_if_in:nnTF {#1} { : }
3529   {
3530     \use:e
3531     {
3532       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3533       { {##1} {##2} }
3534     }
3535     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3536   }
3537   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3538 }
```

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

`\__enumext_getkeyans:nn` The internal function `\__enumext_getkeyans:nn` will check for the existence of the  $\langle prop\ list \rangle$ , if it does not exist it will return an error message, then it will fetch the content specified by the second  $\langle argument \rangle$  from  $\langle prop\ list \rangle$ .

```
3539 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3540 {
3541   \prop_if_exist:cF { g__enumext_#1_prop }
3542   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3543   \group_begin:
3544   \prop_item:cn { g__enumext_#1_prop }{#2}
3545   \group_end:
3546 }
```

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

### 10.38 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the  $\langle sequence \rangle$  defined by the `save-ans` key. The first thing we will do is to define a set of  $\langle keys \rangle$  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`.

```
3547 \keys_define:nn { keyanskey / print }
3548 {
3549   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3550   {
3551     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3552   },
3553   level-1 .initial:n = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3554   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3555   {
3556     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3557   },
3558   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3559   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3560   {
3561     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3562   },
3563   level-3 .initial:n = { nosep, label=\roman*. , first=\small, font=\small },
3564   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3565   {
3566     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3567   },
3568 }
```



```

3568     level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3569     level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3570                     {
3571                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3572                     },
3573     level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3574 }

```

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

```

3575 \NewDocumentCommand \printkeyans { s O{} m }
3576 {
3577     \group_begin:
3578     \tl_use:N \l__enumext_print_keyans_i_tl
3579     \tl_use:N \l__enumext_print_keyans_ii_tl
3580     \tl_use:N \l__enumext_print_keyans_iii_tl
3581     \tl_use:N \l__enumext_print_keyans_iv_tl
3582     \tl_use:N \l__enumext_print_keyans_vii_tl
3583     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3584     \group_end:
3585 }

```

(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

```

3586 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3587 {
3588     \seq_if_exist:cTF { g__enumext_#3_seq }
3589     {
3590         \seq_if_empty:cF { g__enumext_#3_seq }
3591         {
3592             %%\seq_show:c { g__enumext_#3_seq }
3593             \bool_if:nTF {#1}
3594             {
3595                 \begin{enumext*}[#2]
3596                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3597                 \end{enumext*}
3598             }
3599             {
3600                 \begin{enumext}[#2]
3601                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3602                 \end{enumext}
3603             }
3604         }
3605     }
3606     {
3607         \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3608     }
3609 }

```

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

## 10.39 The command `\setenumext`

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

```

3610 \keys_define:nn { enumext / meta-families }
3611 {
3612     level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3613     level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
3614     level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
3615     level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
3616     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },
3617     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} },
3618     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
3619     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } },
3620     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } },
3621     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } },

```

```

3622   print-4   .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3623   print-*   .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3624   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3625 }

```

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

```

3626 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3627 {
3628   level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3629   keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3630 }

```

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

```

3631 \NewDocumentCommand \setenumext { o +m }
3632 {
3633   \tl_if_novalue:nTF {#1}
3634   {
3635     \seq_map_inline:Nn \c__enumext_all_families_seq
3636   }
3637   {
3638     \seq_clear:N \l__enumext_setkey_tmpa_seq
3639     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3640     \int_set:Nn \l__enumext_setkey_tmpa_int
3641     {
3642       \seq_count:N \l__enumext_setkey_tmpb_seq
3643     }
3644     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3645     {
3646       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3647       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3648       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3649       {
3650         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3651       }
3652     }
3653     {
3654       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3655     }
3656     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3657     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3658     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3659   }
3660   {
3661     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3662   }
3663 }

```

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

`\__enumext_set_parse:n`  
`\__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```

3664 \cs_new_protected:Npn \__enumext_set_parse:n #1
3665 {
3666   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3667   \int_step_inline:nnn { 0 } { 4 } % <- max level
3668   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3669   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3670   {
3671     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3672     { \tl_trim_spaces:n {#1} }
3673   }
3674   { \__enumext_set_error:nn {#1} { } }
3675 }
3676 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3677 { \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.

```

3678 \msg_new:nnn { enumext } { package-load }
3679 {
3680   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3681 }

3682 \msg_new:nnn { enumext } { package-not-load }
3683 {
3684   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3685 }

3686 \msg_new:nnn { enumext } { package-load-foot }
3687 {
3688   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3689 }
```

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

```

3690 \msg_new:nnn { enumext } { counters }
3691 {
3692   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3693   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3694 }
```

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

```

3695 \msg_new:nnn { enumext } { invalid-key }
3696 {
3697   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3698 }
3699 \msg_new:nnn { enumext } { unknown-key-family }
3700 {
3701   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3702 }
```

Messages used in length calculation.

```

3703 \msg_new:nnn { enumext } { width-negative }
3704 {
3705   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3706   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
3707 }
3708 \msg_new:nnn { enumext } { width-zero }
3709 {
3710   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3711   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
3712 }
```

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

```

3713 \msg_new:nnn { enumext } { list-lengths }
3714 {
3715   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3716   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3717   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3718   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3719   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3720   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3721   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3722   \__enumext_show_length:nnn { skip } { topsep } {#1}
3723   \__enumext_show_length:nnn { skip } { parsep } {#1}
3724   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3725   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3726   ****
3727 }
```

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

```

3728 \msg_new:nnn { enumext } { list-lengths-not-nested }
3729 {
3730   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3731   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3732   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3733   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3734   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3735   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3736   \__enumext_show_length:nnn { dim } { listparindent } {#1}
```

```

3737     \__enumext_show_length:nnn { skip } { topsep } {#1}
3738     \__enumext_show_length:nnn { skip } { parsep } {#1}
3739     \__enumext_show_length:nnn { skip } { partopsep } {#1}
3740     \__enumext_show_length:nnn { skip } { itemsep } {#1}
3741     *****
3742 }

```

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

```

3743 \msg_new:nnn { enumext } { items-same-answer }
3744 {
3745     *****~Checking~answers~on~'#1'~OK~*****\\
3746     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3747     *****
3748     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3749 }
3750 \msg_new:nnn { enumext } { item-different-answer }
3751 {
3752     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3753     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3754 }

```

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

```

3755 \msg_new:nnn { enumext } { missing-starred }
3756 {
3757     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3758 }

```

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

```

3759 \msg_new:nnn { enumext } { list-too-deep }
3760 {
3761     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
3762     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3763 }

```

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

```

3764 \msg_new:nnn { enumext } { anskey-wrong-place }
3765 {
3766     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3767     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3768 }
3769 \msg_new:nnn { enumext } { anspic-wrong-place }
3770 {
3771     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3772     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3773 }
3774 \msg_new:nnn { enumext } { command-wrong-place }
3775 {
3776     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3777     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3778 }

```

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

```

3779 \msg_new:nnn { enumext } { keyans-nested }
3780 {
3781     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3782 }
3783 \msg_new:nnn { enumext } { keyans-wrong-level }
3784 {
3785     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3786     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3787 }
3788 \msg_new:nnn { enumext } { wrong-place }
3789 {
3790     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3791     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3792 }
3793 \msg_new:nnn { enumext } { keyanspic-nested }
3794 {
3795     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
3796 }
3797 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3798 {
3799     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\

```

```

3800     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3801 }

```

Messages used by `\getkeyans` command.

```

3802 \msg_new:nnn { enumext } { undefined-storage-anskey }
3803 {
3804     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3805 }

```

Messages used by `\miniright` command.

```

3806 \msg_new:nnn { enumext } { missing-miniright }
3807 {
3808     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
3809     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3810 }
3811 \msg_new:nnn { enumext } { wrong-miniright-place }
3812 {
3813     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3814     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3815 }
3816 \msg_new:nnn { enumext } { wrong-miniright-use }
3817 {
3818     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3819     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3820 }

```

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

```

3821 \msg_new:nnn { enumext } { nested }
3822 {
3823     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3824 }
3825 \msg_new:nnn { enumext } { item-joined }
3826 {
3827     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3828 }
3829 \msg_new:nnn { enumext } { item-joined-columns }
3830 {
3831     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3832 }

```

## 10.41 Finish package

Finish package implementation.

```

3833 \file_input_stop:
3834 \endpackage

```

# 11 Index of Implementation

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

Symbols	
\*	401
\+	202
\-	202
\\	210, 2651, 3692, 3705, 3710, 3715, 3730, 3745, 3746, 3761, 3766, 3771, 3776, 3785, 3790, 3799, 3808, 3813, 3818
commands:	
\@starred_item_exec_viii:	93
A	
above	1212
above*	1212
\addvspace	859, 887, 1010, 1089, 1152, 1158, 1186, 1203, 2468, 2490, 2607, 2622, 2846, 2853, 3270, 3277
after	697
align	355
\Alph	29, 33, 34
\Alph	307, 484, 502, 515, 3568
\alph	29, 33, 34
\alph	308, 482, 3558
\anskey	10, 57, 1646
\anspic	12, 78, 2629
\arabic	29, 31
\arabic	306, 481, 501, 3553, 3573
B	
\b	2357, 2370, 2915, 2928
\baselineskip	41
\baselineskip	1606, 1614
before	697
before*	697
below	1212
below*	1212
bool commands:	
\bool_gset_false:N	2513, 2855, 2963, 3166, 3279
\bool_gset_true:N	801, 2341, 2475, 2838, 2856, 2897, 2960, 3262, 3280
\bool_if:NTF	247, 259, 276, 1234, 1248, 1261, 1272, 1283, 1294, 1305, 1316, 1355, 1362, 1373, 1434, 1436, 1568, 1592, 1599, 1627, 1658, 1671, 1673, 1684, 1704, 1829, 1840, 1844, 1878, 1893, 1962, 1977, 1988, 2064, 2095, 2169, 2185, 2253, 2263, 2299, 2304, 2348, 2355, 2368, 2402, 2452, 2466, 2481, 2506, 2536, 2592, 2605, 2613, 2631, 2834, 2843, 2847, 2905, 2913, 2926, 2968, 2978, 3061, 3067, 3070, 3084, 3088, 3103, 3132, 3159, 3258, 3267, 3271, 3408, 3418, 3448, 3457, 3461, 3467, 3496
\bool_if:nTF	1187, 1204, 1712, 2107, 2141, 2205, 2652, 3593
\bool_if_p:N	2242, 2289
\bool_lazy_all:nTF	1414, 1761, 1770, 1783, 1799, 2335, 2891
\bool_lazy_and:nnTF	1694, 1737, 1947, 2240, 2288, 2384, 2471, 2958
\bool_lazy_or:nnTF	2657
\bool_new:N	25, 26, 27, 28, 29, 35, 37, 46, 67, 72, 73, 78, 79, 82, 99, 101, 103, 106, 107, 116, 117, 118, 119, 130, 131, 156, 167, 169
\bool_not_p:n	1696, 1788, 1803, 2337, 2386, 2473, 2893
\bool_set_eq:NN	2073, 2122, 3011, 3363
\bool_set_false:N	256, 1389, 1390, 1495, 1498, 1518, 1521, 2495, 2543, 2625, 2689, 2706, 2964, 3008, 3313, 3360
\bool_set_true:N	238, 242, 348, 625, 1218, 1223, 1350, 1369, 1380, 1494, 1497, 1517, 1520, 1530, 1537, 2069, 2100, 2118, 2130, 2334, 2390, 2395, 2421, 2541, 2567, 2823, 2890, 3017, 3024, 3025, 3247, 3369, 3376, 3377
box commands:	
\box_dp:N	906, 910, 914, 925, 929, 940, 949, 955, 965, 978, 984, 990, 1021, 1022, 1023, 1026, 1036, 1040, 1049, 1056, 1061, 1069, 1098, 1099, 1102, 1109, 1122, 1130, 1136, 1144, 2718
\box_new:N	43, 162
\box_set_wd:Nn	3124, 3488
\box_use:N	3131, 3495
\box_wd:N	314
C	
\c	401, 402, 525, 527, 539, 541
\cB	402
\cE	402
\centering	1189, 1206, 2739, 2849, 3273
check-ans	1382
Document class:	
article	35
clist commands:	
\clist_const:Nn	174
\clist_map_function:nN	2726
\clist_map_inline:Nn	354, 567, 630, 696, 711, 792, 1228
\clist_map_inline:nn	34, 51, 57, 69, 81, 105, 129, 139, 153, 173, 218, 379, 396, 635, 807, 1395, 1507, 1525, 1546, 1758, 1887, 2022, 2234, 2237, 2270, 2280, 2283, 2309
\columnbreak	58
\columnbreak	1698
columns	776
columns*	1526
columns-sep	776
columns-sep*	1526
\columnsep	73, 77
\columnsep	2446, 2589
\columnseprule	73, 77
\columnseprule	2450, 2591
Commands provide by enumext:	
\anskey	23, 24, 50-52, 55, 57, 59-61, 63, 72, 85, 96, 97, 100
\anspic*	23, 61-63, 78-80, 96, 97
\anspic	55, 78-80, 100
\getkeyans	55, 96, 101
\item*	23, 55, 61-63, 65, 66, 86, 93, 96, 97
\itemwidth	81, 89
\item	65, 66, 81, 85-87, 89, 92-94
\miniright	23, 39, 46, 47, 73, 74, 76, 77, 101
\printkeyans	24, 55, 96
\setenumext	23, 97-99
Counters defined by enumext:	
enumXiii	22, 29
enumXii	22, 29
enumXiv	22, 29
enumXi	22, 29

enumXviii	22, 29, 94
enumXvii	22, 29, 87
enumXvi	22, 29
enumXv	22, 29
cs commands:	
\cs_generate_variant:Nn	316, 332, 531, 547, 1551, 1560, 1565, 1645, 2224, 2728
\cs_if_exist:NTF	286
\cs_new:Nn	187
\cs_new:Npn	191, 196, 206
\cs_new_eq:NN	222, 223, 224, 228, 229, 261, 262, 265, 266
\cs_new_protected:Nn	233, 397, 417, 449, 712, 716, 720, 724, 728, 732, 736, 740, 744, 748, 752, 756, 760, 764, 768, 772, 808, 820, 844, 861, 872, 896, 971, 995, 1012, 1074, 1091, 1113, 1148, 1154, 1229, 1243, 1257, 1268, 1279, 1290, 1301, 1312, 1360, 1371, 1432, 1444, 1461, 1566, 1590, 1597, 1625, 1632, 1749, 1876, 1891, 1919, 1945, 2027, 2031, 2050, 2103, 2137, 2153, 2163, 2179, 2329, 2382, 2400, 2407, 2430, 2460, 2479, 2534, 2557, 2575, 2600, 2611, 2648, 2691, 2704, 2724, 2729, 2745, 2813, 2832, 2883, 2940, 2946, 2966, 2976, 2993, 3143, 3169, 3237, 3256, 3305, 3326, 3332, 3345, 3397, 3507
\cs_new_protected:Npn	179, 183, 269, 284, 301, 311, 317, 405, 424, 518, 532, 1176, 1195, 1339, 1400, 1552, 1561, 1681, 1826, 1838, 1860, 1929, 1967, 1975, 2060, 2079, 2114, 2126, 2193, 2227, 2273, 2344, 2353, 2553, 2699, 2764, 2900, 2911, 2999, 3006, 3022, 3030, 3035, 3047, 3188, 3319, 3351, 3358, 3374, 3382, 3392, 3526, 3539, 3586, 3664, 3676
\cs_new_protected_nopar:Nn	2986, 3119, 3338, 3483
\cs_new_protected_nopar:Npn	3053, 3440
\cs_set:Nn	194, 1831
\cs_set:Npn	1759, 1797, 3532
\cs_set_eq:NN	193, 198, 2873, 2874, 3055, 3296, 3297, 3442
\cs_set_protected:Nn	212, 636, 652, 664, 676
\cs_set_protected:Npn	30, 44, 52, 64, 70, 95, 124, 135, 147, 154, 214, 333, 355, 384, 465, 485, 548, 568, 612, 631, 688, 697, 776, 793, 1212, 1382, 1479, 1508, 1526, 1751, 1880, 2011, 2225, 2271
\cs_to_str:N	303, 326

D

\d	202
\DeclareDocumentEnvironment	889
dim commands:	
\dim_abs:n	2198, 2203
\dim_add:Nn	2709, 3414, 3435
\dim_compare:nNnTF	638, 654, 666, 678, 1178, 1197, 2195, 2200, 2206, 2212, 2214, 2216, 2412, 2435, 2561, 2579, 2701, 2747, 2815, 3171, 3239
\dim_compare:nTF	1722
\dim_gset_eq:NN	2824, 3248
\dim_gzero:N	2858, 3282
\dim_new:N	40, 47, 48, 49, 66, 102, 112, 163, 164, 170
\dim_set:Nn	314, 626, 1538, 2093, 2198, 2203, 2205, 2208, 2209, 2213, 2215, 2218, 2219, 2221, 2415, 2438, 2563, 2581, 2731, 2749, 2756, 2799, 2817, 3049, 3173, 3180, 3223, 3241
\dim_set_eq:NN	472, 492, 508, 512, 2088, 2236, 2282, 2372, 2446, 2589, 2806, 2809, 2810, 2930, 3040, 3230, 3233, 3234

\dim_use:N	639, 647, 1179, 1185, 1635, 1638, 1643, 2158, 2160, 2413, 2418, 2419, 2426, 2436, 2440, 2441, 2443
\dim_zero:N	2450, 2591, 2710, 2711, 2712
\dim_zero_new:N	2762, 3186
\l_tmpa_dim	3413, 3414, 3434, 3435
\c_zero_dim	641, 655, 667, 679, 1179, 1197, 1724, 2195, 2200, 2206, 2213, 2413, 2436, 2561, 2579, 2747, 2815, 3171, 3239

E

\end	1182, 1200, 1594, 1629, 2465, 2489, 2604, 2621, 2836, 2852, 3260, 3276, 3597, 3602
\endlist	27
\endlist	223
\endlrbox	3122, 3486
\endminipage	27
\endminipage	229
enumext	4, 2310
enumext internal commands:	
\__enumext_add_pre_parsep:	40, 818, 820, 820
\__enumext_after_args_exec:	37, 712, 724, 2322
\__enumext_after_args_exec_v:	38, 728, 740, 2527
\__enumext_after_args_exec_vii:	744, 768
\__enumext_after_args_exec_viii:	772
\__enumext_after_env:n	74
\__enumext_after_env:nn	75, 89, 183, 183, 2504, 2841, 3157, 3265
\__enumext_after_hyperref:	27, 231, 233, 233
\__enumext_after_list:	74, 85, 92, 2327, 2479, 2479
\l__enumext_after_list_args_v_tl	742
\l__enumext_after_list_args_vii_tl	770, 3113
\l__enumext_after_list_args_viii_tl	774, 3477
\__enumext_after_list_v:	77, 2532, 2611, 2611
\__enumext_after_list_vii:	2881, 2946, 2946
\__enumext_after_list_viii:	3303, 3332, 3332
\__enumext_after_star_env:nn	82
\__enumext_after_stop_list:	37, 38, 712, 720, 2493
\__enumext_after_stop_list_v:	38, 728, 736, 2626
\l__enumext_after_stop_list_v_tl	738
\__enumext_after_stop_list_vii:	744, 760, 2949
\l__enumext_after_stop_list_vii_tl	762
\__enumext_after_stop_list_viii:	764, 3335
\l__enumext_after_stop_list_viii_tl	766
\l__enumext_align_label_vii_str	3105, 3109
\l__enumext_align_label_viii_str	3469, 3473
\l__enumext_align_label_X_str	154
\c_enumext_all_envs_clist	174, 354, 567, 630, 696, 711, 792, 1228
\c_enumext_all_families_seq	98, 3626, 3635, 3657
\__enumext_anskey_wrapper:n	1483, 1836
\__enumext_at_begin_document:n	27, 179, 179, 220, 226
\__enumext_before_args_exec:	37, 712, 712, 2410
\__enumext_before_args_exec_v:	37, 38, 728, 728, 2560
\__enumext_before_args_exec_vii:	744, 744, 2943
\__enumext_before_args_exec_viii:	748, 3329
\__enumext_before_keys_exec:	37, 712, 716, 2320
\__enumext_before_keys_exec_v:	38, 728, 732, 2525
\__enumext_before_keys_exec_vii	744



`\__enumext_before_keys_exec_vii:` 38, 752, 2869  
`\__enumext_before_keys_exec_viii:` .. 38, 756, 3292  
`\__enumext_before_list:` ... 73, 2314, 2407, 2407  
`\__enumext_before_list_v:` . 76, 2520, 2557, 2557  
`\__enumext_before_list_vii:` 84, 2864, 2940, 2940  
`\__enumext_before_list_viii:` .. 92, 3288, 3326, 3326  
`\l__enumext_before_no_starred_key_v_tl` 734  
`\l__enumext_before_no_starred_key_vii_-tl` ..... 754  
`\l__enumext_before_no_starred_key_viii_-tl` ..... 758  
`\l__enumext_before_starred_key_v_tl` ... 730  
`\l__enumext_before_starred_key_vii_tl` . 746  
`\l__enumext_before_starred_key_viii_tl` 750  
`\__enumext_calc_hspace:NNNNNN` 68, 2193, 2193, 2224, 2229, 2275  
`\__enumext_check_ans_active:` .. 53, 1444, 1444, 2510  
`\__enumext_check_ans_active_vii:` . 1444, 1461, 3163  
`\l__enumext_check_ans_bool` ... 50, 65, 116, 1355, 1386, 1390, 1434, 1673, 1962, 2064, 2095, 2472, 2958, 3067  
`\__enumext_check_ans_count:` . 53, 73, 1432, 1432, 2411  
`\__enumext_check_ans_int:n` .. 50, 52, 1357, 1400, 1400  
`\g__enumext_check_ans_item_tl` .. 63, 116, 1961, 1969, 1973  
`\g__enumext_check_ans_show_bool` 74, 116, 2475, 2506, 2513  
`\g__enumext_check_ans_show_h_bool` 116, 2960, 3159, 3166  
`\l__enumext_columns_sep_v_dim` 2579, 2581, 2589  
`\l__enumext_columns_sep_vii_dim` .. 2747, 2749, 2758, 2803, 2932, 3141  
`\l__enumext_columns_sep_viii_dim` . 3171, 3173, 3182, 3227, 3505  
`\l__enumext_columns_v_int` 1017, 2577, 2585, 2597, 2602  
`\l__enumext_columns_vii_int` .. 2752, 2755, 2759, 2767, 2771, 2774, 2780, 2786, 2790, 2919, 3136, 3147  
`\l__enumext_columns_viii_int` . 3176, 3179, 3183, 3191, 3195, 3198, 3204, 3210, 3214, 3500, 3511  
`\l__enumext_compare_items_ans_int` 116, 1446, 1452, 1463, 1470  
`\g__enumext_count_item_all_int` 116, 1420, 1423, 1448, 1465, 2066, 2097, 3076  
`\g__enumext_count_item_i_int` ..... 1425, 1465  
`\g__enumext_count_item_ii_int` 1426, 1448, 1466  
`\g__enumext_count_item_iii_int` 1427, 1449, 1466  
`\g__enumext_count_item_iv_int` 1428, 1449, 1467  
`\g__enumext_count_item_vii_int` ..... 1429  
`\g__enumext_count_item_with_ans_int` 52, 57, 63, 116, 1430, 1452, 1470, 1675, 1964  
`\g__enumext_count_item_X_int` ..... 116  
`\g__enumext_count_level_vii_int` ..... 3077  
`\g__enumext_count_level_X_int` ..... 116  
`\l__enumext_counter_i_tl` ..... 30, 293  
`\l__enumext_counter_ii_tl` ..... 30, 294  
`\l__enumext_counter_iii_tl` ..... 30, 295  
`\l__enumext_counter_iv_tl` ..... 30, 296  
`\l__enumext_counter_style_for_ref_vii_-tl` ..... 432, 442, 453, 455  
`\l__enumext_counter_style_for_ref_viii_-tl` ..... 459, 461  
`\l__enumext_counter_style_for_ref_X_tl` 143  
`\c__enumext_counter_style_tl` ... 31, 143, 399  
`\g__enumext_counter_styles_tl` . 22, 29, 40, 304, 322  
`\l__enumext_counter_v_tl` ..... 30, 297  
`\l__enumext_counter_vi_tl` ..... 30, 298  
`\l__enumext_counter_vii_tl` ..... 30, 299, 429  
`\l__enumext_counter_viii_tl` ..... 30, 300, 439  
`\l__enumext_current_widest_dim` 22, 40, 328, 473, 493, 509, 513  
`\__enumext_default_item:n` ... 2060, 2060, 2111  
`\__enumext_define_counters:Nn` 22, 284, 284, 293, 294, 295, 296, 297, 298, 299, 300  
`\__enumext_endminipage:` . 27, 226, 229, 895, 2741, 3121, 3485  
`\__enumext_fake_item:` ..... 636, 636, 2262  
`\l__enumext_fake_item_indent_v_dim` 655, 660  
`\l__enumext_fake_item_indent_v_tl` 657, 2119, 2123, 2131  
`\l__enumext_fake_item_indent_vii_dim` 667, 672  
`\l__enumext_fake_item_indent_vii_tl` 669, 3117  
`\l__enumext_fake_item_indent_viii_dim` . 679, 684, 3414, 3435  
`\l__enumext_fake_item_indent_viii_tl` .. 681, 3481  
`\l__enumext_fake_item_indent_X_tl` ..... 70  
`\__enumext_fake_item_vii:` .... 636, 664, 2298  
`\__enumext_fake_item_viii:` .... 636, 676, 2303  
`\g__enumext_footnote_arg_seq` . 140, 2033, 2046, 2056  
`\g__enumext_footnote_int` . 140, 2040, 2043, 2045, 2047  
`\g__enumext_footnote_int_seq` . 140, 2034, 2047, 2052, 2055  
`\__enumext_footnotes_key_bool` ..... 27  
`\l__enumext_footnotes_key_bool` 24, 27, 87, 130, 242, 247, 256, 3084, 3132, 3457, 3496  
`\__enumext_footnotetext:nn` ... 2027, 2027, 2057  
`\__enumext_getkeyans:nn` ... 96, 3535, 3539, 3539  
`\__enumext_getkeyans_aux:n` . 96, 3523, 3526, 3526  
`\l__enumext_hyperref_bool` . 24, 27, 28, 130, 238, 259, 276, 1739, 1949, 3061, 3448  
`\__enumext_hypertarget:nn` 28, 233, 261, 265, 281  
`\__enumext_if_is_int:n` ..... 200  
`\__enumext_if_is_int:nTF` ..... 200, 520, 534  
`\l__enumext_item_column_pos_vii_int` 85, 2774, 2780, 2786, 2790, 2797, 2989, 3136, 3139  
`\l__enumext_item_column_pos_viii_int` ... 92, 3198, 3204, 3210, 3214, 3221, 3341, 3500, 3503  
`\l__enumext_item_column_pos_X_int` ..... 154  
`\g__enumext_item_count_all_vii_int` 85, 2798, 2990, 3147, 3154  
`\g__enumext_item_count_all_viii_int` 92, 3222, 3342, 3511, 3518  
`\g__enumext_item_count_all_X_int` ..... 154  
`\__enumext_item_peek_args_vii:` .. 85, 86, 2991, 2993, 2993  
`\__enumext_item_peek_args_viii:` 92, 3343, 3345, 3345  
`\__enumext_item_starred:` .. 67, 2153, 2153, 2171



`\l__enumext_item_starred_vii_bool` 3008, 3024, 3088  
`\l__enumext_item_starred_viii_bool` 3360, 3376, 3461  
`\l__enumext_item_starred_X_bool` ..... 154  
`\__enumext_item_std:w` 27, 65, 67, 80, 220, 224, 2070, 2076, 2101, 2119, 2123, 2131, 2722  
`\g__enumext_item_symbol_aux_vii_tl` 3032, 3090, 3093, 3097, 3099  
`\g__enumext_item_symbol_aux_X_tl` ..... 154  
`\l__enumext_item_symbol_sep_vii_dim` .. 3041, 3049, 3096, 3098  
`\g__enumext_item_symbol_tl` 22, 65, 35, 2085, 2159, 2176  
`\l__enumext_item_symbol_vii_tl` ..... 3093  
`\l__enumext_item_text_vii_box` 3083, 3124, 3131  
`\l__enumext_item_text_viii_box` 3456, 3488, 3495  
`\l__enumext_item_text_X_box` ..... 154  
`\l__enumext_item_width_vii_dim` ... 2756, 2801, 2809, 2810  
`\l__enumext_item_width_viii_dim` .. 3180, 3225, 3233, 3234  
`\l__enumext_item_width_X_dim` ..... 154  
`\l__enumext_itemindent_X_dim` ..... 44  
`\l__enumext_itemsep_vii_skip` ..... 3153  
`\l__enumext_itemsep_viii_skip` ..... 3517  
`\l__enumext_joined_item_aux_vii_int` .. 2795, 2796, 2797, 2798, 2804  
`\l__enumext_joined_item_aux_viii_int` . 3219, 3220, 3221, 3222, 3228  
`\l__enumext_joined_item_aux_X_int` .... 154  
`\__enumext_joined_item_vii:w` .. 86, 2996, 2997, 2999, 2999  
`\l__enumext_joined_item_vii_int` .. 2766, 2767, 2770, 2772, 2778, 2783, 2788, 2793, 2795, 2801  
`\__enumext_joined_item_viii:w` . 92, 3348, 3349, 3351, 3351  
`\l__enumext_joined_item_viii_int` . 3190, 3191, 3194, 3196, 3202, 3207, 3212, 3217, 3219, 3225  
`\l__enumext_joined_item_X_int` ..... 154  
`\l__enumext_joined_width_vii_dim` . 2799, 2806, 2809, 3114, 3126  
`\l__enumext_joined_width_viii_dim` 3223, 3230, 3233, 3478, 3490  
`\l__enumext_joined_width_X_dim` ..... 154  
`\__enumext_keyans_addto_prop:n` 61, 1860, 1860, 2133, 2654  
`\__enumext_keyans_addto_seq:n` . 62, 1929, 1929, 2135, 2656  
`\__enumext_keyans_addto_seq_link:` 1929, 1943, 1945, 3407  
`\__enumext_keyans_anspic_code:nnn` . 78, 2645, 2648, 2648  
`\__enumext_keyans_check_ans:nn` 63, 1967, 1967, 2530, 2686  
`\__enumext_keyans_default_item:n` .. 66, 2114, 2114, 2149  
`\l__enumext_keyans_env_bool` 20, 2386, 2541, 2625  
`\__enumext_keyans_fake_item:` .. 636, 652, 2252  
`\l__enumext_keyans_level_h_int` 20, 1907, 3307, 3308  
`\l__enumext_keyans_level_int` .. 20, 1170, 1662, 1902, 2540, 2544, 2639  
`\__enumext_keyans_make_label:` 30, 68, 2179, 2179, 2251  
`\__enumext_keyans_mini_addvspace:` 45, 76, 1074, 1074, 2569  
`\__enumext_keyans_mini_right_cmd:n` 47, 1172, 1195, 1195  
`\__enumext_keyans_mini_set_vskip:` . 44, 1012, 1012, 1076  
`\__enumext_keyans_multi_addvspace:` . 77, 861, 872, 2594  
`\__enumext_keyans_multi_set_vskip:` . 41, 861, 861, 874  
`\__enumext_keyans_multicols_start:` 76, 2573, 2575, 2575  
`\__enumext_keyans_multicols_stop:` . 77, 1199, 2600, 2600, 2624  
`\__enumext_keyans_parse_keys:n` 2519, 2553, 2553  
`\l__enumext_keyans_pic_above_int` . 111, 2732, 2733, 2735  
`\l__enumext_keyans_pic_above_skip` .. 80, 111, 2677, 2716  
`\__enumext_keyans_pic_arg_two:` 80, 2675, 2704, 2704  
`\l__enumext_keyans_pic_below_int` . 111, 2732, 2733, 2736  
`\l__enumext_keyans_pic_body_seq` .. 78–80, 111, 2643, 2682, 2740  
`\__enumext_keyans_pic_do:n` 80, 2682, 2684, 2724, 2724, 2728  
`\l__enumext_keyans_pic_level_int` .. 20, 1162, 1666, 1863, 1897, 1932, 2693, 2694  
`\__enumext_keyans_pic_row:n` 80, 2726, 2729, 2729  
`\__enumext_keyans_pic_safe_exec:` .. 79, 2671, 2691, 2691  
`\__enumext_keyans_pic_skip_abs:N` .. 80, 2699, 2699, 2715  
`\l__enumext_keyans_pic_width_dim` . 111, 2731, 2738  
`\__enumext_keyans_redefine_item:` .. 67, 2137, 2137, 2250  
`\__enumext_keyans_safe_exec:` . 2518, 2534, 2534  
`\__enumext_keyans_show_left:n` . 67, 1975, 1975, 2129, 2662  
`\__enumext_keyans_starred_item:n` .. 67, 2126, 2126, 2145  
`\__enumext_keyans_store_ref:` .. 61, 1876, 1876, 2134, 2655, 3405  
`\__enumext_keyans_store_ref_aux_i:` 62, 1876, 1888, 1891  
`\__enumext_keyans_store_ref_aux_ii:` 62, 1876, 1917, 1919  
`\l__enumext_keyans_tmpa_tl` .. 23, 82, 2128, 2132, 3387, 3410, 3412, 3413, 3431, 3433, 3434  
`\l__enumext_keyans_tmppb_tl` 82, 3388, 3400, 3402  
`\l__enumext_label_copy_i_tl` .. 1793, 1895, 1900, 1905, 1910  
`\l__enumext_label_copy_v_tl` ..... 1905  
`\l__enumext_label_copy_vi_tl` ..... 1900  
`\l__enumext_label_copy_vii_tl` 1768, 1779, 1810, 1895  
`\l__enumext_label_copy_viii_tl` ..... 1910  
`\l__enumext_label_copy_X_tl` ..... 132  
`\l__enumext_label_fill_left_v_tl` ..... 2183  
`\l__enumext_label_fill_left_X_tl` ..... 70  
`\l__enumext_label_fill_right_v_tl` .... 2190  
`\l__enumext_label_fill_right_X_tl` ..... 70

```

\l__enumext_label_font_style_v_tl 2184, 2666
\l__enumext_label_font_style_vii_tl ... 3102
\l__enumext_label_font_style_viii_tl .. 3466
\l__enumext_label_i_tl ..... 465
\l__enumext_label_ii_tl ..... 465
\l__enumext_label_iii_tl ..... 465
\l__enumext_label_iv_tl ..... 465
\__enumext_label_style:Nnn 22, 29, 317, 317, 332,
470, 490, 506, 510
\l__enumext_label_v_tl .. 61, 62, 503, 1868, 1937,
1979, 1986, 2001, 2008, 2128, 2132, 2522, 2661, 2663
\l__enumext_label_vi_tl . 61, 62, 503, 1865, 1934,
2661, 2663, 2667
\l__enumext_label_vii_tl . 485, 3019, 3044, 3051
\l__enumext_label_viii_tl 485, 3371, 3395, 3399,
3412, 3433
\l__enumext_label_width_by_box .. 40, 313, 314
\__enumext_label_width_by_box:Nn 29, 311, 311,
316, 328, 544, 3413, 3434
\l__enumext_labelsep_i_dim ... 1983, 2005, 3416,
3437
\l__enumext_labelsep_v_dim ..... 2584
\l__enumext_labelsep_vii_dim . 2751, 2760, 2802,
3042, 3112, 3128
\l__enumext_labelsep_viii_dim 3175, 3184, 3226,
3476, 3492
\l__enumext_labelwidth_i_dim . 1982, 2004, 3416,
3437
\l__enumext_labelwidth_v_dim ..... 2584
\l__enumext_labelwidth_vii_dim ... 2751, 2759,
2802, 3105, 3109, 3127
\l__enumext_labelwidth_viii_dim .. 3175, 3183,
3226, 3469, 3473, 3491
\l__enumext_leftmargin_tmp_v_bool . 80, 2706
\l__enumext_leftmargin_tmp_X_bool ..... 44
\l__enumext_leftmargin_tmp_X_dim ..... 44
\l__enumext_leftmargin_X_dim ..... 44
\__enumext_level: 187, 187, 193, 194, 198, 408, 410,
411, 419, 421, 639, 643, 647, 714, 718, 722, 726, 810,
812, 814, 816, 849, 851, 853, 855, 859, 899, 902, 921,
930, 936, 941, 945, 956, 960, 961, 966, 1002, 1006,
1179, 1185, 1232, 1234, 1236, 1239, 1246, 1248, 1250,
1253, 1438, 1439, 1441, 1570, 1578, 1582, 1586, 1831,
1834, 1835, 2067, 2069, 2070, 2074, 2075, 2076, 2083,
2085, 2089, 2090, 2093, 2098, 2100, 2101, 2155, 2158,
2160, 2167, 2168, 2169, 2172, 2175, 2317, 2319, 2355,
2360, 2361, 2362, 2364, 2368, 2373, 2374, 2375, 2377,
2390, 2395, 2402, 2413, 2415, 2418, 2419, 2421, 2426,
2433, 2436, 2438, 2440, 2441, 2442, 2443, 2446, 2452,
2457, 2463, 2466, 2468, 2481, 3072, 3073
\__enumext_level_ ..... 193
\__enumext_level_end:n ..... 191, 196
\l__enumext_level_h_int 20, 427, 451, 1787, 1804,
2338, 2388, 2885, 2886
\l__enumext_level_int 20, 189, 822, 973, 1166, 1417,
1764, 1774, 1780, 1786, 1794, 1802, 1809, 2265, 2331,
2332, 2347, 2393, 2448, 2508, 2548, 2635, 2894, 2970,
2980, 3161, 3314
\__enumext_level_set:n ..... 191, 191
\__enumext_list_arg_two_i: ..... 2225
\__enumext_list_arg_two_ii: ..... 2225
\__enumext_list_arg_two_iii: ..... 2225
\__enumext_list_arg_two_iv: ..... 2225
\__enumext_list_arg_two_v: . 67, 2225, 2524, 2707
\__enumext_list_arg_two_vii: ..... 2271, 2868
\__enumext_list_arg_two_viii: .... 2271, 3291
\l__enumext_listoffset_v_dim ..... 2586
\l__enumext_listparindent_vii_dim .... 3115
\l__enumext_listparindent_viii_dim ... 3479
\__enumext_make_label: 30, 65, 66, 68, 2163, 2163,
2260
\l__enumext_mark_answer_sym_tl . 56, 106, 1486,
1640, 1846, 1990, 3420
\l__enumext_mark_position_str 106, 1490, 1491,
1513, 1514, 1638
\l__enumext_mark_ref_sym_tl .. 106, 1500, 1744,
1957
\__enumext_mini_addvspace: 43, 73, 995, 995, 2423
\__enumext_mini_addvspace_vii: 46, 1148, 1148,
2827
\__enumext_mini_addvspace_viii: 46, 1148, 1154,
3251
__enumext_mini_env* ..... 889
\__enumext_mini_right_cmd:n 47, 1174, 1176, 1176
\__enumext_mini_set_vskip: ... 42, 896, 896, 997
\__enumext_mini_set_vskip_vii: 45, 1091, 1091,
1150
\__enumext_mini_set_vskip_viii: 45, 1091, 1113,
1156
\__enumext_minipage:w 27, 226, 228, 891, 2738, 3114,
3478
\l__enumext_minipage_active_v_bool .. 76, 77,
2567, 2592, 2605, 2613
\g__enumext_minipage_active_vii_bool ... 82,
2838, 2843, 2855
\l__enumext_minipage_active_vii_bool . 2823,
2834
\g__enumext_minipage_active_viii_bool 3262,
3267, 3279
\l__enumext_minipage_active_viii_bool 3247,
3258
\g__enumext_minipage_active_X_bool ... 154
\l__enumext_minipage_active_X_bool .... 58
\g__enumext_minipage_after_skip 58, 1095, 1107,
2853, 3277
\l__enumext_minipage_after_skip 42, 43, 74, 77,
58, 912, 927, 947, 963, 978, 984, 990, 1004, 1014, 1023,
1026, 1038, 1056, 1067, 1083, 1115, 1128, 1142, 2490,
2622
\g__enumext_minipage_center_vii_bool . 2847,
2856
\g__enumext_minipage_center_viii_bool 3271,
3280
\g__enumext_minipage_center_X_bool ... 154
\l__enumext_minipage_hsep_v_dim ... 76, 2565
\l__enumext_minipage_hsep_vii_dim .... 2821
\l__enumext_minipage_hsep_viii_dim ... 3245
\l__enumext_minipage_left_skip 42, 76, 58, 904,
919, 938, 953, 1000, 1010, 1015, 1021, 1030, 1047,
1059, 1079, 1089, 1093, 1098, 1102, 1116, 1120, 1134,
1152, 1158
\l__enumext_minipage_left_v_dim 76, 2563, 2571
\l__enumext_minipage_left_vii_dim 2817, 2829
\l__enumext_minipage_left_viii_dim 3241, 3253
\l__enumext_minipage_left_X_dim ..... 58
\g__enumext_minipage_right_skip 58, 1094, 1099,
1103, 2846, 3270
\l__enumext_minipage_right_skip .. 42, 58, 908,

```

923, 943, 958, 1016, 1022, 1034, 1052, 1063, 1117,  
 1124, 1138, 1186, 1203  
 \l\_\_enumext\_minipage\_right\_v\_dim .. 76, 1197,  
 1202, 2561, 2565  
 \g\_\_enumext\_minipage\_right\_vii\_dim 82, 2825,  
 2845, 2858  
 \l\_\_enumext\_minipage\_right\_vii\_dim 82, 2815,  
 2820, 2826  
 \g\_\_enumext\_minipage\_right\_viii\_dim .. 3249,  
 3269, 3282  
 \l\_\_enumext\_minipage\_right\_viii\_dim .. 3239,  
 3244, 3250  
 \g\_\_enumext\_minipage\_right\_X\_dim ..... 154  
 \g\_\_enumext\_minipage\_right\_X\_skip .... 154  
 \g\_\_enumext\_minipage\_stat\_int . 73, 76, 58, 1191,  
 1208, 2422, 2483, 2488, 2568, 2615, 2620  
 \g\_\_enumext\_miniright\_code\_vii\_tl . 83, 2851,  
 2857  
 \g\_\_enumext\_miniright\_code\_viii\_tl 3275, 3281  
 \g\_\_enumext\_miniright\_code\_X\_tl ..... 154  
 \\_\_enumext\_multi\_addvspace: ... 40, 74, 844, 844,  
 2454  
 \\_\_enumext\_multi\_set\_vskip: .. 40, 808, 808, 846  
 \l\_\_enumext\_multicols\_above\_ii\_skip ... 827  
 \l\_\_enumext\_multicols\_above\_iii\_skip .. 833  
 \l\_\_enumext\_multicols\_above\_iv\_skip ... 839  
 \l\_\_enumext\_multicols\_above\_v\_skip 863, 877,  
 887  
 \l\_\_enumext\_multicols\_above\_X\_skip .... 52  
 \l\_\_enumext\_multicols\_below\_v\_skip 867, 881,  
 2607  
 \l\_\_enumext\_multicols\_below\_X\_skip .... 52  
 \\_\_enumext\_multicols\_start: 73, 2428, 2430, 2430  
 \\_\_enumext\_multicols\_stop: 74, 1181, 2460, 2460,  
 2492  
 \\_\_enumext\_newlabel:nn 24, 28, 60, 269, 269, 1820,  
 1923  
 \l\_\_enumext\_newlabel\_arg\_one\_tl 24, 28, 60, 62,  
 132, 1743, 1813, 1821, 1912, 1924, 1955  
 \l\_\_enumext\_newlabel\_arg\_two\_tl 24, 28, 59, 132,  
 1767, 1777, 1791, 1807, 1822, 1899, 1904, 1909, 1925  
 \\_\_enumext\_parse\_keys:n ..... 2313, 2344, 2344  
 \\_\_enumext\_parse\_keys\_vii:n .. 2863, 2900, 2900  
 \\_\_enumext\_parse\_keys\_viii:n . 3287, 3319, 3319  
 \\_\_enumext\_parse\_store\_keys:n 71, 72, 2350, 2353,  
 2353  
 \\_\_enumext\_parse\_store\_keys\_vii:n . 84, 2907,  
 2911, 2911  
 \l\_\_enumext\_parsep\_i\_skip . 825, 827, 976, 1024  
 \l\_\_enumext\_parsep\_ii\_skip ..... 831, 833, 982  
 \l\_\_enumext\_parsep\_iii\_skip .... 837, 839, 988  
 \l\_\_enumext\_parsep\_vii\_skip ..... 3116  
 \l\_\_enumext\_parsep\_viii\_skip ..... 3480  
 \l\_\_enumext\_partopsep\_v\_skip .. 879, 883, 1050,  
 1054, 1061, 1065, 1081, 1085  
 \l\_\_enumext\_partopsep\_viii\_skip ..... 1126  
 \\_\_enumext\_phantomsection: 28, 233, 262, 266, 282  
 \\_\_enumext\_print\_footnote: ... 2027, 2050, 3134,  
 3498  
 \\_\_enumext\_print\_keyans\_box:NN 56, 1632, 1632,  
 1645, 1833, 1981, 2003, 3416, 3437  
 \l\_\_enumext\_print\_keyans\_i\_tl .... 3549, 3578  
 \l\_\_enumext\_print\_keyans\_ii\_tl ... 3554, 3579  
 \l\_\_enumext\_print\_keyans\_iii\_tl .. 3559, 3580  
 \l\_\_enumext\_print\_keyans\_iv\_tl ... 3564, 3581  
 \l\_\_enumext\_print\_keyans\_vii\_tl .. 3569, 3582  
 \l\_\_enumext\_print\_keyans\_X\_tl ..... 95  
 \\_\_enumext\_printkeyans:nnn . 97, 3583, 3586, 3586  
 \\_\_enumext\_redefine\_item: . 66, 2103, 2103, 2259  
 \l\_\_enumext\_ref\_aux\_tl 143, 408, 410, 413, 429, 431,  
 434, 439, 441, 444  
 \l\_\_enumext\_ref\_key\_arg\_tl .. 143, 402, 407, 414,  
 426, 435, 445  
 \\_\_enumext\_regex\_label\_ref\_key: . 31, 397, 397,  
 409, 430, 440  
 \\_\_enumext\_register\_counter\_style:Nn .. 301,  
 301, 306, 307, 308, 309, 310  
 \\_\_enumext\_remove\_extra\_parsep\_vii: .. 2878,  
 3143, 3143  
 \\_\_enumext\_remove\_extra\_parsep\_viii: . 3301,  
 3507, 3507  
 \\_\_enumext\_renew\_footnote: ... 2027, 2031, 3086,  
 3459  
 \l\_\_enumext\_resume\_bool .... 22, 35, 1369, 2242  
 \\_\_enumext\_resume\_counter: . 51, 1327, 1360, 1360  
 \\_\_enumext\_resume\_counter\_star: ..... 1329  
 \\_\_enumext\_resume\_counter\_vii: 51, 1336, 1360,  
 1371  
 \g\_\_enumext\_resume\_int 22, 75, 35, 1364, 1375, 2243,  
 2496  
 \l\_\_enumext\_resume\_vii\_bool ... 35, 1380, 2289  
 \g\_\_enumext\_resume\_vii\_int .. 85, 35, 2290, 2951  
 \\_\_enumext\_safe\_exec: ..... 2312, 2329, 2329  
 \\_\_enumext\_safe\_exec\_vii: ... 2862, 2883, 2883  
 \\_\_enumext\_safe\_exec\_viii: ... 3286, 3305, 3305  
 \\_\_enumext\_set\_error:nn ..... 3664, 3674, 3676  
 \\_\_enumext\_set\_label\_ref:n ... 31, 405, 405, 477  
 \\_\_enumext\_set\_label\_ref\_h:n . 31, 424, 424, 497  
 \\_\_enumext\_set\_parse:n ..... 3647, 3664, 3664  
 \l\_\_enumext\_setkey\_tmpa\_int ... 90, 3640, 3644  
 \l\_\_enumext\_setkey\_tmpa\_seq 90, 3638, 3648, 3654,  
 3656, 3658, 3671  
 \l\_\_enumext\_setkey\_tmpa\_tl .... 90, 3646, 3650  
 \l\_\_enumext\_setkey\_tmpb\_seq 90, 3639, 3642, 3646,  
 3647  
 \l\_\_enumext\_setkey\_tmpb\_tl 90, 3666, 3668, 3669  
 \l\_\_enumext\_show\_answer\_bool . 106, 1494, 1498,  
 1517, 1521, 1840, 1977, 2658, 3408  
 \\_\_enumext\_show\_length:nnn .. 36, 206, 206, 3716,  
 3717, 3718, 3719, 3720, 3721, 3722, 3723, 3724, 3725,  
 3731, 3732, 3733, 3734, 3735, 3736, 3737, 3738, 3739,  
 3740  
 \l\_\_enumext\_show\_position\_bool . 63, 106, 1495,  
 1497, 1518, 1520, 1844, 1988, 2659, 3418  
 \g\_\_enumext\_standar\_bool ..... 20, 2341  
 \l\_\_enumext\_standar\_bool . 20, 1772, 1785, 1801,  
 2334, 2495, 2893  
 \\_\_enumext\_standard\_item\_vii:w 86, 3004, 3006,  
 3006  
 \\_\_enumext\_standard\_item\_viii:w . 92, 93, 3356,  
 3358, 3358  
 \g\_\_enumext\_starred\_bool . 83, 85, 20, 1416, 1763,  
 1773, 1803, 1893, 2473, 2897, 2958, 2963  
 \l\_\_enumext\_starred\_bool . 83, 85, 20, 1696, 1704,  
 1788, 1829, 2337, 2890, 2964  
 \\_\_enumext\_starred\_columns\_set\_vii: .. 2745,  
 2745, 2871

```

\__enumext_starred_columns_set_viii: . 3169,
    3169, 3294
\__enumext_starred_item:nn ... 2079, 2079, 2109
\__enumext_starred_item_exec: ... 3397, 3463
\__enumext_starred_item_exec_viii: ... 3397
\__enumext_starred_item_vii:w . 86, 3003, 3022,
    3022
\__enumext_starred_item_vii_aux_i:w .. 3022,
    3027, 3030
\__enumext_starred_item_vii_aux_ii:w . 3022,
    3028, 3033, 3035
\__enumext_starred_item_vii_aux_iii:w 3022,
    3038, 3047
\__enumext_starred_item_viii:w .. 92, 93, 3355,
    3374, 3374
\__enumext_starred_item_viii_aux_i:w . 3374,
    3379, 3382
\__enumext_starred_item_viii_aux_ii:w 3374,
    3380, 3390, 3392
\__enumext_starred_joined_item_vii:n . 81, 86,
    2764, 2764, 3001
\__enumext_starred_joined_item_viii:n 89, 92,
    3188, 3188, 3353
\__enumext_start_from:NNn 33, 518, 518, 531, 553
\__enumext_start_item_tmp_vii: 83, 2874, 2986,
    2986
\__enumext_start_item_tmp_viii: 91, 3297, 3338,
    3338
\__enumext_start_item_vii:w . 86, 87, 3014, 3019,
    3044, 3051, 3053, 3053
\__enumext_start_item_viii:w 93, 94, 3366, 3371,
    3395, 3440, 3440
\__enumext_start_list:nn 27, 69, 80, 220, 222, 2316,
    2521, 2672, 2866, 3289
\__enumext_start_mini_vii: . 84, 2813, 2813, 2944
\__enumext_start_mini_viii: 92, 3237, 3237, 3330
\__enumext_start_store_level: . 72, 2315, 2382,
    2382
\__enumext_start_store_level_vii: . 85, 2865,
    2966, 2966
\l__enumext_start_X_int ..... 70, 548
\__enumext_stop_item_tmp_vii: . 83, 85, 87, 2873,
    2877, 2988, 3055
\__enumext_stop_item_tmp_viii: 91, 92, 94, 3296,
    3300, 3340, 3442
\__enumext_stop_item_vii: 87, 88, 3055, 3119, 3119
\__enumext_stop_item_viii: .. 94, 95, 3442, 3483,
    3483
\__enumext_stop_list: .. 27, 220, 223, 2325, 2531,
    2685, 2879, 3302
\__enumext_stop_mini_vii: 82, 85, 2832, 2832, 2948
\__enumext_stop_mini_viii: . 92, 3237, 3256, 3334
\__enumext_stop_store_level: .. 72, 2326, 2382,
    2400
\__enumext_stop_store_level_vii: .. 85, 2880,
    2966, 2976
\l__enumext_store_active_bool 23, 50, 71, 84, 82,
    1350, 1362, 1373, 1658, 2348, 2385, 2536, 2543, 2631,
    2689, 2905, 2968, 2978, 3313
\__enumext_store_addto_prop:n 55, 61, 1551, 1552,
    1560, 1683, 1874, 3404
\__enumext_store_addto_seq:n 55, 63, 1561, 1561,
    1565, 1572, 1586, 1594, 1603, 1621, 1629, 1747, 1960
\l__enumext_store_ans_bool 116, 1389, 1436, 1568,
    1592, 1599, 1627, 1671, 3070
\l__enumext_store_anskey_arg_tl .. 23, 58, 82,
    1689, 1698, 1700, 1706, 1714, 1717, 1727, 1732, 1735,
    1741, 1747
\__enumext_store_anskey_code:nnnn . 57, 1677,
    1681, 1681
\__enumext_store_anskey_show_left:n 60, 1688,
    1838, 1838
\__enumext_store_anskey_show_wrap:n 60, 1826,
    1826, 1842, 1857
\l__enumext_store_columns_break_bool . 1652,
    1695
\l__enumext_store_columns_join_int 82, 1703,
    1708
\l__enumext_store_columns_sep_vii_bool 2926
\l__enumext_store_columns_sep_vii_dim 2931,
    2935
\l__enumext_store_columns_sep_X_bool ... 95
\l__enumext_store_columns_sep_X_dim .... 95
\l__enumext_store_columns_vii_bool ... 2913
\l__enumext_store_columns_vii_int 2918, 2922
\l__enumext_store_columns_X_bool ..... 95
\l__enumext_store_columns_X_int ..... 95
\__enumext_store_internal_ref: .. 57, 59, 1686,
    1749, 1749
\l__enumext_store_item_symbol_sep_dim 1650,
    1724, 1729
\l__enumext_store_item_symbol_tl . 1648, 1715,
    1719
\l__enumext_store_keyans_label_tl 23, 61-63,
    82, 1862, 1865, 1868, 1872, 1874, 1931, 1934, 1937,
    1941, 1951, 1960, 1961, 3384, 3399, 3402, 3404, 3406
\__enumext_store_level_close: . 55, 1566, 1590,
    2404
\__enumext_store_level_close_vii: 1597, 1625,
    2982
\__enumext_store_level_open: .. 54, 55, 72, 1566,
    1566, 2391, 2396
\__enumext_store_level_open_vii: .. 84, 1597,
    1597, 2972
\g__enumext_store_name_tl 23, 74, 82, 1454, 1457,
    1472, 1475, 2476, 2514, 2961, 3167
\l__enumext_store_name_tl 23, 50, 82, 1341, 1342,
    1344, 1346, 1348, 1366, 1377, 1554, 1556, 1563, 1815,
    1816, 1852, 1914, 1915, 1996, 2476, 2497, 2500, 2952,
    2955, 2961, 3426
\l__enumext_store_opt_vii_tl . 1601, 1611, 1617,
    1621, 2920, 2933
\l__enumext_store_opt_X_tl ..... 95
\l__enumext_store_ref_key_bool 57, 1503, 1684,
    1738, 1878, 1948
\l__enumext_store_upper_level_X_bool ... 95
\l__enumext_store_write_aux_file_tl 24, 60, 62,
    132, 1818, 1824, 1921, 1927
\__enumext_storing_set:n 50, 51, 1325, 1334, 1339,
    1339
\l__enumext_the_counter_vii_tl ..... 431
\l__enumext_the_counter_viii_tl ..... 441
\l__enumext_the_counter_X_tl ..... 143
\__enumext_tmp:n 30, 34, 44, 51, 52, 57, 64, 69, 70, 81,
    95, 105, 124, 129, 135, 139, 147, 153, 154, 173, 214,
    218, 631, 635, 1382, 1399, 1479, 1507, 1508, 1525,
    1751, 1758, 1759, 1780, 1794, 1797, 1809, 1880, 1887,
    2225, 2270, 2271, 2309

```

<code>\__enumext_tmp:nn</code>	333, 354, 355, 383, 384, 396, 548, 567, 612, 630, 688, 696, 697, 711, 776, 792, 793, 807, 1212, 1228, 1526, 1550, 2011, 2026
<code>\__enumext_tmp:nnn</code>	465, 481, 482, 483, 484, 485, 501, 502
<code>\__enumext_tmp:nnnnn</code>	568, 593, 596, 599, 601, 603, 606, 609
<code>\__enumext_tmp:w</code>	3532, 3535
<code>\l__enumext_tmpa_vii_int</code>	2755, 2758
<code>\l__enumext_tmpa_viii_int</code>	3179, 3182
<code>\l__enumext_tmpa_X_int</code>	154
<code>\l__enumext_topsep_v_skip</code>	865, 869, 1019, 1032, 1040, 1045, 1065, 1069, 2688, 2719
<code>\l__enumext_topsep_vii_skip</code>	1096, 1105, 1109
<code>\l__enumext_topsep_viii_skip</code>	1118, 1140, 1144
<code>\__enumext_use_key_ref:</code>	31, 417, 417, 2261
<code>\__enumext_use_key_ref_h:</code>	32, 449, 449, 2295
<code>\l__enumext_vspace_a_star_v_bool</code>	1261
<code>\l__enumext_vspace_a_star_vii_bool</code>	1283
<code>\l__enumext_vspace_a_star_viii_bool</code>	1294
<code>\l__enumext_vspace_a_star_X_bool</code>	70
<code>\__enumext_vspace_above:</code>	48, 1229, 1229, 2409
<code>\__enumext_vspace_above_v:</code>	49, 1257, 1257, 2559
<code>\l__enumext_vspace_above_v_skip</code>	1259, 1263, 1265
<code>\__enumext_vspace_above_vii:</code>	49, 1279, 1279, 2942
<code>\l__enumext_vspace_above_vii_skip</code>	1281, 1285, 1287
<code>\__enumext_vspace_above_viii:</code>	49, 1279, 1290, 3328
<code>\l__enumext_vspace_above_viii_skip</code>	1292, 1296, 1298
<code>\l__enumext_vspace_b_star_v_bool</code>	1272
<code>\l__enumext_vspace_b_star_vii_bool</code>	1305
<code>\l__enumext_vspace_b_star_viii_bool</code>	1316
<code>\l__enumext_vspace_b_star_X_bool</code>	70
<code>\__enumext_vspace_below:</code>	48, 1243, 1243, 2494
<code>\__enumext_vspace_below_v:</code>	49, 1268, 1268, 2627
<code>\l__enumext_vspace_below_v_skip</code>	1270, 1274, 1276
<code>\__enumext_vspace_below_vii:</code>	49, 1301, 1301, 2950
<code>\l__enumext_vspace_below_vii_skip</code>	1303, 1307, 1309
<code>\__enumext_vspace_below_viii:</code>	49, 1301, 1312, 3336
<code>\l__enumext_vspace_below_viii_skip</code>	1314, 1318, 1320
<code>\__enumext_widest_from:nnNn</code>	34, 532, 532, 547, 559
<code>\g__enumext_widest_label_tl</code>	22, 29, 40, 321, 325, 329
<code>\l__enumext_wrap_label_opt_v_bool</code>	2122
<code>\l__enumext_wrap_label_opt_vii_bool</code>	86, 3013
<code>\l__enumext_wrap_label_opt_viii_bool</code>	93, 3365
<code>\l__enumext_wrap_label_opt_X_bool</code>	70
<code>\l__enumext_wrap_label_v_bool</code>	2118, 2122, 2130, 2185
<code>\l__enumext_wrap_label_vii_bool</code>	86, 3012, 3017, 3025, 3103
<code>\l__enumext_wrap_label_viii_bool</code>	93, 3364, 3369, 3377, 3467
<code>\l__enumext_wrap_label_X_bool</code>	70
<code>\__enumext_wrapper_label_v:n</code>	2187, 2667
<code>\__enumext_wrapper_label_vii:n</code>	3106
<code>\__enumext_wrapper_label_viii:n</code>	3470
<code>\__enumext_zero_count_level:</code>	212, 212, 1459, 1477
<code>\__enumext_zero_parsep:</code>	43, 916, 971, 971
<code>enumext*</code>	4, 2860
<code>enumXi</code>	293
<code>enumXii</code>	293
<code>enumXiii</code>	293
<code>enumXiv</code>	293
<code>enumXv</code>	293
<code>enumXvi</code>	293
<code>enumXvii</code>	293
<code>enumXviii</code>	293
Environments provide by <b>enumext</b> :	
<code>enumext*</code>	21, 22, 24–26, 29–33, 35, 36, 38, 39, 45, 46, 49–59, 62, 64, 71, 72, 83–85, 87, 89, 90, 92, 96, 99, 101
<code>enumext</code>	21, 22, 24, 26, 29–32, 34–37, 39–48, 50–59, 62, 64–69, 71, 72, 75, 79, 81, 82, 85, 96, 99, 100
<code>keyans*</code>	21–23, 25, 26, 29–33, 35, 36, 38, 39, 45, 46, 49, 51, 54, 55, 61, 64, 91, 94, 99, 101
<code>keyanspic</code>	21–24, 29, 30, 33, 46, 50, 51, 55, 61–63, 78–80, 100
<code>keyans</code>	21–24, 26, 29, 30, 33–39, 41, 44–51, 54, 55, 61–63, 67–69, 75, 76, 78, 80, 82, 92, 99, 100
Environments:	
<code>enumext*</code>	70
<code>keyans*</code>	70
<code>list</code>	26, 27, 68, 69, 71
<code>lrbox</code>	81, 87, 88, 94, 95
<code>minipage</code>	26, 27, 39, 41, 78–81, 87, 88, 95
<code>multicols</code>	39–42, 47, 73, 74, 76, 77
exp commands:	
<code>\exp_after:wN</code>	3535
<code>\exp_args:Ne</code>	2346, 3523
<code>\exp_not:N</code>	151, 324, 413, 434, 444, 645, 659, 660, 671, 672, 683, 684, 1743, 1849, 1850, 1953, 1993, 1994, 3423, 3424, 3532
<code>\exp_not:n</code>	413, 414, 434, 435, 444, 445, 646, 1534, 1541, 1708, 1719, 1729, 1743, 1744, 1821, 1924, 1955, 1957, 2364, 2377, 2922, 2935
<b>F</b>	
<code>\fbox</code>	1484
file commands:	
<code>\file_input_stop:</code>	3833
<code>first</code>	697
<code>font</code>	333
<code>\footnote</code>	64
<code>\footnote</code>	64, 2035
<code>\footnotemark</code>	2045
<code>\footnotesize</code>	1850, 1994, 3424
<code>\footnotetext</code>	2029
<b>G</b>	
<code>\getkeyans</code>	13, 96, 3521
group commands:	
<code>\group_begin:</code>	1670, 1848, 1992, 3082, 3101, 3422, 3455, 3465, 3543, 3577
<code>\group_end:</code>	1679, 1855, 1999, 3111, 3123, 3429, 3475, 3487, 3545, 3584
<b>H</b>	
<code>\hbadness</code>	3130, 3494



hbox commands:

<code>\hbox_set:Nn</code>	313
---------------------------	-----

`\hfill` 363, 367, 372, 373, 1183, 1201, 1743, 1953, 2837, 3261

hook commands:

<code>\hook_gput_code:nnn</code>	9, 181, 185, 231
<code>\hook_gset_rule:nnnn</code>	232

`\hspace` 3141, 3505

`\hyperlink` 59, 63

`\hyperlink` 1743, 1953

`\hypertarget` 28

`\hypertarget` 261

**I**

`\IfHyperBoolean` 239

`\IfPackageLoadedTF` 11, 235, 249

`\ignorespaces` 648

int commands:

<code>\int_add:Nn</code>	2797, 3221
<code>\int_case:nn</code>	822, 973
<code>\int_compare:nNnTF</code>	427, 451, 898, 1017, 1162, 1166, 1170, 1451, 1469, 1662, 1666, 1863, 1897, 1902, 1907, 1932, 2332, 2388, 2393, 2432, 2448, 2462, 2483, 2508, 2544, 2548, 2577, 2602, 2615, 2635, 2639, 2694, 2767, 2777, 2793, 2886, 2970, 2980, 3136, 3145, 3161, 3191, 3201, 3217, 3308, 3314, 3500, 3509, 3644
<code>\int_compare_p:nNn</code>	1417, 1764, 1774, 1786, 1787, 1802, 1804, 2338, 2894
<code>\int_decr:N</code>	2796, 3220
<code>\int_eval:n</code>	1556, 1816, 1850, 1915, 1994, 2243, 2246, 2290, 2293, 2785, 3209, 3424
<code>\int_from_alph:n</code>	526, 540
<code>\int_from_roman:n</code>	528, 542
<code>\int_gadd:Nn</code>	1438, 2798, 3072, 3222
<code>\int_gincr:N</code>	1441, 1675, 1964, 2066, 2067, 2097, 2098, 2422, 2568, 2990, 3076, 3077, 3342
<code>\int_gset:Nn</code>	1364, 1375, 2043
<code>\int_gset_eq:NN</code>	1420, 1423, 1425, 1426, 1427, 1428, 1429, 1430, 2040, 2496, 2499, 2951, 2954
<code>\int_gzero:N</code>	216, 1191, 1208, 2488, 2620, 3154, 3518
<code>\int_if_exist:NTF</code>	1351, 1402, 1404, 1406, 1408, 1410, 1412, 2497, 2952
<code>\int_incr:N</code>	2331, 2540, 2693, 2885, 2989, 3307, 3341
<code>\int_mod:nn</code>	3147, 3511
<code>\int_new:N</code>	20, 21, 22, 23, 24, 36, 38, 58, 74, 86, 92, 100, 113, 114, 121, 122, 123, 126, 127, 140, 157, 158, 159, 160, 161, 1353, 1403, 1405, 1407, 1409, 1411, 1413
<code>\int_set:Nn</code>	522, 526, 528, 1446, 1463, 1531, 1703, 2732, 2733, 2755, 2766, 2772, 2788, 3130, 3179, 3190, 3196, 3212, 3494, 3640
<code>\int_set_eq:NN</code>	2359, 2795, 2917, 3219
<code>\int_step_function:nnN</code>	1780, 1794, 1809
<code>\int_step_inline:nnn</code>	2734, 3667
<code>\int_to_roman:n</code>	189, 1760, 1798
<code>\int_use:N</code>	899, 1366, 1377, 1439, 2246, 2265, 2293, 2347, 2433, 2442, 2457, 2463, 2770, 2771, 2783, 3073, 3194, 3195, 3207
<code>\int_zero:N</code>	3139, 3503
<code>\c_one_int</code>	1556, 1996, 2755, 2774, 2780, 2786, 2790, 2793, 3179, 3198, 3204, 3210, 3214, 3217
<code>\c_zero_int</code>	1417, 1764, 1774, 1786, 1787, 1802, 1804, 2338, 2894, 2970, 2980, 3150, 3514

`\item` 27, 37, 38, 56, 65, 78, 80, 81, 83, 91

`\item` 65, 66, 85, 87, 92, 94, 224, 1574, 1580, 1605, 1613, 1700, 1934, 1937, 2105, 2139, 2872, 2874, 3295, 3297, 3406

`\item*` 5, 11, 2137

`item-pos*` 2011

<code>item-sym*</code>	2011
<code>\itemindent</code>	22, 69
<code>\itemindent</code>	68
<code>itemindent</code>	612
<code>\itemsep</code>	79, 80
<code>\itemsep</code>	2708, 2714
<code>\itemwidth</code>	2762, 2806, 2810, 3186, 3230, 3234

K

<code>keyans</code>	11, 2516
<code>keyans*</code>	11, 3284
<code>keyanspic</code>	12, 2669

Keys for environments provide by [enumext](#):

<code>above*</code>	23, 48, 49
<code>above</code>	23, 48, 49, 73, 76, 84, 92
<code>after</code>	37, 38, 75, 78, 85, 92
<code>align</code>	23, 30, 67, 87
<code>before*</code>	37, 38, 73, 84, 92
<code>before</code>	37, 38, 76
<code>below*</code>	23, 48, 49
<code>below</code>	23, 48, 49, 75, 78, 85, 92
<code>check-ans</code>	23, 24, 26, 50, 52, 57, 63, 65, 66, 73–75, 89, 100
<code>columns-sep*</code>	24, 54, 72, 84
<code>columns-sep</code>	39, 55, 72, 73, 77, 84
<code>columns*</code>	24, 54, 72, 84
<code>columns</code>	22, 39, 42, 48, 55, 72, 73, 76, 84
<code>first</code>	37, 38, 87
<code>font</code>	30, 67, 87
<code>item-pos*</code>	57, 58, 64
<code>item-sym*</code>	22, 57, 58, 64, 65
<code>item*-sep</code>	65
<code>itemindent</code>	23, 35, 36, 67, 87
<code>itemsep</code>	34, 70
<code>labelsep</code>	30, 65, 69, 87
<code>labelwidth</code>	29, 30, 32–34, 69
<code>label</code>	22, 29, 33, 34, 81
<code>lisparindent</code>	70
<code>list-indent</code>	22, 35, 80
<code>list-offset</code>	35
<code>listparindent</code>	35, 87
<code>mark-ans</code>	24, 53, 60
<code>mark-pos</code>	53, 54
<code>mark-ref</code>	24, 53, 59
<code>mini-env</code>	23, 39, 41, 47, 48, 64, 73, 76, 82, 84, 90, 92
<code>mini-sep</code>	23, 39, 73, 76
<code>miniright*</code>	23, 39
<code>miniright</code>	23, 39, 46, 83
<code>minirigth*</code>	26
<code>minirigth</code>	26
<code>no-store</code>	24, 51–53
<code>noitemsep</code>	34, 43
<code>nosep</code>	34, 43
<code>parindent</code>	70
<code>parsep</code>	34, 70, 87
<code>partopsep</code>	34
<code>ref</code>	25, 31
<code>resume</code>	22, 50, 51, 69, 75, 85
<code>rightmargin</code>	35
<code>save-ans</code>	23, 50, 51, 55, 57, 61, 62, 66, 75, 78, 85, 92, 96
<code>save-key</code>	24
<code>show-ans</code>	24, 53, 54, 56, 58, 60, 67
<code>show-length</code>	26, 36, 69, 99
<code>show-pos</code>	24, 53, 54, 56, 58, 60, 67
<code>start</code>	23, 26, 33, 34, 69
<code>store-brk</code>	57, 58

store-ref ..... 24, 28, 53, 57, 59, 61, 63, 67  
topsep ..... 34  
widest ..... 22, 26, 34  
wrap-ans ..... 53, 56, 60  
wrap-label\* ..... 30, 65, 67, 86, 87, 93  
wrap-label ..... 30, 67, 86, 87, 93

keys commands:

\keys\_define:nn 335, 357, 386, 467, 487, 503, 550, 570,  
614, 633, 690, 699, 778, 795, 1214, 1323, 1332, 1384,  
1481, 1510, 1528, 1646, 2013, 3547, 3610  
\l\_keys\_key\_str ..... 3701  
\keys\_set:nn . 349, 802, 1219, 1224, 1692, 2346, 2555,  
2904, 3323, 3612, 3613, 3614, 3615, 3616, 3617, 3618,  
3619, 3620, 3621, 3622, 3623, 3661

L

label ..... 465, 485, 503

Labels provide by enumext:

\Alph\* ..... 29  
\Roman\* ..... 29  
\alph\* ..... 29  
\arabic\* ..... 29, 31  
\roman\* ..... 29  
\labelsep ..... 80  
\labelsep ..... 2709, 2712  
labelsep ..... 333  
\labelwidth ..... 29, 80  
\labelwidth ..... 2709, 2710  
labelwidth ..... 333  
\leftmargin ..... 22, 69  
\leftmargin ..... 68, 2709  
legacy commands:  
  \legacy\_if:nTF ..... 3056, 3059, 3443, 3446  
  \legacy\_if\_gset\_false:n ..... 892  
  \legacy\_if\_set\_false:n ..... 3058, 3445  
  \legacy\_if\_set\_true:n 3018, 3043, 3050, 3063, 3370,  
  3394, 3450  
\linewidth ..... 73, 76  
\linewidth .... 2417, 2565, 2731, 2758, 2819, 3182, 3243  
\list ..... 27  
\list ..... 222  
list-indent ..... 612  
list-offset ..... 612  
\listparindent ..... 2711  
listparindent ..... 612  
\lrbox ..... 3083, 3456

M

\makebox ..... 81  
\makebox .. 1636, 1638, 2159, 3097, 3105, 3109, 3469, 3473  
\makelabel ..... 65, 67, 68, 81  
\makelabel ..... 67, 68, 2165, 2181  
\makesavenoteenv ..... 255  
mark-ans ..... 1479  
mark-pos ..... 1479, 1508  
mark-ref ..... 1479  
mini-env ..... 776  
mini-sep ..... 776  
\minipage ..... 27  
\minipage ..... 228  
\miniright ..... 9, 46, 1160, 2486, 2618  
\miniright\* ..... 9  
mode commands:  
  \mode\_if\_vertical:TF ..... 847, 875, 998, 1077

\mode\_leave\_vertical: .. 645, 659, 671, 683, 1605,  
1613, 1634, 2157, 3095  
msg commands:  
  \msg\_error:nn .. 2546, 2550, 2637, 2696, 2888, 3310,  
  3316, 3624  
  \msg\_error:nnn . 1164, 1168, 1193, 1210, 3537, 3542,  
  3607, 3677  
  \msg\_error:nnnn 1660, 1664, 1668, 2538, 2633, 2641  
  \msg\_fatal:nn ..... 2333  
  \msg\_fatal:nnn ..... 287  
  \msg\_info:nnn ..... 13, 16, 237, 251  
  \msg\_line\_context: .. 3705, 3710, 3715, 3730, 3753,  
  3757, 3761, 3766, 3771, 3776, 3781, 3785, 3790, 3795,  
  3799, 3804, 3808, 3813, 3818, 3823, 3827, 3831  
  \msg\_new:nnn 3678, 3682, 3686, 3690, 3695, 3699, 3703,  
  3708, 3713, 3728, 3743, 3750, 3755, 3759, 3764, 3769,  
  3774, 3779, 3783, 3788, 3793, 3797, 3802, 3806, 3811,  
  3816, 3821, 3825, 3829  
  \msg\_term:nnn ..... 1454, 1472  
  \msg\_term:nnnn ..... 2255, 2265, 2300, 2305  
  \msg\_warning:nn ..... 2485, 2617  
  \msg\_warning:nnn ..... 1457, 1475  
  \msg\_warning:nnnn 1971, 2197, 2202, 2769, 2782, 3193,  
  3206  
\multicolsep ..... 73, 77  
\multicolsep ..... 2447, 2590

N

\NeedsTeXFormat ..... 3  
\newcounter ..... 290  
\NewDocumentCommand 1160, 1656, 2629, 3521, 3575, 3631  
\NewDocumentEnvironment . 2310, 2516, 2669, 2860, 3284  
\newlabel ..... 28  
\newlabel ..... 273  
no-store ..... 1382  
\noindent ..... 83, 91  
\noindent . 2424, 2570, 2828, 2873, 3138, 3252, 3296, 3502  
\nointerlineskip ..... 2424, 2570, 2828, 3252  
noitemsep ..... 568  
\nopagebreak ..... 858, 886, 1009, 1088, 1151, 1157  
\normalfont ..... 1849, 1993, 3423  
nosep ..... 568

P

Packages:

enumext ..... 21, 50, 68, 69, 78, 99  
enumitem ..... 28, 29  
expl3 ..... 81  
footnotehyper ..... 27  
hyperref ..... 24, 26–28, 31, 59, 63, 87, 94, 99  
lua-visual-debug ..... 41  
multicol ..... 21, 99  
shortlst ..... 81  
\par 858, 886, 1009, 1088, 1151, 1157, 1186, 1203, 1828, 2468,  
2490, 2607, 2622, 2743, 2846, 2853, 3138, 3152, 3270,  
3277, 3502, 3516  
\parindent ..... 3115, 3479  
\parsep ..... 40, 43, 79, 80  
\parsep ..... 1606, 1614, 2285, 2708, 2715, 2720  
parsep ..... 568  
\parskip ..... 3116, 3480  
\partopsep ..... 80  
\partopsep ..... 2286, 2713  
partopsep ..... 568

## peek commands:

\peek_meaning:NTF	2995, 3009, 3026, 3037, 3347, 3361, 3378
\peek_meaning_remove:NTF	3002, 3354
\peek_remove_spaces:n	2143
\phantomsection	28
\phantomsection	262
prg commands:	
\prg_do_nothing:	266
\prg_new_protected_conditional:Npnn	200
\prg_replicate:nn	209, 3748
\prg_return_false:	204
\prg_return_true:	203
\printkeyans	13, 96, 3575
prop commands:	
\prop_count:N	1556, 1816, 1852, 1915, 1996, 3426
\prop_gput_if_not_in:Nnn	1551, 1554
\prop_if_exist:NTF	1342, 3541
\prop_item:Nn	3544
\prop_new:N	1344
\ProvidesExplPackage	4

## R

\raggedcolumns	2456, 2596
\ref	59, 61
ref	465, 485
\refstepcounter	3065, 3452
regex commands:	
\regex_match:nnTF	202, 525, 527, 539, 541, 2357, 2370, 2915, 2928
\regex_replace_once:nnN	401
\renewcommand	413, 434, 444
\RenewDocumentCommand	2035, 2105, 2139, 2165, 2181
\RequirePackage	17
resume	1323
resume*	1323
rightmargin	612
\Roman	29, 33, 34
\Roman	309
\roman	29, 33, 34
\roman	310, 483, 3563

## S

save-ans	1323
scan commands:	
\scan_stop:	80, 2722, 2872, 3295, 3532, 3535
seq commands:	
\seq_clear:N	3638
\seq_const_from_clist:Nn	3626
\seq_count:N	2682, 3642
\seq_gclear:N	2033, 2034
\seq_gput_right:Nn	1563, 2046, 2047
\seq_if_empty:NTF	2052, 3590, 3656
\seq_if_exist:NTF	1346, 3588
\seq_item:Nn	2740
\seq_map_function:NN	3647
\seq_map_inline:Nn	3596, 3601, 3635, 3657, 3658
\seq_map_pairwise_function:NNN	2054
\seq_new:N	93, 94, 111, 141, 142, 1348
\seq_pop_left:NN	3646
\seq_put_right:Nn	2643, 3654, 3671
\seq_set_from_clist:Nn	3639
\seq_set_map_e:NNn	3648
\seq_show:N	3592
\setcounter	536, 540, 542, 2243, 2245, 2290, 2292, 2687
\setenumext	5-8, 97, 3551, 3556, 3561, 3566, 3571, 3631

\setlength	1607, 1615
show-ans	1479, 1508
show-length	688

## skip commands:

\skip_add:Nn	827, 833, 839, 849, 853, 877, 881, 978, 984, 990, 1000, 1004, 1026, 1079, 1083, 2708
\skip_eval:n	1606, 1614
\skip_gset:Nn	1099, 1103, 1107
\skip_gzero_new:N	1094, 1095
\skip_horizontal:N	660, 672, 684, 3098, 3112, 3476
\skip_horizontal:n	646, 1635, 1643, 2158, 2160, 3096
\skip_if_eq:nnTF	825, 831, 837, 901, 935, 976, 982, 988, 1019, 1024, 1045, 1096, 1118, 1231, 1245, 1259, 1270, 1281, 1292, 1303, 1314
\skip_new:N	54, 55, 59, 60, 61, 62, 63, 115, 171
\skip_set:Nn	810, 814, 863, 867, 904, 908, 912, 919, 923, 927, 938, 943, 947, 953, 958, 963, 1021, 1022, 1023, 1030, 1034, 1038, 1047, 1052, 1056, 1059, 1063, 1067, 1098, 1102, 1120, 1124, 1128, 1134, 1138, 1142, 2702, 2716
\skip_set_eq:NN	2238, 2284, 2285, 3115, 3116, 3479, 3480
\skip_use:N	812, 816, 851, 855, 859, 879, 883, 902, 921, 930, 936, 941, 945, 956, 960, 961, 966, 1002, 1006, 1032, 1232, 1236, 1239, 1246, 1250, 1253, 2468
\skip_zero:N	2286, 2447, 2590, 2713, 2714
\skip_zero_new:N	1014, 1015, 1016, 1093, 1115, 1116, 1117
\c_zero_skip	825, 831, 837, 902, 936, 976, 982, 988, 1019, 1024, 1045, 1096, 1118, 1232, 1246, 1259, 1270, 1281, 1292, 1303, 1314
\small	3553, 3558, 3563, 3568, 3573
\star	2017
start	548
\stepcounter	2039, 2650
store-ref	1479
str commands:	
\c_backslash_str	3757, 3766, 3767, 3771, 3772, 3776, 3777, 3808, 3809, 3813, 3818, 3819
\c_colon_str	1815, 1914, 3532
\str_count:n	209, 3748
\str_if_eq:nnTF	2248, 2296
\str_if_eq_p:nn	2241, 2289
\str_if_in:nnTF	3528
\str_new:N	110, 166
\str_set:Nn	389, 390, 391, 1490, 1491, 1513, 1514
\string	255
\strutbox	906, 910, 914, 925, 929, 940, 949, 955, 965, 978, 984, 990, 1021, 1022, 1023, 1026, 1036, 1040, 1049, 1056, 1061, 1069, 1098, 1099, 1102, 1109, 1122, 1130, 1136, 1144, 2718

## T

TeX and L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> commands:

\@auxout	271
\protected@write	271
text commands:	
\text_expand:n	3524
\textasteriskcentered	1487, 1501
\thepage	277
tl commands:	
\c_space_tl	1872, 1941, 1986, 2008, 3387, 3388, 3715, 3730
\tl_clear:N	362, 368, 1689, 1862, 1931, 3384
\tl_clear_new:N	319



`\tl_const:Nn` ..... 143, 303  
`\tl_gclear:N` 1973, 2176, 2514, 2857, 3099, 3167, 3281  
`\tl_gput_right:Nn` ..... 304  
`\tl_greplace_all:Nnn` ..... 325  
`\tl_gset:Nn` ..... 1961, 2476, 2961, 3032  
`\tl_gset_eq:NN` ..... 321, 2085, 3092  
`\tl_if_blank:nTF` ..... 3090, 3400, 3410, 3431  
`\tl_if_empty:nTF` . 419, 453, 459, 1570, 1601, 1715, 1969, 2155, 3669  
`\tl_if_novalue:nTF` . . 1690, 1701, 1870, 1939, 1985, 2007, 2037, 2062, 2081, 2086, 2116, 2680, 2902, 3321, 3385, 3633  
`\tl_map_inline:Nn` ..... 322, 399  
`\tl_new:N` 32, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 90, 91, 97, 98, 108, 109, 120, 132, 133, 134, 137, 145, 146, 149, 150, 165, 168  
`\tl_put_left:Nn` 1578, 1611, 1698, 1979, 2001, 3399, 3406  
`\tl_put_right:Nn` 320, 411, 432, 442, 1532, 1539, 1582, 1617, 1700, 1706, 1714, 1717, 1727, 1732, 1735, 1741, 1767, 1777, 1791, 1807, 1813, 1818, 1865, 1868, 1872, 1899, 1904, 1909, 1912, 1921, 1934, 1937, 1941, 1951, 1986, 2008, 2362, 2375, 2920, 2933, 3402, 3412, 3433, 3549, 3554, 3559, 3564, 3569  
`\tl_remove_all:Nn` ..... 3668  
`\tl_remove_once:Nn` ..... 1755, 1884  
`\tl_replace_all:Nnn` ..... 324  
`\tl_reverse:N` ..... 1754, 1756, 1883, 1885  
`\tl_set:Nn` 151, 289, 363, 367, 372, 373, 407, 426, 643, 657, 669, 681, 1341, 1486, 1500, 1846, 1990, 2083, 3387, 3388, 3420, 3666  
`\tl_set_eq:NN` 330, 408, 410, 429, 431, 439, 441, 1753, 1882, 1895, 2128, 2132, 2661, 2663

`\tl_to_str:n` ..... 3524  
`\tl_trim_spaces:n` ..... 320, 3654, 3666, 3672  
`\tl_use:N` . 326, 329, 421, 455, 461, 714, 718, 722, 726, 730, 734, 738, 742, 746, 750, 754, 758, 762, 766, 770, 774, 1640, 1760, 1768, 1779, 1793, 1798, 1810, 2070, 2076, 2101, 2119, 2123, 2131, 2159, 2167, 2168, 2175, 2183, 2184, 2190, 2317, 2522, 2666, 2851, 3102, 3113, 3117, 3275, 3413, 3434, 3466, 3477, 3481, 3578, 3579, 3580, 3581, 3582, 3650

token commands:

`\token_to_str:N` ..... 273  
`\topsep` ..... 1607, 1615  
`topsep` ..... 568  
`\typeout` ..... 241, 244, 254, 255

U

`\u` ..... 402  
use commands:  
`\use:N` ..... 210, 2172, 2319  
`\use:n` ..... 3530  
`\use_none:nn` ..... 265  
`\usecounter` ..... 2239, 2287

V

`\value` ..... 2496, 2501, 2951, 2956  
`\vspace` 893, 1236, 1239, 1250, 1253, 1263, 1265, 1274, 1276, 1285, 1287, 1296, 1298, 1307, 1309, 1318, 1320, 1606, 1614, 2677, 2688, 3153, 3517

W

`widest` ..... 548  
`wrap-ans` ..... 1479  
`wrap-label` ..... 333  
`wrap-label*` ..... 333