

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-04-12<sup>\*</sup>

©2024 by Pablo González<sup>†</sup>

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

## Abstract

This package provides “*enumerated list*” environments for creating “*simple exercise sheets*” along with “*multiple choice questions*”, storing the *(answers)* to these in memory using the **multicol** package and the **l3seq** and **l3prop** modules.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>	<b>4</b>	<b>The storage system</b>	<b>9</b>
1.1	Description and usage	3	4.1	Keys for storage	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</b>	<b>The environment <code>enumext</code></b>	<b>4</b>	4.7	Printing stored content	13
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	13
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	13
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>5</b>	<b>5</b>	<b>Full examples</b>	<b>14</b>
3.1	Keys for <code>label</code> and <code>ref</code>	6	<b>6</b>	<b>The way of non-enumerated lists</b>	<b>16</b>
3.2	Keys for spaces	6	<b>7</b>	<b>References</b>	<b>18</b>
3.2.1	Vertical spaces	7	<b>8</b>	<b>Change history</b>	<b>18</b>
3.2.2	Horizontal spaces	8	<b>9</b>	<b>Index of Documentation</b>	<b>19</b>
3.3	Keys for add code	8	<b>10</b>	<b>Implementation</b>	<b>21</b>
3.4	Keys for start and resume	9	<b>11</b>	<b>Index of Implementation</b>	<b>94</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  $\TeX$  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  $\TeX$  distribution such as  $\TeX$  Live or Mi $\TeX$ . It has been tested with the standard classes provided by  $\TeX$ : **book**, **report**, **article** and **letter** on 10pt, 11pt and 12pt.

<sup>\*</sup>This file describes a documentation for v1.0, last revised 2024-04-12.

<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.  $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II.  $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

3. Third type of questions

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

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

(A) value

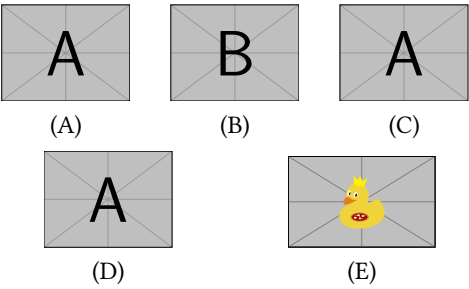
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
- 

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

1. (B)  $x = 5$

2. (D)

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

5. (D) “other note”

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

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

1.1 Description and usage

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

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

<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.

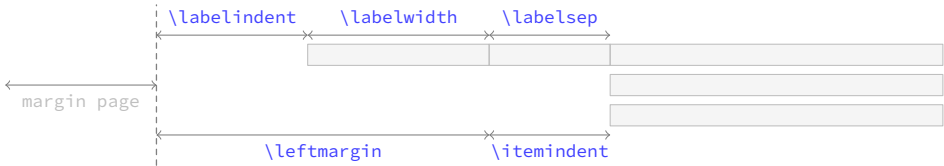


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

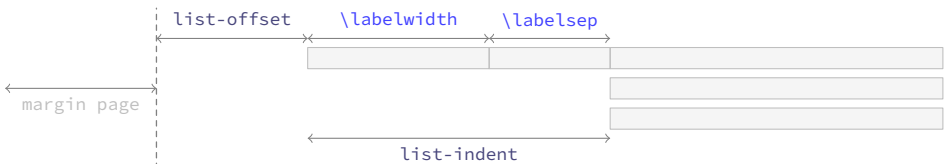


Figure 2: Representation of horizontal lengths concept in `enumext`.

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  $[\langle key = val \rangle]$  options.

1.3.1 Internal counters

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

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

### 1.3.2 Support for multicols

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

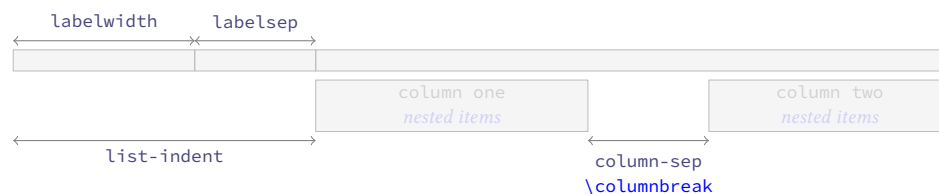


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

### 1.3.3 Support for minipage

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



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

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

### 1.3.4 The \label and \ref system

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

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

### 1.3.5 Support for \footnote

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

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

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

## 2 The environment enumext

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

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

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

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

### Example

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

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

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

---

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

---

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

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

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

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

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

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

## 3 The command `\setenumext`

---

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

---

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

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

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

### 3.1 Keys for label and ref

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

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

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

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

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

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

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

`labelsep = { $\langle rigid length \rangle$ }` default: *0.3333em*

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

`labelwidth = { $\langle rigid length \rangle$ }` default: *by label*

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

`widest = { $\langle integer | string \rangle$ }` default: *empty*

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

`font = { $\langle font commands \rangle$ }` default: *empty*

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

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

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

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

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

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

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

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

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

### 3.2 Keys for spaces

`show-length = { $\langle true | false \rangle$ }` default: *false*

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

### 3.2.1 Vertical spaces

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

default: *by levels*

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

`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  $4.0\text{pt}$  plus  $2.0\text{pt}$  minus  $1.0\text{pt}$ , for second level are  $2.0\text{pt}$  plus  $1.0\text{pt}$  minus  $1.0\text{pt}$ , for third and fourth level are  $0\text{pt}$ .

`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  $2.0\text{pt}$  plus  $1.0\text{pt}$  minus  $1.0\text{pt}$ , for third and fourth level are  $1.0\text{pt}$  minus  $1.0\text{pt}$ .

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

- |  |  |
|--|--|
| <p>★ 1. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* <span style="border: 1px solid black; padding: 2px;">first answer</span></p> <p>2. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">(a) Question.</p> <p style="margin-left: 40px;">* <span style="border: 1px solid black; padding: 2px;">second answer</span></p> | <p>3. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* <span style="border: 1px solid black; padding: 2px;">third answer</span></p> <p>4. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* <span style="border: 1px solid black; padding: 2px;">fourth answer</span></p> |
|--|--|

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

## 4.5 The environment keyans

---

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

---

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

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

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

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

---

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

---

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

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

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

### Example

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

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

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

1. Text containing a question.

(A) Choice

\* (B) Correct choice

(C) Choice

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

(A) Choice

(B) Choice

(C) Choice

(D) Choice

\* (E) [note] Correct choice



4.6 The environment keyanspic

keyanspic

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

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

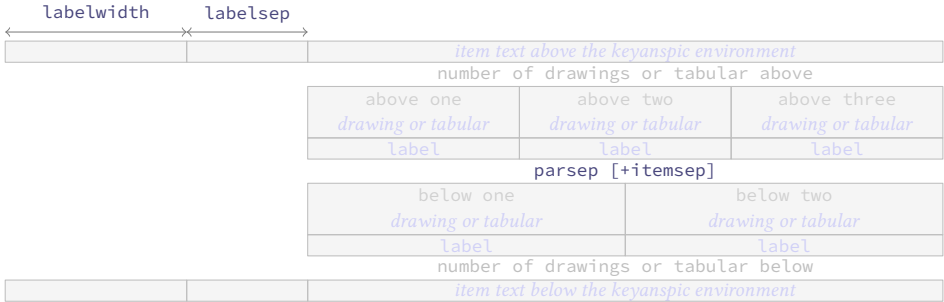


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

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

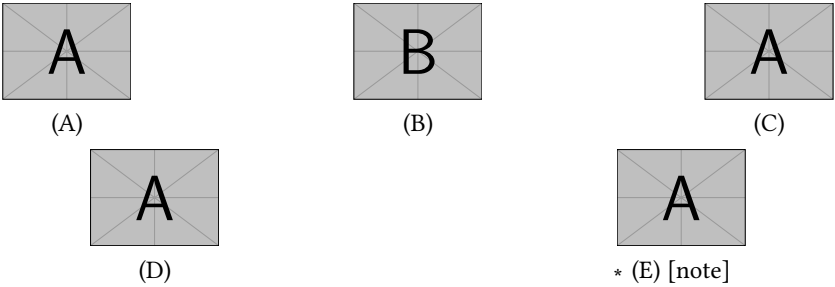
The `\anspic` command take three arguments, the *starred version* “\*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

The “content” can only be accessed “after” it is stored, if the `<store name>` does not exist the command will return an error. The form taken by the argument `<store name> : <position>` is the same as that used to generate the internal “label and ref” system when `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.  $3(x + y + z)$

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- \*

\*

\*

\*

\*

5 Full examples

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

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

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

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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


C

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

D

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

Example 2

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

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

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

☒ A

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

B

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

C

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

D

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

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

☒ A

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

B

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

C

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

D

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

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

©2024 by Pablo González L

14 / 104



Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

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

II.  $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

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

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

A

 value

B

 value

C

 value

D

 value

E

 value
4. Question with image and label below:



A



B



C



D



E

5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

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

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

 True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?
- 4

 Related to Linux
- (a)

 You use linux?
- (b)

 Usually uses the package manager?
- (c)

 Rate the following package and class
- i.

 xsim-exam
- ii.

 xsim
- iii.

 exsheets

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

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

 Yes, dnf
- (c)

 i. doesn't exist for now :(
- ii. very good
- iii. obsolete

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

Here we set the label key using the default settings in L<sup>A</sup>T<sub>E</sub>X for the four levels \textbullet, \textendash, \textasteriskcentered and \textperiodcentered together with the nosepe key to reduce the vertical spaces in the left side example and set the label key in mathematical mode for the right side as \ast, \diamond, \circ and \star for the four levels together with the nosepe key

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

Fake description environment

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

- Something** A short one-line description.

This is an entry without a label.

**Something** A short one-line description text.

**Something long** A much longer description text may take more than one line or more than one paragraph.

    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

- Something** A short one-line description.

This is an entry without a label.

**Something** A short one-line description text.

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

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

Description indented by label

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

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

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

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

**Something** A short one-line description.

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

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

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

**Something** A short one-line description.

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

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

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

**Something** A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

**Something** A short one-line description.

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

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

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

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

Final notes

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

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

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

### Why list environments?

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

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

### Why not random questions and other utilities

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

## 7 References

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

## 8 Change history

**v1.0** 2024-04-12 – 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 / 104

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

### 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]
```

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

```
4 \IfPackageLoadedTF {multicol}
5 {
6   %%%\msg_info:nnn { enumext } { package-load } { multicol }
7 }
8 {
9   \RequirePackage{multicol}[2023-03-30]
10 }
```

Finally we declare the `enumext` package.

```
11 \ProvidesExplPackage
12 {enumext}
13 {2024-04-12}
14 {1.0}
15 {Enumerate exercise sheets}
```

### 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.

```
16 \int_new:N \l__enumext_level_int
17 \int_new:N \l__enumext_level_h_int
18 \int_new:N \l__enumext_keyans_level_int
19 \int_new:N \l__enumext_keyans_level_h_int
20 \int_new:N \l__enumext_keyans_pic_level_int
21 \bool_new:N \l__enumext_starred_bool
22 \bool_new:N \g__enumext_starred_bool
23 \bool_new:N \l__enumext_standar_bool
24 \bool_new:N \g__enumext_standar_bool
25 \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).

```

26 \cs_set_protected:Npn \__enumext_tmp:n #1
27 {
28   \tl_new:c { l__enumext_counter_#1_tl }
29 }
30 \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).

```

31 \bool_new:N \l__enumext_resume_bool
32 \int_new:N \g__enumext_resume_int
33 \bool_new:N \l__enumext_resume_vii_bool
34 \int_new:N \g__enumext_resume_vii_int
35 \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.

```

36 \dim_new:N \l__enumext_current_widest_dim
37 \tl_new:N \g__enumext_counter_styles_tl
38 \tl_new:N \g__enumext_widest_label_tl
39 \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`.

```

40 \cs_set_protected:Npn \__enumext_tmp:n #1
41 {
42   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
43   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
44   \dim_new:c { l__enumext_leftmargin_#1_dim }
45   \dim_new:c { l__enumext_itemindent_#1_dim }
46 }
47 \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).

```

48 \cs_set_protected:Npn \__enumext_tmp:n #1
49 {
50   \skip_new:c { l__enumext_multicols_above_#1_skip }
51   \skip_new:c { l__enumext_multicols_below_#1_skip }
52 }
53 \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).

```

54 \int_new:N \g__enumext_minipage_stat_int
55 \skip_new:N \l__enumext_minipage_left_skip
56 \skip_new:N \l__enumext_minipage_right_skip
57 \skip_new:N \l__enumext_minipage_after_skip
58 \skip_new:N \g__enumext_minipage_right_skip
59 \skip_new:N \g__enumext_minipage_after_skip
60 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

61 {
62   \dim_new:c { \__enumext_minipage_left_#1_dim }
63   \bool_new:c { \__enumext_minipage_active_#1_bool }
64 }
65 \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.)

```

\__enumext_wrap_label_X_bool
\__enumext_wrap_label_opt_X_bool
\__enumext_start_X_int
\__enumext_fake_item_indent_X_tl
\__enumext_label_fill_left_X_tl
\__enumext_label_fill_right_X_tl
\__enumext_vspace_a_star_X_bool
\__enumext_vspace_b_star_X_bool

```

The integer variable `\__enumext_start_X_int` are used by the `start` key (§10.12), the token list `\__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\__enumext_label_fill_left_X_tl` and `\__enumext_label_fill_right_X_tl` are used by the `align` key (§10.10). The boolean vars `\__enumext_vspace_a_star_X_bool`, `\__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```

66 \cs_set_protected:Npn \__enumext_tmp:n #1
67 {
68   \bool_new:c { \__enumext_wrap_label_#1_bool }
69   \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
70   \int_new:c { \__enumext_start_#1_int }
71   \tl_new:c { \__enumext_fake_item_indent_#1_tl }
72   \tl_new:c { \__enumext_label_fill_left_#1_tl }
73   \tl_new:c { \__enumext_label_fill_right_#1_tl }
74   \bool_new:c { \__enumext_vspace_a_star_#1_bool }
75   \bool_new:c { \__enumext_vspace_b_star_#1_bool }
76 }
77 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_store_active_bool
\__enumext_store_name_tl
\g__enumext_store_name_tl
\__enumext_store_anskey_arg_tl
\__enumext_store_columns_join_int
\__enumext_store_keyans_label_tl
\__enumext_keyans_tmpa_tl

```

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

The variable `\__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 `\__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\__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 `\__enumext_keyans_tmpa_tl` is a temporary variable used by `keyans` and `keyanspic` at various points.

```

78 \bool_new:N \__enumext_store_active_bool
79 \tl_new:N \__enumext_store_name_tl
80 \tl_new:N \g__enumext_store_name_tl
81 \tl_new:N \__enumext_store_anskey_arg_tl
82 \int_new:N \__enumext_store_columns_join_int
83 \tl_new:N \__enumext_store_keyans_label_tl
84 \tl_new:N \__enumext_keyans_tmpa_tl

```

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

```

\__enumext_setkey_tmpa_tl
\__enumext_setkey_tmpp_tl
\__enumext_setkey_tmpa_int
\__enumext_setkey_tmpa_seq
\__enumext_setkey_tmpp_seq

```

Internal variables used by the command `\setenumext` (§10.38).

```

85 \tl_new:N \__enumext_setkey_tmpa_tl
86 \tl_new:N \__enumext_setkey_tmpp_tl
87 \int_new:N \__enumext_setkey_tmpa_int
88 \seq_new:N \__enumext_setkey_tmpa_seq
89 \seq_new:N \__enumext_setkey_tmpp_seq

```

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

```

\__enumext_store_opt_X_tl
\__enumext_print_keyans_X_tl
\__enumext_store_columns_X_bool
\__enumext_store_columns_X_int
\__enumext_store_columns_sep_X_bool
\__enumext_store_columns_sep_X_dim
\__enumext_store_upper_level_X_bool

```

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.37) and the keys `columns*` and `columns-sep*`.

```

90 \cs_set_protected:Npn \__enumext_tmp:n #1
91 {
92   \tl_new:c { \__enumext_store_opt_#1_tl }
93   \tl_new:c { \__enumext_print_keyans_#1_tl }
94   \bool_new:c { \__enumext_store_columns_#1_bool }
95   \int_new:c { \__enumext_store_columns_#1_int }
96   \bool_new:c { \__enumext_store_columns_sep_#1_bool }
97   \dim_new:c { \__enumext_store_columns_sep_#1_dim }
98   \bool_new:c { \__enumext_store_upper_level_#1_bool }
99 }
100 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__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).

```
101 \bool_new:N \l__enumext_show_answer_bool
102 \bool_new:N \l__enumext_show_position_bool
103 \tl_new:N \l__enumext_mark_ref_sym_tl
104 \tl_new:N \l__enumext_mark_answer_sym_tl
105 \str_new:N \l__enumext_mark_position_str
```

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

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

```
106 \seq_new:N \l__enumext_keyans_pic_body_seq
107 \dim_new:N \l__enumext_keyans_pic_width_dim
108 \int_new:N \l__enumext_keyans_pic_above_int
109 \int_new:N \l__enumext_keyans_pic_below_int
110 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

```
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\l__enumext_compare_items_ans_int
\g__enumext_count_item_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.

```
111 \bool_new:N \l__enumext_check_ans_bool
112 \bool_new:N \g__enumext_check_ans_show_bool
113 \bool_new:N \g__enumext_check_ans_show_h_bool
114 \tl_new:N \g__enumext_check_ans_item_tl
115 \int_new:N \l__enumext_compare_items_ans_int
116 \int_new:N \g__enumext_count_item_ans_int
117 \int_new:N \g__enumext_count_item_all_int
118 \cs_set_protected:Npn \__enumext_tmp:n #1
119 {
120   \int_new:c { g__enumext_count_level_#1_int }
121   \int_new:c { g__enumext_count_item_#1_int }
122 }
123 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

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

```
124 \bool_new:N \l__enumext_hyperref_bool
125 \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.

```
126 \tl_new:N \l__enumext_newlabel_arg_one_tl
127 \tl_new:N \l__enumext_newlabel_arg_two_tl
128 \tl_new:N \l__enumext_store_write_aux_file_tl
129 \cs_set_protected:Npn \__enumext_tmp:n #1
130 {
131   \tl_new:c { l__enumext_label_copy_#1_tl }
132 }
133 \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`.

```
134 \int_new:N \g__enumext_footnote_int
135 \seq_new:N \g__enumext_footnote_arg_seq
136 \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`.)

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

```

137 \tl_const:Nn \c__enumext_counter_style_tl
138   { { arabic } { roman } { Roman } { alph } { Alph } }
139 \tl_new:N \l__enumext_ref_key_arg_tl
140 \tl_new:N \l__enumext_ref_aux_tl
141 \cs_set_protected:Npn \__enumext_tmp:n #1
142   {
143     \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
144     \tl_new:c { l__enumext_the_counter_#1_tl }
145     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
146   }
147 \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.)

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

```

148 \cs_set_protected:Npn \__enumext_tmp:n #1
149   {
150     \bool_new:c { l__enumext_item_starred_#1_bool }
151     \int_new:c { l__enumext_item_column_pos_#1_int }
152     \int_new:c { g__enumext_item_count_all_#1_int }
153     \int_new:c { l__enumext_joined_item_#1_int }
154     \int_new:c { l__enumext_joined_item_aux_#1_int }
155     \int_new:c { l__enumext_tmpa_#1_int }
156     \box_new:c { l__enumext_item_text_#1_box }
157     \dim_new:c { l__enumext_joined_width_#1_dim }
158     \dim_new:c { l__enumext_item_width_#1_dim }
159     \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
160     \str_new:c { l__enumext_align_label_#1_str }
161     \bool_new:c { g__enumext_minipage_active_#1_bool }
162     \tl_new:c { g__enumext_miniright_code_#1_tl }
163     \bool_new:c { g__enumext_minipage_center_#1_bool }
164     \dim_new:c { g__enumext_minipage_right_#1_dim }
165     \skip_new:c { g__enumext_minipage_right_#1_skip }
166   }
167 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

An internal `clist-var` variable to run with `\__enumext_tmp:n`.

```

168 \clist_const:Nn \c__enumext_all_envs_clist
169   {
170     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
171     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
172   }

```

(End of definition for `\c__enumext_all_envs_clist`.)

## 10.5 Some utility functions

A internal “hook” function used for copying plain `list` and `minipage` environments definition and `hyperref` detection.

```

173 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
174   {
175     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
176   }

```

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

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.

```

177 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
178   {
179     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
180   }

```

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

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

```
181 \cs_new:Nn \__enumext_level:
182 {
183   \int_to_roman:n { \__enumext_level_int }
184 }
```

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

`\__enumext_level_set:n` Function for set level in `enumext*`, `keyans*` and `keyans`.

```
\__enumext_level_end:n
185 \cs_new:Npn \__enumext_level_set:n #1
186 {
187   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
188   \cs_set:Nn \__enumext_level: { #1 }
189 }
190 \cs_new:Npn \__enumext_level_end:n #1
191 {
192   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
193 }
```

(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`  
`\__enumext_if_is_int:nF` keys. This function is taken directly from the answer given by Henri Menke in [How to test if an expl3](#)  
`\__enumext_if_is_int:nTF` [function argument is an integer expression?](#).

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

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

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

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

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

## 10.6 Copying list and minipage environments

The `list` environment provided by L<sup>A</sup>T<sub>E</sub>X has the following plain form:

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

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

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

```
206 \__enumext_at_begin_document:n
207 {
208   \cs_new_eq:NN \__enumext_start_list:nn \list
209   \cs_new_eq:NN \__enumext_stop_list: \endlist
210   \cs_new_eq:NN \__enumext_item_std:w \item
211 }
```

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

The `minipage` environment provided by L<sup>A</sup>T<sub>E</sub>X 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.

```

212 \__enumext_at_begin_document:n
213 {
214     \cs_new_eq:NN \__enumext_minipage:w \minipage
215     \cs_new_eq:NN \__enumext_endminipage: \endminipage
216 }

```

(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.

```

217 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
218 \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”.

```

219 \cs_new_protected:Nn \__enumext_after_hyperref:
220 {
221     \IfPackageLoadedTF {hyperref}
222     {
223         \bool_set_true:N \l__enumext_hyperref_bool
224         \IfHyperBoolean{hyperfootnotes}
225         {
226             \typeout{hyperfootnotes=true}
227             \bool_set_true:N \l__enumext_footnotes_key_bool
228         }
229         { \typeout{hyperfootnotes=false} }
230     }
231     { }

```

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`.

```

232 \bool_if:NT \l__enumext_footnotes_key_bool
233 {
234     \IfPackageLoadedTF {footnotehyper}
235     {
236         \typeout{OK ~ hyperref ~ and ~ footnotehyper}
237     }
238     {
239         \typeout{No ~ footnotehyper ~ load}
240         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
241         \bool_set_false:N \l__enumext_footnotes_key_bool
242     }
243 }

```

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.

```

244 \bool_if:NTF \l__enumext_hyperref_bool
245 {
246     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
247     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
248 }
249 {
250     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
251     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
252 }
253 }

```

(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.

```

254 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
255 {
256   \protected@write \@auxout { }
257   {
258     \token_to_str:N \newlabel {#1}
259     {
260       {#2}
261       \bool_if:NT \l__enumext_hyperref_bool
262       { { \thepage } {#2} {#1} }
263       { }
264     }
265   }
266   \__enumext_hypertarget:nn {#1} { }
267   \__enumext_phantomsection:
268 }

```

(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.

```

269 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
270 {
271   \cs_if_exist:cTF { c@ #2 }
272   { \msg_fatal:nnn { enumext } { counters } { #2 } }
273   {
274     \tl_set:Nn #1 { #2 }
275     \newcounter { #2 }
276   }
277 }

```

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

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

```

enumXi      278 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     279 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    280 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii    281 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii   282 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXv      283 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXvi     284 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXvii    285 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

## 10.9 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```

286 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
287 {
288   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
289   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
290 }
291 \__enumext_register_counter_style:Nn \arabic { 0 }
292 \__enumext_register_counter_style:Nn \Alph { M }
293 \__enumext_register_counter_style:Nn \alph { m }
294 \__enumext_register_counter_style:Nn \Roman { VIII }
295 \__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.

```
296 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1#2
297 {
298   \hbox_set:Nn \__enumext_label_width_by_box {#2}
299   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
300 }
301 \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.

```
302 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
303 {
304   \tl_clear_new:N #1
305   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
306   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
307   \tl_map_inline:Nn \g__enumext_counter_styles_tl
308   {
309     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
310     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
311     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
312   }
313   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
314   { \tl_use:N \g__enumext_widest_label_tl }
315   \tl_set_eq:cN { the #2 } #1
316 }
317 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

## 10.10 Setting keys associated with label

`font`  
`labelsep`  
`labelwidth`  
`wrap-label`  
`wrap-label*`

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

```
318 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
319 {
320   \keys_define:nn { enumext / #1 }
321   {
322     font      .tl_set:c    = { l__enumext_label_font_style_#2_tl },
323     font      .value_required:n = true,
324     labelsep   .dim_set:c   = { l__enumext_labelsep_#2_dim },
325     labelsep   .initial:n    = {0.3333em},
326     labelsep   .value_required:n = true,
327     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
328     labelwidth .value_required:n = true,
329     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
330     wrap-label .initial:n    = {##1},
331     wrap-label .value_required:n = true,
332     wrap-label* .code:n = {
333       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
334       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
335     },
336     wrap-label* .value_required:n = true,
337   }
338 }
339 \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.

```

340 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
341 {
342   \keys_define:nn { enumext / #1 }
343   {
344     align .choice:,
345     align / left .code:n =
346       {
347         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
348         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
349       },
350     align / right .code:n =
351       {
352         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
353         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
354       },
355     align / center .code:n =
356       {
357         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
358         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
359       },
360     align .initial:n = left,
361     align .value_required:n = true,
362   }
363 }
364 \clist_map_inline:nn
365 {
366   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
367 }
368 { \__enumext_tmp:nn #1 }

```

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

```

369 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
370 {
371   \keys_define:nn { enumext / #1 }
372   {
373     align .choice:,
374     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
375     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
376     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
377     align .initial:n = left,
378     align .value_required:n = true,
379   }
380 }
381 \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.

```

382 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
383 {
384   \tl_map_inline:Nn \c__enumext_counter_style_tl
385   {
386     \regex_replace_once:nnN { \c{##1}\* }
387     { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
388   }
389 }

```

(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`.

```

390 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
391 {
392   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
393   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_counter_ \l__enumext_level: _tl }
394   \__enumext_regex_label_ref_key:
395   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_the_counter_ \l__enumext_level: _tl }
396   \tl_put_right:ce { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
397   {
398     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
399     { \exp_not:V \l__enumext_ref_key_arg_tl }
400   }
401 }

```

(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`.

```

402 \cs_new_protected:Nn \__enumext_use_key_ref:
403 {
404   \tl_if_empty:cF { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
405   {
406     \tl_use:c { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
407   }
408 }

```

(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.

```

409 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
410 {
411   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
412   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
413   {
414     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
415     \__enumext_regex_label_ref_key:
416     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
417     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
418     {
419       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
420       { \exp_not:V \l__enumext_ref_key_arg_tl }
421     }
422   }
423   {
424     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
425     \__enumext_regex_label_ref_key:
426     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
427     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
428     {
429       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
430       { \exp_not:V \l__enumext_ref_key_arg_tl }
431     }
432   }
433 }

```

(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*`.

```

434 \cs_new_protected:Nn \__enumext_use_key_ref_h:
435 {
436   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
437   {

```

```

438         \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
439         {
440             \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
441         }
442     }
443 {
444     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
445     {
446         \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
447     }
448 }
449 }

```

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

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

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

```

\__enumext_label_i_tl 450 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_ii_tl 451 {
\__enumext_label_iii_tl 452     \keys_define:nn { enumext / #1 }
\__enumext_label_iv_tl 453     {
454         label .code:n = {
455             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
456             { \__enumext_counter_#2_tl } {##1}
457             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
458             \l__enumext_current_widest_dim
459         },
460         label .initial:n = #3,
461         label .value_required:n = true,
462         ref .code:n = \__enumext_set_label_ref:n {##1},
463         ref .value_required:n = true,
464     }
465 }
466 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
467 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
468 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
469 \__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 *labels* for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key.

```

\__enumext_label_vii_tl 470 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_viii_tl 471 {
472     \keys_define:nn { enumext / #1 }
473     {
474         label .code:n = {
475             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
476             { \__enumext_counter_#2_tl } {##1}
477             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
478             \l__enumext_current_widest_dim
479         },
480         label .initial:n = #3,
481         label .value_required:n = true,
482         ref .code:n = \__enumext_set_label_ref_h:n {##1},
483         ref .value_required:n = true,
484     }
485 }
486 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
487 \__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 *label* for `keyans` and `keyanspic` environment, along with the default value for `labelwidth`. The `keyanspic` environment use the same *label* as the `keyans` environment.  
Define and set `label` key for `keyans` environment.

```

488 \keys_define:nn { enumext / keyans }
489 {

```



```

490     label .code:n = {
491         \__enumext_label_style:cvn { l__enumext_label_v_tl }
492         { l__enumext_counter_v_tl } {#1}
493         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
494         \l__enumext_current_widest_dim
495         \__enumext_label_style:cvn { l__enumext_label_vi_tl }
496         { l__enumext_counter_vi_tl } {#1}
497         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
498         \l__enumext_current_widest_dim
499     },
500     label .initial:n = (\Alph*),
501     label .value_required:n = true,
502 }

```

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

## 10.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

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

```

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

```

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

```

503 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
504 {
505     \__enumext_if_is_int:nTF { #3 }
506     {
507         \int_set:Nn #2 {#3}
508     }
509     {
510         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
511         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
512         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
513         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
514     }
515 }
516 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

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

```

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

```

517 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
518 {
519     \__enumext_if_is_int:nTF {#4}
520     {
521         \setcounter{enumX#1} { #4 }
522     }
523     {
524         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
525         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
526         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
527         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
528     }
529     \__enumext_label_width_by_box:cv
530     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
531 }
532 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

```

start
widest
\l__enumext_start_X_int
533 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
534 {
535   \keys_define:nn { enumext / #1 }
536   {
537     start .code:n = {
538       \__enumext_start_from:ccn
539       { l__enumext_label_#2_tl }
540       { l__enumext_start_#2_int } {##1}
541     },
542     start .initial:n = 1,
543     widest .code:n = {
544       \__enumext_widest_from:nccn {#2}
545       { l__enumext_label_#2_tl }
546       { l__enumext_labelwidth_#2_dim } {##1}
547     },
548     widest .value_required:n = true,
549     start .value_required:n = true,
550   }
551 }
552 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 10.13 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
553 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
554 {
555   \keys_define:nn { enumext / #1 }
556   {
557     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
558     topsep .initial:n = {#3},
559     topsep .value_required:n = true,
560     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
561     partopsep .initial:n = {#4},
562     partopsep .value_required:n = true,
563     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
564     parsep .initial:n = {#5},
565     parsep .value_required:n = true,
566     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
567     itemsep .initial:n = {#6},
568     itemsep .value_required:n = true,
569     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
570     noitemsep .value_forbidden:n = true,
571     nosep .meta:n = {
572       itemsep = 0pt, parsep = 0pt,
573       topsep = 0pt, partopsep = 0pt,
574     },
575     nosep .value_forbidden:n = true,
576   }
577 }

```

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

```

578 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
579 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
580 { 4.0pt plus 2.0pt minus 1.0pt }
581 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
582 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
583 { 2.0pt plus 1.0pt minus 1.0pt }
584 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
585 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
586 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
587 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
588 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
589 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
590 { 2.0pt plus 1.0pt minus 1.0pt }
591 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
592 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
593 { 4.0pt plus 2.0pt minus 1.0pt }

```

```

594 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
595 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
596 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

## 10.14 Setting keys for horizontal spaces

```

itemindent
rightmargin
listparindent
list-offset
list-indent

```

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

```

597 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
598 {
599   \keys_define:nn { enumext / #1 }
600   {
601     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
602     itemindent .value_required:n = true,
603     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
604     rightmargin .value_required:n = true,
605     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
606     listparindent .value_required:n = true,
607     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
608     list-offset .value_required:n = true,
609     list-indent .code:n =
610       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
611       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
612     list-indent .value_required:n = true,
613   }
614 }
615 \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.

```

616 \cs_set_protected:Npn \__enumext_tmp:n #1
617 {
618   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
619 }
620 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

### 10.14.1 Functions for setting the fake itemindent

```

\__enumext_fake_item:
\__enumext_keyans_fake_item:
\__enumext_fake_item_vii:
\__enumext_fake_item_viii:

```

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

```

621 \cs_set_protected:Nn \__enumext_fake_item:
622 {
623   \dim_compare:nNnT
624     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
625     >
626     { \c_zero_dim }
627   {
628     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
629     {
630       \exp_not:N \mode_leave_vertical:
631       \exp_not:n { \skip_horizontal:n }
632       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
633       \ignorespaces
634     }
635   }
636 }
637 \cs_set_protected:Nn \__enumext_keyans_fake_item:
638 {
639   \dim_compare:nNnT
640     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
641     {
642       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
643       {
644         \exp_not:N \mode_leave_vertical:
645         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
646       }
647     }

```

```

648     }
649     \cs_set_protected:Nn \__enumext_fake_item_vii:
650     {
651         \dim_compare:nNnT
652         { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
653         {
654             \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
655             {
656                 \exp_not:N \mode_leave_vertical:
657                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
658             }
659         }
660     }
661     \cs_set_protected:Nn \__enumext_fake_item_viii:
662     {
663         \dim_compare:nNnT
664         { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
665         {
666             \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
667             {
668                 \exp_not:N \mode_leave_vertical:
669                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
670             }
671         }
672     }

```

(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.

```

673 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
674 {
675     \keys_define:nn { enumext / #1 }
676     {
677         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
678         show-length .initial:n = false,
679     }
680 }
681 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 10.16 Setting before, after and first keys

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

```

before* 682 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    683 {
first    684     \keys_define:nn { enumext / #1 }
        685     {
        686         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
        687         before .value_required:n = true,
        688         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
        689         before* .value_required:n = true,
        690         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
        691         after .value_required:n = true,
        692         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
        693         first .value_required:n = true,
        694     }
        695 }
        696 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

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

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

```

697 \cs_new_protected:Nn \__enumext_before_args_exec:

```

```

698   {
699       \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
700   }

```

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⟩`.

```

701 \cs_new_protected:Nn \__enumext_before_keys_exec:
702   {
703       \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
704   }

```

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

```

705 \cs_new_protected:Nn \__enumext_after_stop_list:
706   {
707       \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
708   }

```

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`.

```

709 \cs_new_protected:Nn \__enumext_after_args_exec:
710   {
711       \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
712   }

```

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

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:

```

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

```

713 \cs_new_protected:Nn \__enumext_before_args_exec_v:
714   {
715       \tl_use:N \l__enumext_before_starred_key_v_tl
716   }

```

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⟩`.

```

717 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
718   {
719       \tl_use:N \l__enumext_before_no_starred_key_v_tl
720   }

```

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

```

721 \cs_new_protected:Nn \__enumext_after_stop_list_v:
722   {
723       \tl_use:N \l__enumext_after_stop_list_v_tl
724   }

```

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`.

```

725 \cs_new_protected:Nn \__enumext_after_args_exec_v:
726   {
727       \tl_use:N \l__enumext_after_list_args_v_tl
728   }

```

(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.

```

729 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
730   {
731       \tl_use:N \l__enumext_before_starred_key_vii_tl
732   }
733 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
734   {
735       \tl_use:N \l__enumext_before_starred_key_viii_tl
736   }

```

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`.

```

737 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
738 {
739     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
740 }
741 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
742 {
743     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
744 }

```

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

```

745 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
746 {
747     \tl_use:N \l__enumext_after_stop_list_vii_tl
748 }
749 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
750 {
751     \tl_use:N \l__enumext_after_stop_list_viii_tl
752 }

```

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`.

```

753 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
754 {
755     \tl_use:N \l__enumext_after_list_args_vii_tl
756 }
757 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
758 {
759     \tl_use:N \l__enumext_after_list_args_viii_tl
760 }

```

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

`columns-sep`

`columns`

```

761 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
762 {
763     \keys_define:nn { enumext / #1 }
764     {
765         mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
766         mini-env .value_required:n = true,
767         mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
768         mini-sep .initial:n = 0.3333em,
769         mini-sep .value_required:n = true,
770         columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
771         columns-sep .value_required:n = true,
772         columns .int_set:c = { \l__enumext_columns_#2_int },
773         columns .initial:n = 1,
774         columns .value_required:n = true,
775     }
776 }
777 \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`.

```

778 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
779 {
780     \keys_define:nn { enumext / #1 }
781     {
782         columns .initial:n = 2,
783         miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
784         miniright .value_required:n = true,
785         miniright* .code:n = {

```

```

786             \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
787             \keys_set:nn { enumext / #1 } { miniright = {##1} }
788             },
789     miniright* .value_required:n = true,
790 }
791 }
792 \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.

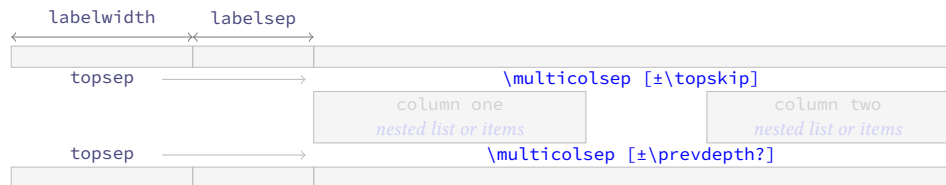


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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 multicols in enumext

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

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

```

793 \cs_new_protected:Nn \__enumext_multi_set_vskip:
794 {
795     \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
796     {
797         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
798     }
799     \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
800     {
801         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
802     }
803     \__enumext_add_pre_parsep:
804 }

```

(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*.

```

805 \cs_new_protected:Nn \__enumext_add_pre_parsep:
806 {
807     \int_case:nn { \l__enumext_level_int }
808     {
809         { 2 }{
810             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
811             {
812                 \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
813             }
814         }
815         { 3 }{
816             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
817             {

```



```

818             \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_iii_skip
819         }
820     }
821     { 4 }{
822         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
823         {
824             \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
825         }
826     }
827 }
828 }

```

(End of definition for `\__enumext_add_pre_parsep:`)

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

```

829 \cs_new_protected:Nn \__enumext_multi_addvspace:
830 {
831     \__enumext_multi_set_vskip:
832     \mode_if_vertical:T
833     {
834         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
835         {
836             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
837         }
838         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
839         {
840             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
841         }
842     }
843     \par\nopagebreak
844     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
845 }

```

(End of definition for `\__enumext_multi_addvspace:`)

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

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

`\__enumext_keyans_multi_addvspace:`

```

846 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
847 {
848     \skip_set:Nn \l__enumext_multicols_above_v_skip
849     {
850         \l__enumext_topsep_v_skip
851     }
852     \skip_set:Nn \l__enumext_multicols_below_v_skip
853     {
854         \l__enumext_topsep_v_skip
855     }
856 }
857 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
858 {
859     \__enumext_keyans_multi_set_vskip:
860     \mode_if_vertical:T
861     {
862         \skip_add:Nn \l__enumext_multicols_above_v_skip
863         {
864             \skip_use:N \l__enumext_partopsep_v_skip
865         }
866         \skip_add:Nn \l__enumext_multicols_below_v_skip
867         {
868             \skip_use:N \l__enumext_partopsep_v_skip
869         }
870     }
871     \par\nopagebreak
872     \addvspace{ \l__enumext_multicols_above_v_skip }
873 }

```

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

## 10.19 Adjustment of vertical spaces for minipage

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



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

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

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

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

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

```

874 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
875 {
876   \__enumext_minipage:w [ t ] { #1 }
877   \legacy_if_gset_false:n { @minipage }
878   \vspace { 0pt }
879 }
880 { \__enumext_endminipage: }
```

(End of definition for `__enumext_mini_env*`.)

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

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

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

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

```

881 \cs_new_protected:Nn \__enumext_mini_set_vskip:
882 {
883   \int_compare:nNnTF
884     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
885     {
```

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.

```

886   \skip_if_eq:nnTF
887     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
888     {
889       \skip_set:Nn \l__enumext_minipage_left_skip
890         {
891           -0.150\box_dp:N \strutbox
892         }
893       \skip_set:Nn \l__enumext_minipage_right_skip
894         {
895           0.695\box_dp:N \strutbox
896         }

```

```

897         \skip_set:Nn \l__enumext_minipage_after_skip
898         {
899             \box_dp:N \strutbox
900         }
901     \__enumext_zero_parsep:
902 }
903 {
904     \skip_set:Nn \l__enumext_minipage_left_skip
905     {
906         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
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         1.85\box_dp:N \strutbox
915         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
916     }
917 }
918 }
919 {

```

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.

```

920     \skip_if_eq:nnTF
921     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
922     {
923         \skip_set:Nn \l__enumext_minipage_left_skip
924         {
925             0.5\box_dp:N \strutbox
926             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
927         }
928         \skip_set:Nn \l__enumext_minipage_right_skip
929         {
930             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
931         }
932         \skip_set:Nn \l__enumext_minipage_after_skip
933         {
934             1.6\box_dp:N \strutbox
935         }
936     }
937     {
938         \skip_set:Nn \l__enumext_minipage_left_skip
939         {
940             0.5875\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_topsep_ \__enumext_level: _skip }
946             + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
947         }
948         \skip_set:Nn \l__enumext_minipage_after_skip
949         {
950             0.325\box_dp:N \strutbox
951             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
952         }
953     }
954 }
955 }

```

(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.

```

956 \cs_new_protected:Nn \__enumext_zero_parsep:
957 {

```

```

958 \int_case:nn { \l__enumext_level_int }
959 {
960   { 2 }{
961     \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
962     {
963       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
964     }
965   }
966   { 3 }{
967     \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
968     {
969       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
970     }
971   }
972   { 4 }{
973     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
974     {
975       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
976     }
977   }
978 }
979 }

```

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

`\__enumext_mini_addvspace:` The function `\__enumext_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\__enumext_mini_env*` environment in `enumext`, taking into account whether  $\text{\TeX}$  is in *horizontal mode* or *vertical mode*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

980 \cs_new_protected:Nn \__enumext_mini_addvspace:
981 {
982   \__enumext_mini_set_vskip:
983   \mode_if_vertical:T
984   {
985     \skip_add:Nn \l__enumext_minipage_left_skip
986     {
987       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
988     }
989     \skip_add:Nn \l__enumext_minipage_after_skip
990     {
991       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
992     }
993   }
994   \par\nopagebreak
995   \addvspace { \l__enumext_minipage_left_skip }
996 }

```

(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`.

```

997 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
998 {
999   \skip_zero_new:N \l__enumext_minipage_after_skip
1000   \skip_zero_new:N \l__enumext_minipage_left_skip
1001   \skip_zero_new:N \l__enumext_minipage_right_skip
1002   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1003   {
1004     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1005     {
1006       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1007       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1008       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1009       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1010       {
1011         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1012       }
1013     }
1014   }

```

```

1014     {
1015         \skip_set:Nn \l__enumext_minipage_left_skip
1016         {
1017             \skip_use:N \l__enumext_topsep_v_skip
1018         }
1019         \skip_set:Nn \l__enumext_minipage_right_skip
1020         {
1021             0.705\box_dp:N \strutbox
1022         }
1023         \skip_set:Nn \l__enumext_minipage_after_skip
1024         {
1025             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1026         }
1027     }
1028 }
1029 {
1030     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1031     {
1032         \skip_set:Nn \l__enumext_minipage_left_skip
1033         {
1034             0.5\box_dp:N \strutbox
1035             + \l__enumext_partopsep_v_skip
1036         }
1037         \skip_set:Nn \l__enumext_minipage_right_skip
1038         {
1039             \l__enumext_partopsep_v_skip
1040         }
1041         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1042     }
1043     {
1044         \skip_set:Nn \l__enumext_minipage_left_skip
1045         {
1046             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1047         }
1048         \skip_set:Nn \l__enumext_minipage_right_skip
1049         {
1050             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1051         }
1052         \skip_set:Nn \l__enumext_minipage_after_skip
1053         {
1054             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1055         }
1056     }
1057 }
1058 }

```

(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  $\langle horizontal mode \rangle$  or  $\langle vertical mode \rangle$ . 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`.

```

1059 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1060 {
1061     \__enumext_keyans_mini_set_vskip:
1062     \mode_if_vertical:T
1063     {
1064         \skip_add:Nn \l__enumext_minipage_left_skip
1065         {
1066             \l__enumext_partopsep_v_skip
1067         }
1068         \skip_add:Nn \l__enumext_minipage_after_skip
1069         {
1070             \l__enumext_partopsep_v_skip
1071         }
1072     }
1073     \par\nopagebreak
1074     \addvspace { \l__enumext_minipage_left_skip }
1075 }

```

(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*`.

```

1076 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1077 {
1078   \skip_zero_new:N \l__enumext_minipage_left_skip
1079   \skip_gzero_new:N \g__enumext_minipage_right_skip
1080   \skip_gzero_new:N \g__enumext_minipage_after_skip
1081   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1082   {
1083     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1084     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1085   }
1086   {
1087     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1088     \skip_gset:Nn \g__enumext_minipage_right_skip
1089     {
1090       \l__enumext_topsep_vii_skip
1091     }
1092     \skip_gset:Nn \g__enumext_minipage_after_skip
1093     {
1094       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1095     }
1096   }
1097 }
1098 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1099 {
1100   \skip_zero_new:N \l__enumext_minipage_after_skip
1101   \skip_zero_new:N \l__enumext_minipage_left_skip
1102   \skip_zero_new:N \l__enumext_minipage_right_skip
1103   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1104   {
1105     \skip_set:Nn \l__enumext_minipage_left_skip
1106     {
1107       0.5\box_dp:N \strutbox
1108     }
1109     \skip_set:Nn \l__enumext_minipage_right_skip
1110     {
1111       \l__enumext_partopsep_viii_skip
1112     }
1113     \skip_set:Nn \l__enumext_minipage_after_skip
1114     {
1115       1.6\box_dp:N \strutbox
1116     }
1117   }
1118   {
1119     \skip_set:Nn \l__enumext_minipage_left_skip
1120     {
1121       0.5875\box_dp:N \strutbox
1122     }
1123     \skip_set:Nn \l__enumext_minipage_right_skip
1124     {
1125       \l__enumext_topsep_viii_skip
1126     }
1127     \skip_set:Nn \l__enumext_minipage_after_skip
1128     {
1129       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1130     }
1131   }
1132 }

```

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

`\__enumext_mini_addvspace_vii:`  
`\__enumext_mini_addvspace_viii:`

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

Here we will NOT take into account whether T<sub>E</sub>X is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to 0pt in both environments.

```

1133 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:

```

```

1134 {
1135   \__enumext_mini_set_vskip_vii:
1136   \par\nopagebreak
1137   \addvspace { \__enumext_minipage_left_skip }
1138 }
1139 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1140 {
1141   \__enumext_mini_set_vskip_viii:
1142   \par\nopagebreak
1143   \addvspace { \__enumext_minipage_left_skip }
1144 }

```

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

#### 10.19.4 The command \miniright

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

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

```

1145 \NewDocumentCommand \miniright { s }
1146 {
1147   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1148   {
1149     \msg_error:nnn { enumext } { wrong-miniright-place }
1150   }
1151   \int_compare:nNnT { \__enumext_level_int } = { 0 }
1152   {
1153     \msg_error:nnn { enumext } { wrong-miniright-place }
1154   }
1155   \int_compare:nNnTF { \__enumext_keyans_level_int } = { 1 }
1156   {
1157     \__enumext_keyans_mini_right_cmd:n {#1}
1158   }
1159   { \__enumext_mini_right_cmd:n {#1} }
1160 }

```

(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.

```

1161 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1162 {
1163   \dim_compare:nNnTF
1164   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1165   {
1166     \__enumext_multicols_stop:
1167     \end{\__enumext_mini_env*}
1168     \hfill
1169     \begin{\__enumext_mini_env*}
1170     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1171     \par\addvspace { \__enumext_minipage_right_skip }
1172     \bool_if:nF {#1}
1173     {
1174       \centering
1175     }
1176     \int_gzero:N \g__enumext_minipage_stat_int
1177   }
1178   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1179 }

```



(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 `\mini_right` 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.

```

1180 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1181 {
1182   \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1183   {
1184     \__enumext_keyans_multicols_stop:
1185     \end{__enumext_mini_env*}
1186     \hfill
1187     \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1188     \par\addvspace { \__enumext_minipage_right_skip }
1189     \bool_if:nF {#1}
1190     {
1191       \centering
1192     }
1193     \int_gzero:N \g__enumext_minipage_stat_int
1194   }
1195   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1196 }

```

(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.

```

1197 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1198 {
1199   \keys_define:nn { enumext / #1 }
1200   {
1201     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1202     above .value_required:n = true,
1203     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1204                       \keys_set:nn { enumext / #1 } { above = {##1} },
1205     above* .value_required:n = true,
1206     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1207     below .value_required:n = true,
1208     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1209                       \keys_set:nn { enumext / #1 } { below = {##1} },
1210     below* .value_required:n = true,
1211   }
1212 }
1213 \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.

```

1214 \cs_new_protected:Nn \__enumext_vspace_above:
1215 {
1216   \skip_if_eq:nnF
1217   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1218   {
1219     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1220     {
1221       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1222     }
1223     {
1224       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1225     }
1226   }
1227 }

```

(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.

```

1228 \cs_new_protected:Nn \__enumext_vspace_below:
1229 {
1230   \skip_if_eq:nnF
1231     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1232   {
1233     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1234     {
1235       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1236     }
1237     {
1238       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1239     }
1240   }
1241 }

```

(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.

```

1242 \cs_new_protected:Nn \__enumext_vspace_above_v:
1243 {
1244   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1245   {
1246     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1247     {
1248       \vspace*{ \__enumext_vspace_above_v_skip }
1249     }
1250     { \vspace { \__enumext_vspace_above_v_skip } }
1251   }
1252 }

```

(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.

```

1253 \cs_new_protected:Nn \__enumext_vspace_below_v:
1254 {
1255   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1256   {
1257     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1258     {
1259       \vspace*{ \__enumext_vspace_below_v_skip }
1260     }
1261     { \vspace { \__enumext_vspace_below_v_skip } }
1262   }
1263 }

```

(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:`

```

1264 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1265 {
1266   \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1267   {
1268     \bool_if:NTF \__enumext_vspace_a_star_vii_bool
1269     {
1270       \vspace*{ \__enumext_vspace_above_vii_skip }
1271     }
1272     { \vspace { \__enumext_vspace_above_vii_skip } }
1273   }
1274 }
1275 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1276 {
1277   \skip_if_eq:nnF { \__enumext_vspace_above_viii_skip } { \c_zero_skip }

```

```

1278     {
1279         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1280         {
1281             \vspace*{ \l__enumext_vspace_above_viii_skip }
1282         }
1283         { \vspace { \l__enumext_vspace_above_viii_skip } }
1284     }
1285 }

```

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

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

```

1286 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1287 {
1288     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1289     {
1290         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1291         {
1292             \vspace*{ \l__enumext_vspace_below_vii_skip }
1293         }
1294         { \vspace { \l__enumext_vspace_below_vii_skip } }
1295     }
1296 }
1297 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1298 {
1299     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1300     {
1301         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1302         {
1303             \vspace*{ \l__enumext_vspace_below_viii_skip }
1304         }
1305         { \vspace { \l__enumext_vspace_below_viii_skip } }
1306     }
1307 }

```

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

## 10.21 Setting save-ans and resume keys

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

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

```

save-ans
resume
resume*
1308 \keys_define:nn { enumext / level-1 }
1309 {
1310     save-ans .code:n = \l__enumext_storing_set:n {#1},
1311     save-ans .value_required:n = true,
1312     resume .code:n = \l__enumext_resume_counter:,
1313     resume .value_forbidden:n = true,
1314     resume* .code:n = \l__enumext_resume_counter_star:,
1315     resume* .value_forbidden:n = true,
1316 }
1317 \keys_define:nn { enumext / enumext* }
1318 {
1319     save-ans .code:n = \l__enumext_storing_set:n {#1},
1320     save-ans .value_required:n = true,
1321     resume .code:n = \l__enumext_resume_counter_vii:,
1322     resume .value_forbidden:n = true,
1323 }

```

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

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

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

```

1324 \cs_new_protected:Npn \__enumext_storing_set:n #1
1325 {
1326   \tl_set:Nx \l__enumext_store_name_tl {#1}
1327   \bool_set_true:N \l__enumext_store_active_bool
1328   \int_if_exist:cF { g__enumext_resume_#1_int }
1329   {
1330     \int_new:c { g__enumext_resume_#1_int }
1331   }
1332   \bool_if:NT \l__enumext_check_ans_bool
1333   {
1334     \__enumext_check_ans_int:n {#1}
1335   }
1336 }

```

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

```

\__enumext_resume_counter:
\__enumext_resume_counter_vii:

```

The functions \\_\_enumext\_resume\_counter: and \\_\_enumext\_resume\_counter\_vii: used by resume key in enumext and enumext\*. If save-ans key present then set the start value from integer created by \\_\_enumext\_storing\_set:n.

```

1337 \cs_new_protected:Nn \__enumext_resume_counter:
1338 {
1339   \bool_if:NT \l__enumext_store_active_bool
1340   {
1341     \int_gset:Nn \g__enumext_resume_int
1342     {
1343       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1344     }
1345   }
1346   \bool_set_true:N \l__enumext_resume_bool
1347 }
1348 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1349 {
1350   \bool_if:NT \l__enumext_store_active_bool
1351   {
1352     \int_gset:Nn \g__enumext_resume_int
1353     {
1354       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1355     }
1356   }
1357   \bool_set_true:N \l__enumext_resume_vii_bool
1358 }

```

(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 starts with an \item or \item\* and does not open a nested environment it must have a \anskey, if the line starts with an \item or \item\* and opens a *nested environment*, each \item or \item\* in the *nested* environment must have a “once” \anskey.

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

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

Each \item and \item\* in the environment must be matched with the counter associated with \anskey (\g\_\_enumext\_count\_item\_ans\_int). 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 1359 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1360 {
1361   \keys_define:nn { enumext / #1 }
1362   {
1363     check-ans .bool_set:N = \__enumext_check_ans_bool,
1364     check-ans .initial:n = false,
1365     no-store .bool_set:N = \__enumext_store_ans_bool,
1366     no-store .initial:n = false,
1367   }
1368 }
1369 \clist_map_inline:nn
1370 {
1371   level-1, level-2, level-3, level-4, enumext*
1372 }
1373 { \__enumext_tmp:n {#1} }
```

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

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`

```

1374 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1375 {
1376   \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1377   { \int_new:c { g__enumext_count_#1_item_ans_int } }
1378   \int_if_exist:cF { g__enumext_count_#1_i_int }
1379   { \int_new:c { g__enumext_count_#1_i_int } }
1380   \int_if_exist:cF { g__enumext_count_#1_ii_int }
1381   { \int_new:c { g__enumext_count_#1_ii_int } }
1382   \int_if_exist:cF { g__enumext_count_#1_iii_int }
1383   { \int_new:c { g__enumext_count_#1_iii_int } }
1384   \int_if_exist:cF { g__enumext_count_#1_iv_int }
1385   { \int_new:c { g__enumext_count_#1_iv_int } }
1386   \int_if_exist:cF { g__enumext_count_#1_vii_int }
1387   { \int_new:c { g__enumext_count_#1_vii_int } }
```

We make `\g__enumext_count_item_X_int` equal to the integer variable that contains all the occurrences of `\item` and `\item*` in the different levels and we will make `\g__enumext_count_item_ans_int` equal to the integer variable handled by the `\anskey` command.

```

1388   \bool_lazy_all:nTF
1389   {
1390     { \g__enumext_starred_bool }
1391     { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
1392   }
1393   {
1394     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1395   }
1396   {
1397     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1398   }
1399   \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1400   \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1401   \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1402   \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1403   \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1404   \int_gset_eq:Nc \g__enumext_count_item_ans_int { g__enumext_count_#1_item_ans_int }
1405 }
```

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

### 10.22.2 Set-up check answer mechanism

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.

```

1406 \cs_new_protected:Nn \__enumext_check_ans_count:
1407 {
1408   \bool_if:NT \l__enumext_check_ans_bool
```

```

1409     {
1410         \bool_if:NTF \l__enumext_store_ans_bool
1411         {
1412             \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1413             { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1414         }
1415         { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1416     }
1417 }

```

(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:

```

1418 \cs_new_protected:Nn \__enumext_check_ans_active:
1419 {
1420     \int_set:Nn \l__enumext_compare_items_ans_int
1421     {
1422         \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1423         - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1424     }
1425     \int_compare:nNnTF
1426     { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_ans_int }
1427     {
1428         \msg_term:nnV { enumext } { items-same-answer }
1429         \g__enumext_store_name_tl
1430     }
1431     {
1432         \msg_warning:nnV { enumext } { item-different-answer }
1433         \g__enumext_store_name_tl
1434     }
1435 }

```

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

```

1435     \int_gzero:N \g__enumext_count_level_i_int
1436     \int_gzero:N \g__enumext_count_level_ii_int
1437     \int_gzero:N \g__enumext_count_level_iii_int
1438     \int_gzero:N \g__enumext_count_level_iv_int
1439     \int_gzero:N \g__enumext_count_level_vii_int
1440 }

1441 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1442 {
1443     \int_set:Nn \l__enumext_compare_items_ans_int
1444     {
1445         \g__enumext_count_item_all_int
1446         - \g__enumext_count_item_i_int
1447         - \g__enumext_count_item_ii_int
1448         - \g__enumext_count_item_iii_int
1449         - \g__enumext_count_item_iv_int
1450     }
1451     \int_compare:nNnTF
1452     { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_ans_int }
1453     {
1454         \msg_term:nnV { enumext } { items-same-answer }
1455         \g__enumext_store_name_tl
1456     }
1457     {
1458         \msg_warning:nnV { enumext } { item-different-answer }
1459         \g__enumext_store_name_tl
1460     }
1461     \int_gzero:N \g__enumext_count_level_i_int
1462     \int_gzero:N \g__enumext_count_level_ii_int
1463     \int_gzero:N \g__enumext_count_level_iii_int
1464     \int_gzero:N \g__enumext_count_level_iv_int
1465     \int_gzero:N \g__enumext_count_level_vii_int
1466 }

```

(End of definition for \\_\_enumext\_check\_ans\_active: and \\_\_enumext\_check\_ans\_active\_vii:.)

## 10.23 Keys and functions associated with storage

wrap-ans 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*`.

```

1467 \cs_set_protected:Npn \__enumext_tmp:n #1
1468 {
1469   \keys_define:nn { enumext / #1 }
1470   {
1471     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1472     wrap-ans .initial:n = \fbox{##1},
1473     wrap-ans .value_required:n = true,
1474     mark-ans .code:n = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1475     mark-ans .initial:n = \textasteriskcentered,
1476     mark-ans .value_required:n = true,
1477     mark-pos .choice:,
1478     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1479     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1480     mark-pos .initial:n = right,
1481     mark-pos .value_required:n = true,
1482     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1483               \bool_set_false:N \l__enumext_show_position_bool,
1484     show-ans .value_forbidden:n = true,
1485     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1486               \bool_set_false:N \l__enumext_show_answer_bool,
1487     show-pos .value_forbidden:n = true,
1488     mark-ref .code:n = \tl_set:Nn \l__enumext_mark_ref_sym_tl {##1},
1489     mark-ref .initial:n = \textasteriskcentered,
1490     mark-ref .value_required:n = true,
1491     store-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1492     store-ref .initial:n = false,
1493   }
1494 }
1495 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1496 \cs_set_protected:Npn \__enumext_tmp:n #1
1497 {
1498   \keys_define:nn { enumext / #1 }
1499   {
1500     mark-pos .choice:,
1501     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1502     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1503     mark-pos .initial:n = right,
1504     mark-pos .value_required:n = true,
1505     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1506               \bool_set_false:N \l__enumext_show_position_bool,
1507     show-ans .value_forbidden:n = true,
1508     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1509               \bool_set_false:N \l__enumext_show_answer_bool,
1510     show-pos .value_forbidden:n = true,
1511   }
1512 }
1513 \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.

```

1514 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1515 {
1516   \keys_define:nn { enumext / #1 }
1517   {
1518     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1519               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1520               \tl_put_right:ce { l__enumext_store_opt_#2_tl }

```



```

1521         {
1522             columns = \exp_not:v { l__enumext_store_columns_#2_int },
1523         },,
1524         columns* .value_required:n = true,
1525         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1526             \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1527             \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1528             {
1529                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1530             },
1531         columns-sep* .value_required:n = true,
1532     }
1533 }
1534 \clist_map_inline:nn
1535 {
1536     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1537 }
1538 { \__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, if it does not exist it will create it globally. The “*stored content*” is retrieved by means of the `\getkeyans` command.

The form in which the content is “*stored*” in the *⟨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.

```

1539 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1540 {
1541     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1542     { \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1543     \prop_gput:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1544     {
1545         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } +1 }
1546     } { #1 }
1547 }
1548 \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, if it does not exist it will create it globally. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`. The form in which the content is stored in *⟨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.

```

1549 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1550 {
1551     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1552     { \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq } }
1553     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1554 }
1555 \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 `opt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```

1556 \cs_new_protected:Nn \__enumext_store_level_open:
1557 {
1558     \bool_if:NF \l__enumext_store_ans_bool
1559     {

```

```

1560 \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1561 {
1562     \__enumext_store_addto_seq:n
1563     {
1564         \item \begin{enumext}
1565     }
1566 }
1567 {
1568     \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1569     {
1570         \item \begin{enumext} [
1571     ]
1572     \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1573     {
1574     ]
1575     }
1576     \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1577 }
1578 }
1579 }
1580 \cs_new_protected:Nn \__enumext_store_level_close:
1581 {
1582     \bool_if:NF \l__enumext_store_ans_bool
1583     {
1584         \__enumext_store_addto_seq:n { \end{enumext} }
1585     }
1586 }

```

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

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

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

```

1587 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1588 {
1589     \bool_if:NF \l__enumext_store_ans_bool
1590     {
1591         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1592         {
1593             \__enumext_store_addto_seq:n
1594             {
1595                 \item \mode_leave_vertical:
1596                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1597                 \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1598             }
1599         }
1600         {
1601             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1602             {
1603                 \item \mode_leave_vertical:
1604                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1605                 \begin{enumext*}[before={\setlength{\topsep}{0pt}},
1606             ]
1607             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1608             {
1609             ]
1610             }
1611             \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1612         }
1613     }
1614 }
1615 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1616 {
1617     \bool_if:NF \l__enumext_store_ans_bool
1618     {
1619         \__enumext_store_addto_seq:n { \end{enumext*} }
1620     }
1621 }

```

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

#### 10.23.4 Function for show marks and position

`\__enumext_print_keyans_box:NN`  
`\__enumext_print_keyans_box:cc`

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

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

1622 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1623 {
1624   \mode_leave_vertical:
1625   \skip_horizontal:n { -\dim_use:N #2 }
1626   \makebox[0pt][ r ]
1627   {
1628     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1629     {
1630       \tl_use:N \l__enumext_mark_answer_sym_tl
1631     }
1632   }
1633   \skip_horizontal:n { \dim_use:N #2 }
1634 }
1635 \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`.

```
1636 \keys_define:nn { enumext / anskey }
1637 {
1638   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1639   item-sym* .value_required:n = true,
1640   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1641   item-pos* .value_required:n = true,
1642   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1643   store-brk .default:n = true,
1644   store-brk .value_forbidden:n = true,
1645 }
```

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_ans_int`, then call internal function `\__enumext_store_anskey_code:nnnn` will “*store content*” in the *(sequence)* and in the *(prop list)*.

```
1646 \NewDocumentCommand \anskey { d() s o +m }
1647 {
1648   \bool_if:NF \l__enumext_store_active_bool
1649   {
1650     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1651   }
1652   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1653   {
1654     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1655   }
1656   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1657   {
1658     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1659   }
1660   \group_begin:
1661     \bool_if:NF \l__enumext_store_ans_bool
1662     {
1663       \bool_if:NT \l__enumext_check_ans_bool
1664       {
1665         \int_gincr:N \g__enumext_count_item_ans_int
1666       }
1667       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1668     }
1669 }
```

```

1669     \group_end:
1670 }

```

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

```
\__enumext_store_anskey_code:nnnn
```

The internal function `\__enumext_store_anskey_code:nnnn` first we pass the command `<argument>` to the `<prop list>`, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `store-ref` key and will call the function `\l__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.

```

1671 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1672 {
1673     \__enumext_store_addto_prop:n {#4}
1674     \bool_if:NT \l__enumext_store_ref_key_bool
1675     {
1676         \__enumext_store_internal_ref:
1677     }
1678     \__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`.

```

1679     \tl_clear:N \l__enumext_store_anskey_arg_tl
1680     \tl_if_novalue:nF {#3}
1681     {
1682         \keys_set:nn { enumext / anskey } {#3}
1683     }
1684     \bool_lazy_and:nnT
1685     { \l__enumext_store_columns_break_bool }
1686     { \bool_not_p:n { \l__enumext_starred_bool } }
1687     {
1688         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1689     }
1690     \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).

```

1691     \tl_if_novalue:nF {#1}
1692     {
1693         \int_set:Nn \l__enumext_store_columns_join_int {#1}
1694         \bool_if:NT \l__enumext_starred_bool
1695         {
1696             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1697             {
1698                 ( \exp_not:V \l__enumext_store_columns_join_int )
1699             }
1700         }
1701     }

```

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`.

```

1702     \bool_if:nTF {#2}
1703     {
1704         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1705         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1706         {
1707             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1708             {
1709                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1710             }
1711         }
1712         \dim_compare:nT
1713         {
1714             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1715         }
1716         {
1717             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1718             {
1719                 [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]

```

```

1720     }
1721   }
1722   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1723 }
1724 {
1725   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1726 }

```

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`.

```

1727   \bool_lazy_and:nnT
1728   { \l__enumext_store_ref_key_bool }
1729   { \l__enumext_hyperref_bool }
1730   {
1731     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1732     {
1733       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1734       { \exp_not:V \l__enumext_mark_ref_sym_tl }
1735     }
1736   }
1737   \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1738 }

```

(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`.

```

1739 \cs_new_protected:Nn \__enumext_store_internal_ref:
1740 {
1741   \cs_set_protected:Npn \__enumext_tmp:n ##1
1742   {
1743     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1744     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1745     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1746     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1747   }
1748   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1749   \cs_set:Npn \__enumext_tmp:n ##1
1750   { . \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.

```

1751   \bool_lazy_all:nT
1752   {
1753     { \g__enumext_starred_bool }
1754     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1755   }
1756   {
1757     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1758     { \tl_use:N \l__enumext_label_copy_vii_tl }
1759   }
1760   \bool_lazy_all:nT
1761   {
1762     { \l__enumext_standar_bool }
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     {
1769       \tl_use:N \l__enumext_label_copy_vii_tl
1770       \int_step_function:nnN { 1 } { { \l__enumext_level_int } \__enumext_tmp:n
1771     }
1772   }

```

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.

```

1773   \bool_lazy_all:nT
1774   {

```

```

1775     { \l__enumext_standar_bool }
1776     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1777     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1778     { \bool_not_p:n { \l__enumext_starred_bool } }
1779   }
1780   {
1781     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1782     {
1783       \tl_use:N \l__enumext_label_copy_i_tl
1784       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1785     }
1786   }
1787   \cs_set:Npn \__enumext_tmp:n ##1
1788   { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1789   \bool_lazy_all:nT
1790   {
1791     { \l__enumext_standar_bool }
1792     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1793     { \bool_not_p:n { \g__enumext_starred_bool } }
1794     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1795   }
1796   {
1797     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1798     {
1799       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1800       . \tl_use:N \l__enumext_label_copy_vii_tl
1801     }
1802   }

```

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

```

1803   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1804   {
1805     \l__enumext_store_name_tl \c_colon_str
1806     \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
1807   }

```

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.

```

1808   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1809   {
1810     \__enumext_newlabel:nn
1811     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1812     { \l__enumext_newlabel_arg_two_tl }
1813   }
1814   \l__enumext_store_write_aux_file_tl
1815 }

```

(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.

```

1816 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1817 {
1818   \par
1819   \bool_if:NT \l__enumext_starred_bool
1820   {
1821     \cs_set:Nn \__enumext_level: { vii }
1822   }
1823   \__enumext_print_keyans_box:cc
1824   { \l__enumext_labelwidth_ \__enumext_level: _dim }
1825   { \l__enumext_labelsep_ \__enumext_level: _dim }
1826   \__enumext_anskey_wrapper:n { #1 }
1827 }

```

(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.

```

1828 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1829 {
1830   \bool_if:NT \l__enumext_show_answer_bool
1831   {
1832     \__enumext_store_anskey_show_wrap:n { #1 }
1833   }
1834   \bool_if:NT \l__enumext_show_position_bool
1835   {
1836     \tl_set:Nx \l__enumext_mark_answer_sym_tl
1837     {
1838       \group_begin:
1839       \exp_not:N \normalfont
1840       \exp_not:N \footnotesize [ \int_eval:n
1841       {
1842         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1843       }
1844       ]
1845       \group_end:
1846     }
1847     \__enumext_store_anskey_show_wrap:n { #1 }
1848   }
1849 }

```

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

## 10.25 Common functions for keyans and keyanspic

### 10.25.1 Storing content in prop list

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

The function \\_\_enumext\_keyans\_addto\_prop:n will pass the contents of the \l\_\_enumext\_label\_v\_tl (current *label*) for the **keyans** environment and the \l\_\_enumext\_label\_vi\_tl (current *label*) for the **keyanspic** environment when using \item\* and \anspic\*, followed by the *contents* of the optional argument of both commands to the \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.

```

1850 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1851 {
1852   \tl_clear:N \l__enumext_store_keyans_label_tl
1853   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1854   {
1855     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1856   }
1857   {
1858     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1859   }
1860   \tl_if_novalue:nF { #1 }
1861   {
1862     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1863   }
1864   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1865 }

```

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

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

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

\\_\_enumext\_keyans\_internal\_ref:

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

```

1866 \cs_new_protected:Nn \__enumext_keyans_internal_ref:
1867 {

```

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.

```

1868   \cs_set_protected:Npn \__enumext_tmp:n ##1
1869   {
1870     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1871     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1872     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }

```



```

1873     \tl_reverse:c { l__enumext_label_copy_##1_tl }
1874   }
1875   \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1876   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1877   {
1878     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1879     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
1880   }
1881   {
1882     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1883     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
1884   }

```

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

```

1885   \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
1886   {
1887     \__enumext_store_name_tl \c_colon_str
1888     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
1889   }

```

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

```

1890   \tl_put_right:Ne \__enumext_store_write_aux_file_tl
1891   {
1892     \__enumext_newlabel:nn
1893     { \exp_not:V \__enumext_newlabel_arg_one_tl }
1894     { \__enumext_newlabel_arg_two_tl }
1895   }
1896   \bool_if:NT \__enumext_store_ref_key_bool
1897   {
1898     \__enumext_store_write_aux_file_tl
1899   }
1900 }

```

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

### 10.25.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n`

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the `\__enumext_label_v_tl` (“current label”) for the `keyans` environment and the `\__enumext_label_vi_tl` (current label) for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the contents of the optional argument of both commands to the `\__enumext_store_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

1901 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1902 {
1903   \tl_clear:N \__enumext_store_keyans_label_tl
1904   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1905   {
1906     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_vi_tl }
1907   }
1908   {
1909     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_v_tl }
1910   }
1911   \tl_if_novalue:nF { #1 }
1912   {
1913     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \c_space_tl #1 }
1914   }

```

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.

```

1915   \bool_lazy_and:nnT
1916   { \__enumext_store_ref_key_bool }
1917   { \__enumext_hyperref_bool }
1918   {
1919     \tl_put_right:Ne \__enumext_store_keyans_label_tl
1920     {
1921       \hfill \exp_not:N \hyperlink
1922       {
1923         \exp_not:V \__enumext_newlabel_arg_one_tl
1924       }
1925       { \exp_not:V \__enumext_mark_ref_sym_tl }
1926     }

```

```

1927     }
1928     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl

```

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

```

1929     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1930     \bool_if:NT \l__enumext_check_ans_bool
1931     {
1932         \int_gincr:N \g__enumext_count_item_ans_int
1933     }
1934 }

```

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

#### 10.25.4 Check for starred commands

`\__enumext_keyans_check_ans:nn`

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

```

1935 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1936 {
1937     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1938     {
1939         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1940     }
1941     { \tl_gclear:N \g__enumext_check_ans_item_tl }
1942 }

```

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

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

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

`\__enumext_keyans_show_left:n`

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

```

1943 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1944 {
1945     \bool_if:NT \l__enumext_show_answer_bool
1946     {
1947         \tl_put_left:Nn \l__enumext_label_v_tl
1948         {
1949             \__enumext_print_keyans_box:NN
1950             \l__enumext_labelwidth_i_dim
1951             \l__enumext_labelsep_i_dim
1952         }
1953         \tl_if_novalue:nF { #1 }
1954         { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1955     }
1956     \bool_if:NT \l__enumext_show_position_bool
1957     {
1958         \tl_set:Nc \l__enumext_mark_answer_sym_tl
1959         {
1960             \group_begin:
1961             \exp_not:N \normalfont
1962             \exp_not:N \footnotesize [ \int_eval:n
1963             {
1964                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1965                 + \l__enumext_keyans_level_int
1966             }
1967             ]
1968             \group_end:
1969         }
1970         \tl_put_left:Nn \l__enumext_label_v_tl
1971         {
1972             \__enumext_print_keyans_box:NN
1973             \l__enumext_labelwidth_i_dim
1974             \l__enumext_labelsep_i_dim
1975         }
1976         \tl_if_novalue:nF { #1 }

```

```

1977         { \tl_put_right:Nn \__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1978     }
1979 }

```

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

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

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

```

item-sym* Define and set item-sym* and item-pos* keys for enumext and enumext*.
item-pos*
1980 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1981 {
1982     \keys_define:nn { enumext / #1 }
1983     {
1984         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
1985         item-sym* .value_required:n = true,
1986         item-sym* .initial:n = { $\star$ },
1987         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
1988         item-pos* .value_required:n = true,
1989     }
1990 }
1991 \clist_map_inline:nn
1992 {
1993     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1994 }
1995 { \__enumext_tmp:nn #1 }

```

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

## 10.27 Redefining `\footnote` command

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

```

1996 \cs_new_protected:Nn \__enumext_footnotetext:nn
1997 {
1998     \footnotetext[#1]{#2}
1999 }
2000 \cs_new_protected:Nn \__enumext_renew_footnote:
2001 {
2002     \seq_gclear:N \g__enumext_footnote_arg_seq
2003     \seq_gclear:N \g__enumext_footnote_int_seq
2004     \RenewDocumentCommand \footnote { o +m }
2005     {
2006         \tl_if_novalue:nTF {##1}
2007         {
2008             \stepcounter{footnote}
2009             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2010         }
2011         {
2012             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2013         }
2014         \footnotemark [ \g__enumext_footnote_int ]
2015         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2016         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2017     }
2018 }
2019 \cs_new_protected:Nn \__enumext_print_footnote:
2020 {
2021     \seq_if_empty:NF \g__enumext_footnote_int_seq
2022     {
2023         \seq_map_pairwise_function:NNN
2024         \g__enumext_footnote_int_seq
2025         \g__enumext_footnote_arg_seq
2026         \__enumext_footnotetext:nn
2027     }
2028 }

```

(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

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)`.

```

2029 \cs_new_protected:Npn \__enumext_default_item:n #1
2030 {
2031   \tl_if_novalue:nTF {#1}
2032   {
2033     \bool_if:NT \l__enumext_check_ans_bool
2034     {
2035       \int_gincr:N \g__enumext_count_item_all_int
2036       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2037     }
2038     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2039     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2040   }
2041   {
2042     \bool_set_eq:cc
2043     { l__enumext_wrap_label_ \__enumext_level: _bool }
2044     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2045     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2046   }
2047 }

```

(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)`.

```

2048 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2049 {
2050   \tl_if_novalue:nF {#1}
2051   {
2052     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2053   }
2054   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2055   \tl_if_novalue:nTF {#2}
2056   {
2057     \dim_set_eq:cc
2058     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2059     { l__enumext_labelsep_ \__enumext_level: _dim }
2060   }

```

```

2061     {
2062         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
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 }

```

(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).

```

2072 \cs_new_protected:Nn \__enumext_redefine_item:
2073 {
2074     \RenewDocumentCommand \item { s o o }
2075     {
2076         \bool_if:nTF {##1}
2077         {
2078             \__enumext_starred_item:nn {##2} {##3}
2079         }
2080         { \__enumext_default_item:n {##2} }
2081     }
2082 }

```

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

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

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

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

```

2083 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2084 {
2085     \tl_if_novalue:nTF { #1 }
2086     {
2087         \bool_set_true:N \l__enumext_wrap_label_v_bool
2088         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2089     }
2090     {
2091         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2092         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2093     }
2094 }

```

(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*.

```

2095 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2096 {
2097     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2098     \__enumext_keyans_show_left:n { #1 }
2099     \bool_set_true:N \l__enumext_wrap_label_v_bool
2100     \__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*.

```

2101     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2102     \__enumext_keyans_addto_prop:n { #1 }
2103     \__enumext_keyans_internal_ref:
2104     \__enumext_keyans_addto_seq:n { #1 }
2105 }

```

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

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

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

```
2106 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2107 {
2108   \RenewDocumentCommand \item { s o }
2109   {
2110     \bool_if:nTF {##1}
2111     {
2112       \peek_remove_spaces:n
2113       {
2114         \__enumext_keyans_starred_item:n {##2}
2115       }
2116     }
2117     {
2118       \__enumext_keyans_default_item:n {##2}
2119     }
2120   }
2121 }
```

(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.

```
2122 \cs_new_protected:Nn \__enumext_item_starred:
2123 {
2124   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2125   {
2126     \mode_leave_vertical:
2127     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2128     \makebox[0pt][r]{ \tl_use:N \g__enumext_item_symbol_tl }
2129     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2130   }
2131 }
```

(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).

```
2132 \cs_new_protected:Nn \__enumext_make_label:
2133 {
2134   \RenewDocumentCommand \makeLabel { m }
2135   {
2136     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2137     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2138     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2139     {
2140       \__enumext_item_starred:
2141       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2142     }
2143     { ##1 }
2144     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2145     \tl_gclear:N \g__enumext_item_symbol_tl
2146   }
2147 }
```

(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).

```

2148 \cs_new_protected:Nn \__enumext_keyans_make_label:
2149 {
2150   \RenewDocumentCommand \makeLabel { m }
2151   {
2152     \tl_use:N \l__enumext_label_fill_left_v_tl
2153     \tl_use:N \l__enumext_label_font_style_v_tl
2154     \bool_if:NTF \l__enumext_wrap_label_v_bool
2155     {
2156       \__enumext_wrapper_label_v:n { ##1 }
2157     }
2158     { ##1 }
2159     \tl_use:N \l__enumext_label_fill_right_v_tl
2160   }
2161 }

```

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

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

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

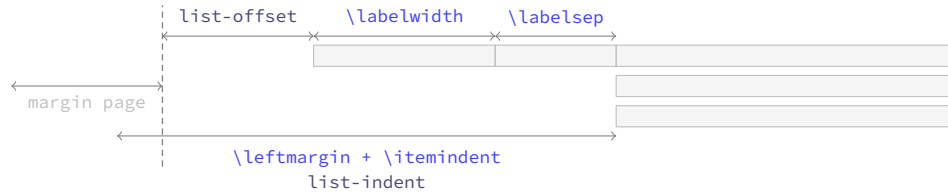


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

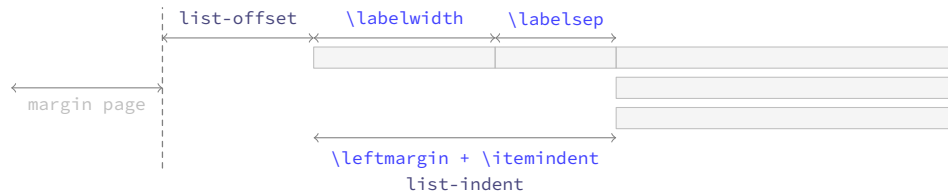


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

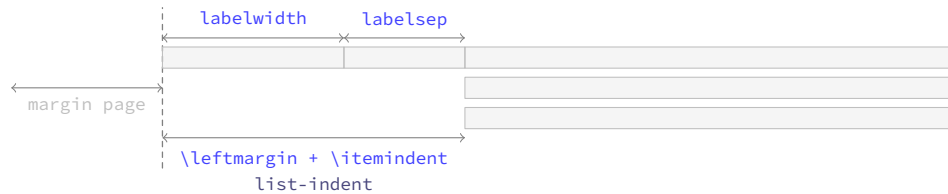


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNN
\__enumext_calc_hspace:cccccc

```

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

```

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

```

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

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

```

2162 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1#2#3#4#5#6#7
2163 {
2164   \dim_compare:nNnT { #1 } < { \c_zero_dim }

```



```

2165     {
2166       \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2167       \dim_set:Nn #1 { \dim_abs:n { #1 } }
2168     }
2169     \dim_compare:nNnT { #2 } < { \c_zero_dim }
2170     {
2171       \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2172       \dim_set:Nn #2 { \dim_abs:n { #2 } }
2173     }

```

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

```

2174     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2175     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2176     {
2177       \dim_set:Nn #6 { #1 + #2 - #4 }
2178       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2179     }
2180     {
2181       \dim_compare:nNnT { #4 } = { #1 + #2 }
2182       { \dim_set:Nn #6 { \c_zero_dim } }
2183       \dim_compare:nNnT { #4 } < { #1 + #2 }
2184       { \dim_set:Nn #6 { #1 + #2 - #4 } }
2185       \dim_compare:nNnT { #4 } > { #1 + #2 }
2186       {
2187         \dim_set:Nn #6 { -#1 - #2 + #4 }
2188         \dim_set:Nn #6 { #6*-1 }
2189       }
2190       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2191     }
2192   }
2193   \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:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

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.

```

2194   \cs_set_protected:Npn \__enumext_tmp:n #1
2195   {
2196     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2197     {
2198       \__enumext_calc_hspace:cccccc
2199       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2200       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2201       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2202       { \__enumext_leftmargin_tmp_#1_bool }
2203       \clist_map_inline:nn
2204       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2205       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2206       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2207       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2208       \usecounter { enumX#1 }
2209       \bool_lazy_and:nnTF
2210       { \str_if_eq_p:nn {#1} { i } }
2211       { \bool_if_p:N \__enumext_resume_bool }
2212       { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2213       {
2214         \setcounter { enumX#1 }
2215         { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2216       }
2217       \str_if_eq:nnTF {#1} { v }

```

```

2218     {
2219         \__enumext_keyans_redefine_item:
2220         \__enumext_keyans_make_label:
2221         \__enumext_keyans_fake_item:
2222         \bool_if:cT { \__enumext_show_length_#1_bool }
2223         {
2224             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2225         }
2226     }
2227     {
2228         \__enumext_redefine_item:
2229         \__enumext_make_label:
2230         \__enumext_use_key_ref:
2231         \__enumext_fake_item:
2232         \bool_if:cT { \__enumext_show_length_#1_bool }
2233         {
2234             \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_#1 }
2235         }
2236     }
2237 }
2238 }
2239 \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.

```

2240 \cs_set_protected:Npn \__enumext_tmp:n #1
2241 {
2242     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2243     {
2244         \__enumext_calc_hspace:ccccc
2245         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2246         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2247         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2248         { \__enumext_leftmargin_tmp_#1_bool }
2249         \clist_map_inline:nn
2250         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2251         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2252         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2253         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2254         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2255         \skip_zero:N \partopsep
2256         \usecounter { enumX#1 }
2257         \bool_lazy_and:nnTF
2258         { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2259         { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2260         {
2261             \setcounter { enumX#1 }
2262             { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2263         }
2264         \__enumext_use_key_ref_h:
2265         \str_if_eq:nnTF {#1} { vii }
2266         {
2267             \__enumext_fake_item_vii:
2268             \bool_if:cT { \__enumext_show_length_vii_bool }
2269             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2270         }
2271         {
2272             \__enumext_fake_item_viii:
2273             \bool_if:cT { \__enumext_show_length_#1_bool }
2274             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2275         }
2276     }
2277 }
2278 \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 and the internal `__enumext__mini_env*` environment.

```

2279 \NewDocumentEnvironment{enumext}{ 0{} }
2280 {
2281   \__enumext_safe_exec:
2282   \__enumext_parse_keys:n {#1}
2283   \__enumext_before_list:
2284   \__enumext_start_store_level:
2285   \__enumext_start_list:nn
2286   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2287   {
2288     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2289     \__enumext_before_keys_exec:
2290   }
2291   \__enumext_after_args_exec:
2292 }
2293 {
2294   \__enumext_stop_list:
2295   \__enumext_stop_store_level:
2296   \__enumext_after_list:
2297 }

```

(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.

```

2298 \cs_new_protected:Nn \__enumext_safe_exec:
2299 {
2300   \int_incr:N \l__enumext_level_int
2301   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2302   { \msg_fatal:nn { enumext } { list-too-deep } }
2303   \bool_set_true:N \l__enumext_standar_bool
2304   \bool_lazy_all:nT
2305   {
2306     { \bool_not_p:n { \l__enumext_starred_bool } }
2307     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2308   }
2309   {
2310     \bool_gset_true:N \g__enumext_standar_bool
2311   }
2312 }

```

(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.

```

2313 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2314 {
2315   \exp_args:Ne \keys_set:nn
2316   { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2317   \bool_if:NT \l__enumext_store_active_bool
2318   {
2319     \__enumext_parse_store_keys:n {#1}
2320   }
2321 }

```

(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:`.

```

2322 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2323 {
2324   \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2325   {
2326     \regex_match:nnT { \b columns\b } {#1}
2327     {
2328       \int_set_eq:cc

```

```

2329         { l__enumext_store_columns_ \__enumext_level: _int }
2330         { l__enumext_columns_ \__enumext_level: _int }
2331     \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2332     {
2333         columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2334     }
2335 }
2336 }
2337 \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2338 {
2339     \regex_match:nnT { \b columns-sep \b} {#1}
2340     {
2341         \dim_set_eq:cc
2342         { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2343         { l__enumext_columns_sep_ \__enumext_level: _dim }
2344     \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2345     {
2346         columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2347     }
2348 }
2349 }
2350 }

```

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

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

The \\_\_enumext\_start\_store\_level: and \\_\_enumext\_stop\_store\_level: functions activate the level saving mechanism for storage in *sequence* of the \anskey command.

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

If enumext are nested in enumext\* add \\_\_enumext\_store\_level\_open: to preserve the stored structure.

```

2351 \cs_new_protected:Nn \__enumext_start_store_level:
2352 {
2353     \bool_lazy_and:nnT { \l__enumext_store_active_bool } { \bool_not_p:n { \l__enumext_keyans_env.
2354     {
2355         \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2356         {
2357             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2358             \__enumext_store_level_open:
2359         }
2360         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2361         {
2362             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2363             \__enumext_store_level_open:
2364         }
2365     }
2366 }
2367 \cs_new_protected:Nn \__enumext_stop_store_level:
2368 {
2369     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2370     {
2371         \__enumext_store_level_close:
2372     }
2373 }

```

(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.

```

2374 \cs_new_protected:Nn \__enumext_before_list:
2375 {
2376     \__enumext_vspace_above:
2377     \__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.

```

2378     \__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.

```

2379 \dim_compare:nNtT
2380 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2381 {
2382   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2383   {
2384     \linewidth
2385     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2386     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2387   }

```

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

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

```

2388   \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2389   \int_gincr:N \g__enumext_minipage_stat_int
2390   \__enumext_mini_addvspace:
2391   \nointerlineskip\noindent
2392   \begin{\__enumext_mini_env*}
2393   { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2394   }
2395   \__enumext_multicols_start:
2396   }

```

(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.

```

2397 \cs_new_protected:Nn \__enumext_multicols_start:
2398 {
2399   \int_compare:nNtT
2400   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2401   {
2402     \dim_compare:nNtT
2403     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2404     {
2405       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2406       {
2407         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2408         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2409         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2410         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2411       }
2412     }
2413     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2414     \skip_zero:N \multicolsep
2415     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2416     {
2417       \dim_zero:N \columnseprule
2418     }

```

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.

```

2419   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2420   {
2421     \__enumext_multi_addvspace:
2422   }
2423   \raggedcolumns
2424   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2425   }
2426   }

```

(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.

```

2427 \cs_new_protected:Nn \__enumext_multicols_stop:
2428 {
2429   \int_compare:nNt
2430     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2431   {
2432     \end{multicols}
2433     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2434     {
2435       \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2436     }
2437   }

```

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.

```

2438   \bool_lazy_and:nnT { \l__enumext_check_ans_bool }{ \bool_not_p:n { \g__enumext_starred_bool }
2439   {
2440     \bool_gset_true:N \g__enumext_check_ans_show_bool
2441     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2442   }
2443 }

```

(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.

```

2444 \cs_new_protected:Nn \__enumext_after_list:
2445 {
2446   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2447   {
2448     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2449     {
2450       \msg_warning:nn { enumext } { missing-miniright }
2451       \miniright
2452     }
2453     \int_gzero:N \g__enumext_minipage_stat_int
2454     \end{__enumext_mini_env*}
2455     \par\addvspace { \l__enumext_minipage_after_skip }
2456   }
2457   { \__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.

```

2458   \__enumext_after_stop_list:
2459   \__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.

```

2460   \bool_set_false:N \l__enumext_standar_bool
2461   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2462   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2463   {
2464     \int_gset_eq:cN
2465     { g__enumext_resume_ \l__enumext_store_name_tl _int }
2466     { \value{enumXi} }
2467   }
2468 }

```

(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`.

```

2469 \__enumext_after_env:nn {enumext}
2470 {

```

```

2471 \bool_if:NT \g__enumext_check_ans_show_bool
2472 {
2473   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2474   {
2475     \__enumext_check_ans_active:
2476   }
2477 }
2478 \bool_gset_false:N \g__enumext_check_ans_show_bool
2479 \tl_gclear:N \g__enumext_store_name_tl
2480 }

```

### 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.

```

2481 \NewDocumentEnvironment{keyans}{0}{}
2482 {
2483   \__enumext_keyans_safe_exec:
2484   \__enumext_keyans_parse_keys:n {#1}
2485   \__enumext_before_list_v:
2486   \__enumext_start_list:nn
2487   { \tl_use:N \l__enumext_label_v_tl }
2488   {
2489     \__enumext_list_arg_two_v:
2490     \__enumext_before_keys_exec_v:
2491   }
2492   \__enumext_after_args_exec_v:
2493 }
2494 {
2495   \__enumext_keyans_check_ans:nn { item }{ keyans }
2496   \__enumext_stop_list:
2497   \__enumext_after_list_v:
2498 }

```

(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.

```

2499 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2500 {
2501   \bool_if:NF \l__enumext_store_active_bool
2502   {
2503     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2504   }
2505   \int_incr:N \l__enumext_keyans_level_int
2506   \bool_set_true:N \l__enumext_keyans_env_bool
2507   % set false for interfering with enumext nested in keyans (yes, its posible and crayze)
2508   \bool_set_false:N \l__enumext_store_active_bool
2509   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2510   {
2511     \msg_error:nn { enumext } { keyans-nested }
2512   }
2513   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2514   {
2515     \msg_error:nn { enumext } { keyans-wrong-level }
2516   }
2517 }

```

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

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

```

2518 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2519 {
2520   \keys_set:nn { enumext / keyans } {#1}
2521 }

```

(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.

```

2522 \cs_new_protected:Nn \__enumext_before_list_v:
2523 {
2524   \__enumext_vspace_above_v:
2525   \__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.

```

2526   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2527   {
2528     \dim_set:Nn \l__enumext_minipage_left_v_dim
2529     {
2530       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2531     }

```

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*.

```

2532   \bool_set_true:N \l__enumext_minipage_active_v_bool
2533   \int_gincr:N \g__enumext_minipage_stat_int
2534   \__enumext_keyans_mini_addvspace:
2535   \nointerlineskip\noindent
2536   \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2537 }

```

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

```

2538   \__enumext_keyans_multicols_start:
2539 }

```

(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.

```

2540 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2541 {
2542   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2543   {

```

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

```

2544     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2545     {
2546       \dim_set:Nn \l__enumext_columns_sep_v_dim
2547       {
2548         (
2549           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2550         ) / \l__enumext_columns_v_int
2551         - \l__enumext_listoffset_v_dim
2552       }
2553     }
2554     \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).

```

2555     \skip_zero:N \multicolsep
2556     \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.

```

2557     \bool_if:NF \l__enumext_minipage_active_v_bool
2558     {
2559       \__enumext_keyans_multi_addvspace:
2560     }
2561     \raggedcolumns

```

```

2562     \begin{multicols}{\l__enumext_columns_v_int }
2563   }
2564 }

```

(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.

```

2565 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2566 {
2567   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
2568   {
2569     \end{multicols}
2570     \bool_if:NF \l__enumext_minipage_active_v_bool
2571     {
2572       \par\addvspace{ \l__enumext_multicols_below_v_skip }
2573     }
2574   }
2575 }

```

(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.

```

2576 \cs_new_protected:Nn \__enumext_after_list_v:
2577 {
2578   \bool_if:NTF \l__enumext_minipage_active_v_bool
2579   {
2580     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2581     {
2582       \msg_warning:nn { enumext } { missing-miniright }
2583       \miniright
2584     }
2585     \int_gzero:N \g__enumext_minipage_stat_int
2586     \end{\__enumext_mini_env*}
2587     \par\addvspace{ \l__enumext_minipage_after_skip }
2588   }
2589   { \__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.

```

2590   \bool_set_false:N \l__enumext_keyans_env_bool
2591   \__enumext_after_stop_list_v:
2592   \__enumext_vspace_below_v:
2593 }

```

(End of definition for `\__enumext_after_list_v:`)

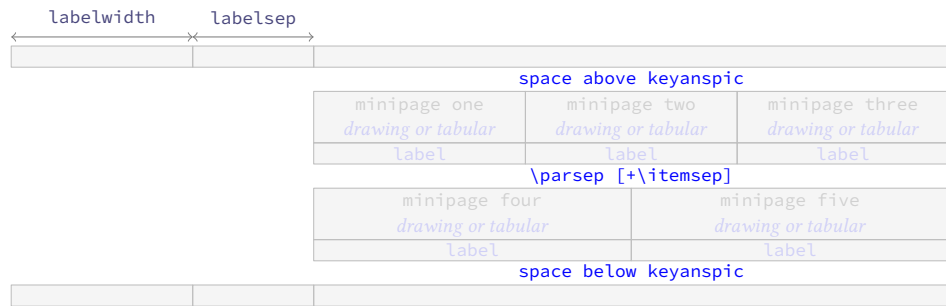
## 10.34 The environment `keyanspic` and `\anspic`

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.

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?](#).

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

### 10.34.1 The command `\anspic`

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

```
2594 \NewDocumentCommand \anspic { s o +m }
2595 {
```

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

```
2596   \bool_if:NF \l__enumext_store_active_bool
2597   {
2598     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2599   }
2600   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2601   {
2602     \msg_error:nn { enumext } { keyanspic-wrong-level }
2603   }
2604   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2605   {
2606     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2607   }
```

The three arguments are handled by the function `\__enumext_keyans_anspic_code:nnn` and stored in the sequence `\l__enumext_keyans_pic_body_seq` which is processed by the `keyanspic` environment.

```
2608   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2609   {
2610     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2611   }
2612 }
```

(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.

```
2613 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2614 {
2615   \stepcounter { enumXvi }
2616   #3 \\\
2617   \bool_if:nT { #1 }
2618   {
2619     \__enumext_keyans_addto_prop:n { #2 }
2620     \__enumext_keyans_internal_ref:
2621     \__enumext_keyans_addto_seq:n { #2 }
2622     \bool_lazy_or:nnT
2623     { \l__enumext_show_answer_bool }
2624     { \l__enumext_show_position_bool }
2625     {
2626       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2627       \__enumext_keyans_show_left:n { #2 }
2628       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2629     }
2630   }
2631   \tl_use:N \l__enumext_label_font_style_v_tl
2632   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
2633 }
```

(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.

```

2634 \NewDocumentEnvironment{keyanspic}{ o }
2635 {
2636   \__enumext_keyans_pic_safe_exec:
2637   \__enumext_start_list:nn
2638   { }
2639   {
2640     \__enumext_keyans_pic_arg_two:
2641   }

```

We apply the “adjusted” vertical spacing above the environment

```

2642   \vspace { \__enumext_keyans_pic_above_skip }
2643 }

```

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

```

2644 {
2645   \tl_if_novalue:nTF { #1 }
2646   {
2647     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2648   }
2649   { \__enumext_keyans_pic_do:n { #1 } }
2650   \__enumext_stop_list:
2651   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2652   \setcounter { enumXvi } { 0 }
2653   \vspace { \__enumext_topsep_v_skip }
2654   %\bool_set_false:N \l__enumext_store_active_bool
2655 }

```

(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.

```

2656 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2657 {
2658   \int_incr:N \l__enumext_keyans_pic_level_int
2659   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2660   {
2661     \msg_error:nn { enumext } { keyanspic-nested }
2662   }
2663 }

```

(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`.

```

2664 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2665 {
2666   \dim_compare:nNnT { #1 } < { 0pt }
2667   { \skip_set:Nn #1 { -#1 } }
2668 }

```

(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.

```

2669 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2670 {

```

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.

```

2671   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2672   \__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.

```

2673 \skip_add:Nn \parsep { \itemsep }
2674 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2675 \dim_zero:N \labelwidth
2676 \dim_zero:N \listparindent
2677 \dim_zero:N \labelsep
2678 \skip_zero:N \partopsep
2679 \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.

```

2680 \__enumext_keyans_pic_skip_abs:N \parsep
2681 \skip_set:Nn \l__enumext_keyans_pic_above_skip
2682 {
2683   \box_dp:N \strutbox
2684   + \l__enumext_topsep_v_skip
2685   - \parsep
2686 }
2687 \__enumext_item_std:w \scan_stop:
2688 }

```

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

```

\__enumext_keyans_pic_do:n
\__enumext_keyans_pic_do:e

```

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

```

2689 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2690 {
2691   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2692 }
2693 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

```
\__enumext_keyans_pic_row:n
```

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

```

2694 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2695 {
2696   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2697   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2698   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2699   \int_step_inline:nnn
2700     { \l__enumext_keyans_pic_above_int + 1 }
2701     { \l__enumext_keyans_pic_below_int }
2702     {
2703       \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2704       \centering
2705       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2706       \__enumext_endminipage:
2707     }
2708   \par
2709 }

```

(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

\\_enumext\_starred\_columns\_set\_vii:

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

```

2710 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2711 {
2712   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2713   {
2714     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2715     {
2716       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2717       / \l__enumext_columns_vii_int
2718     }
2719   }
2720   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2721   \dim_set:Nn \l__enumext_item_width_vii_dim
2722   {
2723     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2724     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2725     - \l__enumext_labelsep_vii_dim
2726   }
2727   \dim_zero_new:N \itemwidth
2728 }

```

(End of definition for \\_enumext\_starred\_columns\_set\_vii:.)

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

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

```

2729 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2730 {
2731   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2732   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2733   {
2734     \msg_warning:nnee { enumext } { item-joined }
2735     { \int_use:N \l__enumext_joined_item_vii_int }
2736     { \int_use:N \l__enumext_columns_vii_int }
2737     \int_set:Nn \l__enumext_joined_item_vii_int
2738     {
2739       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2740     }
2741   }
2742   \int_compare:nNnT
2743   { \l__enumext_joined_item_vii_int }
2744   >
2745   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2746   {
2747     \msg_warning:nnee { enumext } { item-joined-columns }
2748     { \int_use:N \l__enumext_joined_item_vii_int }
2749     {
2750       \int_eval:n
2751       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2752     }
2753     \int_set:Nn \l__enumext_joined_item_vii_int
2754     {
2755       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2756     }
2757   }

```

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

```

2758   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2759   {
2760     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2761     \int_decr:N \l__enumext_joined_item_aux_vii_int
2762     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2763     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2764     \dim_set:Nn \l__enumext_joined_width_vii_dim
2765     {
2766       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2767       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2768         + \l__enumext_columns_sep_vii_dim
2769         ) * \l__enumext_joined_item_aux_vii_int

```

```

2770     }
2771     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2772   }
2773   {
2774     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2775     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2776   }
2777 }

```

(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`.

```

2778 \cs_new_protected:Nn \__enumext_start_mini_vii:
2779 {
2780   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2781   {
2782     \dim_set:Nn \l__enumext_minipage_left_vii_dim
2783     {
2784       \linewidth
2785       - \l__enumext_minipage_right_vii_dim
2786       - \l__enumext_minipage_hsep_vii_dim
2787     }
2788     \bool_set_true:N \l__enumext_minipage_active_vii_bool
2789     \dim_gset_eq:NN
2790       \g__enumext_minipage_right_vii_dim
2791       \l__enumext_minipage_right_vii_dim
2792     \__enumext_mini_addvspace_vii:
2793     \nointerlineskip\noindent
2794     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2795   }
2796 }

```

(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*”.

```

2797 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2798 {
2799   \bool_if:NT \l__enumext_minipage_active_vii_bool
2800   {
2801     \end{__enumext_mini_env*}
2802     \hfill
2803     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2804   }
2805 }

```

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*”.

```

2806 \__enumext_after_env:nn {enumext*}
2807 {
2808   \bool_if:NT \g__enumext_minipage_active_vii_bool
2809   {
2810     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2811     \par\addvspace { \g__enumext_minipage_right_skip }
2812     \bool_if:NF \g__enumext_minipage_center_vii_bool
2813     {
2814       \centering
2815     }
2816     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2817     \end{__enumext_mini_env*}
2818     \par\addvspace{ \g__enumext_minipage_after_skip }
2819   }
2820   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2821   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2822   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2823   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2824 }

```



(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.

```

2825 \NewDocumentEnvironment{enumext*}{o }
2826 {
2827   \__enumext_safe_exec_vii:
2828   \__enumext_parse_keys_vii:n {#1}
2829   \__enumext_before_list_vii:
2830   \__enumext_start_store_level_vii:
2831   \__enumext_start_list:nn { }
2832   {
2833     \__enumext_list_arg_two_vii:
2834     \__enumext_before_keys_exec_vii:
2835   }
2836   \__enumext_starred_columns_set_vii:
2837   \item[] \scan_stop:
2838   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2839   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2840 }
2841 {
2842   \__enumext_stop_item_tmp_vii:
2843   \__enumext_remove_extra_parsep_vii:
2844   \__enumext_stop_list:
2845   \__enumext_stop_store_level_vii:
2846   \__enumext_after_list_vii:
2847 }

```

(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`.

```

2848 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2849 {
2850   \int_incr:N \l__enumext_level_h_int
2851   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2852   {
2853     \msg_error:nn { enumext } { nested }
2854   }
2855   \bool_set_true:N \l__enumext_starred_bool
2856   \bool_lazy_all:nT
2857   {
2858     { \bool_not_p:n { \l__enumext_standar_bool } }
2859     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2860   }
2861   {
2862     \bool_gset_true:N \g__enumext_starred_bool
2863   }
2864 }

```

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

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

```

2865 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2866 {
2867   \tl_if_novalue:nF {#1}
2868   {
2869     \keys_set:nn { enumext / enumext* } {#1}
2870     \bool_if:NT \l__enumext_store_active_bool
2871     {
2872       \__enumext_parse_store_keys_vii:n {#1}
2873     }
2874   }
2875 }

```

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

\\_\_enumext\_parse\_store\_keys\_vii:n

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

```

2876 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2877 {
2878   \bool_if:NF \l__enumext_store_columns_vii_bool
2879   {
2880     \regex_match:nnT { \b columns\b } {#1}
2881     {
2882       \int_set_eq:NN
2883         \l__enumext_store_columns_vii_int
2884         \l__enumext_columns_vii_int
2885       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2886         {
2887           columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2888         }
2889     }
2890   }
2891   \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2892   {
2893     \regex_match:nnT { \b columns-sep\b } {#1}
2894     {
2895       \dim_set_eq:NN
2896         \l__enumext_store_columns_sep_vii_dim
2897         \l__enumext_columns_sep_vii_dim
2898       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2899         {
2900           columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2901         }
2902     }
2903   }
2904 }

```

(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`.

```

2905 \cs_new_protected:Nn \__enumext_before_list_vii:
2906 {
2907   \__enumext_vspace_above_vii:
2908   \__enumext_before_args_exec_vii:
2909   \__enumext_start_mini_vii:
2910 }

```

(End of definition for \\_\_enumext\_before\_list\_vii:.)

\\_\_enumext\_after\_list\_vii:

The function \\_\_enumext\_after\_list: firsts 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.

```

2911 \cs_new_protected:Nn \__enumext_after_list_vii:
2912 {
2913   \__enumext_stop_mini_vii:
2914   \__enumext_after_stop_list_vii:
2915   \__enumext_vspace_below_vii:
2916   \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2917   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2918   {
2919     {
2920       \int_gset_eq:cN
2921         { g__enumext_resume_ \l__enumext_store_name_tl _int }
2922         { \value{enumXvii} }
2923     }
2924     \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2925     {

```

```

2926         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2927         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2928     }
2929     \bool_gset_false:N \g__enumext_starred_bool
2930     \bool_set_false:N \l__enumext_starred_bool
2931 }

```

(End of definition for `\__enumext_after_list_vii:`)

```

\__enumext_start_store_level_vii:
\__enumext_stop_store_level_vii:

```

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

```

2932 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2933 {
2934     \bool_if:NT \l__enumext_store_active_bool
2935     {
2936         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2937         {
2938             \__enumext_store_level_open_vii:
2939         }
2940     }
2941 }
2942 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2943 {
2944     \bool_if:NT \l__enumext_store_active_bool
2945     {
2946         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2947         {
2948             \__enumext_store_level_close_vii:
2949         }
2950     }
2951 }

```

(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`.

```

2952 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2953 {
2954     \__enumext_stop_item_tmp_vii:
2955     \int_incr:N \l__enumext_item_column_pos_vii_int
2956     \int_gincr:N \g__enumext_item_count_all_vii_int
2957     \__enumext_item_peek_args_vii:
2958 }

```

(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).

```

2959 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2960 {
2961     \peek_meaning:NTF (
2962         { \__enumext_joined_item_vii:w }
2963         { \__enumext_joined_item_vii:w (1) }
2964     }

```

(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`.

```

2965 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2966 {

```

```

2967     \__enumext_starred_joined_item_vii:n {#1}
2968     \peek_meaning_remove:NTF *
2969     { \__enumext_starred_item_vii:w }
2970     { \__enumext_standard_item_vii:w }
2971 }

```

(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 ].

```

2972 \cs_new_protected:Npn \__enumext_standard_item_vii:w
2973 {
2974     \bool_set_false:N \l__enumext_item_starred_vii_bool
2975     \peek_meaning:NTF [
2976     {
2977         \bool_set_eq:NN
2978         \l__enumext_wrap_label_vii_bool
2979         \l__enumext_wrap_label_opt_vii_bool
2980         \__enumext_start_item_vii:w
2981     }
2982     {
2983         \bool_set_true:N \l__enumext_wrap_label_vii_bool
2984         \legacy_if_set_true:n { @noitemarg }
2985         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
2986     }
2987 }

```

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

\\_\_enumext\_starred\_item\_vii:w

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

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

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

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

```

2988 \cs_new_protected:Npn \__enumext_starred_item_vii:w
2989 {
2990     \bool_set_true:N \l__enumext_item_starred_vii_bool
2991     \bool_set_true:N \l__enumext_wrap_label_vii_bool
2992     \peek_meaning:NTF [
2993     { \__enumext_starred_item_vii_aux_i:w }
2994     { \__enumext_starred_item_vii_aux_ii:w }
2995 }
2996 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
2997 {
2998     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
2999     \__enumext_starred_item_vii_aux_ii:w
3000 }
3001 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3002 {
3003     \peek_meaning:NTF [
3004     { \__enumext_starred_item_vii_aux_iii:w }
3005     {
3006         \dim_set_eq:NN
3007         \l__enumext_item_symbol_sep_vii_dim
3008         \l__enumext_labelsep_vii_dim
3009         \legacy_if_set_true:n { @noitemarg }
3010         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3011     }
3012 }
3013 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3014 {
3015     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3016     \legacy_if_set_true:n { @noitemarg }
3017     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3018 }

```

(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.

```

3019 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3020 {
3021   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3022   \legacy_if:nT { @noitemarg }
3023   {
3024     \legacy_if_set_false:n { @noitemarg }
3025     \legacy_if:nT { @nmbrlist }
3026     {
3027       \bool_if:NT \__enumext_hyperref_bool
3028       {
3029         \legacy_if_set_true:n { @hyper@item }
3030       }
3031       \refstepcounter{enumXvii}
3032       % code for check-ans
3033       \bool_if:NT \__enumext_check_ans_bool
3034       {
3035         % If true |no-store| key => nested in |enumext|
3036         \bool_if:NTF \__enumext_store_ans_bool
3037         {
3038           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3039           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3040         }
3041         {
3042           \int_gincr:N \g__enumext_count_item_all_int
3043           \int_gincr:N \g__enumext_count_level_vii_int
3044         }
3045       }
3046     }
3047   }

```

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 `\__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.

```

3048 \group_begin:
3049 \lrbox{ \__enumext_item_text_vii_box }
3050 \bool_if:NF \__enumext_footnotes_key_bool
3051 {
3052   \__enumext_renew_footnote:
3053 }
3054 \bool_if:NT \__enumext_item_starred_vii_bool
3055 {
3056   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3057   {
3058     \tl_gset_eq:NN
3059     \g__enumext_item_symbol_aux_vii_tl \__enumext_item_symbol_vii_tl
3060   }
3061   \mode_leave_vertical:
3062   \skip_horizontal:n { -\__enumext_item_symbol_sep_vii_dim }
3063   \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3064   \skip_horizontal:N \__enumext_item_symbol_sep_vii_dim
3065   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3066 }
3067 \group_begin:
3068 \tl_use:N \__enumext_label_font_style_vii_tl
3069 \bool_if:NTF \__enumext_wrap_label_vii_bool
3070 {
3071   \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]

```

```

3072         { \__enumext_wrapper_label_vii:n {#1} }
3073     }
3074     {
3075         \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3076     }
3077     \group_end:
3078     \skip_horizontal:N \__enumext_labelsep_vii_dim
3079     \tl_use:N \__enumext_after_list_args_vii_tl
3080     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3081     \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3082     \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3083     \tl_use:N \__enumext_fake_item_indent_vii_tl
3084 }

```

(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.

```

3085 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3086 {
3087     \__enumext_endminipage:
3088     \endlrbox
3089     \group_end:
3090     \box_set_wd:Nn \__enumext_item_text_vii_box
3091     {
3092         \__enumext_joined_width_vii_dim
3093         + \__enumext_labelwidth_vii_dim
3094         + \__enumext_labelsep_vii_dim
3095     }
3096     \int_set:Nn \hbadness { 10000 }
3097     \box_use:N \__enumext_item_text_vii_box
3098     \bool_if:NF \__enumext_footnotes_key_bool
3099     {
3100         \__enumext_print_footnote:
3101     }
3102     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3103     {
3104         \par\noindent
3105         \int_zero:N \__enumext_item_column_pos_vii_int
3106     }
3107     { \hspace{ \__enumext_columns_sep_vii_dim } }
3108 }

```

(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.

```

3109 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3110 {
3111     \int_compare:nNnT
3112     {
3113         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3114     }
3115     =
3116     { \c_zero_int }
3117     {
3118         \par
3119         \vspace{ -\__enumext_itemsep_vii_skip }
3120         \int_gzero:N \g__enumext_item_count_all_vii_int
3121     }
3122 }

```

(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.

```

3123 \__enumext_after_env:nn {enumext*}
3124 {
3125     \bool_if:NT \g__enumext_check_ans_show_h_bool
3126     {

```

```

3127         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3128         {
3129             \__enumext_check_ans_active_vii:
3130         }
3131     }
3132     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3133     \tl_gclear:N \g__enumext_store_name_tl
3134 }

```

### 10.36 The command \getkeyans

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

```

3135 \NewDocumentCommand \getkeyans { m }
3136 {
3137     \exp_args:Ne \__enumext_getkeyans_aux:n
3138     { \tl_to_str:e { \text_expand:n {#1} } }
3139 }

```

(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  $\langle \text{argument} \rangle$  using “:”. If “:” is omitted it will return an error.

```

3140 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3141 {
3142     \str_if_in:nnTF {#1} { : }
3143     {
3144         \use:e
3145         {
3146             \cs_set:Npn \exp_not:N \__enumext_tmp:w #1 \c_colon_str ##2 \scan_stop:
3147             { {##1} {##2} }
3148         }
3149         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3150     }
3151     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3152 }

```

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

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

```

3153 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3154 {
3155     \prop_if_exist:cF { g__enumext_#1_prop }
3156     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3157     \group_begin:
3158     \prop_item:cn { g__enumext_#1_prop }{#2}
3159     \group_end:
3160 }

```

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

### 10.37 The command \printkeyans

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

```

3161 \keys_define:nn { keyanskey / print }
3162 {
3163     level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3164                         {
3165                             \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3166                         },
3167     level-1 .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3168     level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3169                         {
3170                             \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3171                         },
3172     level-2 .initial:n   = { nosep, label=(\alph*), first=\small, font=\small },

```



```

3173   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3174                       {
3175                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3176                       },
3177   level-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
3178   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3179                       {
3180                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3181                       },
3182   level-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
3183   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3184                       {
3185                         \setenumext[enumext*] {#1} %\setenumext[print,*] {#1}
3186                       },
3187   level-* .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3188   }

```

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

```

3189 \NewDocumentCommand \printkeyans { s O{} m }
3190 {
3191   \group_begin:
3192     \tl_use:N \l__enumext_print_keyans_i_tl
3193     \tl_use:N \l__enumext_print_keyans_ii_tl
3194     \tl_use:N \l__enumext_print_keyans_iii_tl
3195     \tl_use:N \l__enumext_print_keyans_iv_tl
3196     \tl_use:N \l__enumext_print_keyans_vii_tl
3197     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3198   \group_end:
3199 }

```

(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

```

3200 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3201 {
3202   \seq_if_exist:cTF { g__enumext_#3_seq }
3203   {
3204     \seq_if_empty:cF { g__enumext_#3_seq }
3205     {
3206       \bool_if:nTF {#1}
3207       {
3208         \begin{enumext*}[#2]
3209         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3210         \end{enumext*}
3211       }
3212       {
3213         \begin{enumext}[#2]
3214         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3215         \end{enumext}
3216       }
3217     }
3218   }
3219   {
3220     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3221   }
3222 }

```

(End of definition for *\\_\_enumext\_printkeyans:nnn*.)

## 10.38 The command *\setenumext*

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

```

3223 \keys_define:nn { enumext / meta-families }
3224 {
3225   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3226   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },

```

```

3227   level-3   .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3228   level-4   .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3229   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
3230   enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3231   keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3232   print-1   .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3233   print-2   .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3234   print-3   .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3235   print-4   .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3236   print-*   .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3237   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3238 }

```

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

```

3239 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3240 {
3241   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3242   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3243 }

```

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

```

3244 \NewDocumentCommand \setenumext { o +m }
3245 {
3246   \tl_if_novalue:nTF {#1}
3247   {
3248     \seq_map_inline:Nn \c__enumext_all_families_seq
3249   }
3250   {
3251     \seq_clear:N \l__enumext_setkey_tmpa_seq
3252     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3253     \int_set:Nn \l__enumext_setkey_tmpa_int
3254     {
3255       \seq_count:N \l__enumext_setkey_tmpb_seq
3256     }
3257     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3258     {
3259       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3260       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3261       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3262       {
3263         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3264       }
3265     }
3266     {
3267       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3268     }
3269     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3270     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3271     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3272   }
3273   {
3274     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3275   }
3276 }

```

(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.

```

3277 \cs_new_protected:Npn \__enumext_set_parse:n #1
3278 {
3279   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3280   \int_step_inline:nnn { 0 } { 4 } % <- max level
3281   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3282   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3283   {
3284     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3285     { \tl_trim_spaces:n {#1} }
3286   }
3287   { \__enumext_set_error:nn {#1} { } }
3288 }
3289 \cs_new_protected:Npn \__enumext_set_error:nn #1#2
3290 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\__enumext_set_parse:n` and `\__enumext_set_error:nn`)

### 10.39 Messages

Message used by package-load for `multicol` and package.

```
3291 \msg_new:nnn { enumext } { package-load }
3292 {
3293   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3294 }
```

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

```
3295 \msg_new:nnn { enumext } { counters }
3296 {
3297   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3298   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3299 }
```

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

```
3300 \msg_new:nnn { enumext } { invalid-key }
3301 {
3302   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3303 }
3304 \msg_new:nnn { enumext } { unknown-key-family }
3305 {
3306   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3307 }
```

Messages used in length calculation.

```
3308 \msg_new:nnn { enumext } { width-negative }
3309 {
3310   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3311   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
3312 }
3313 \msg_new:nnn { enumext } { width-zero }
3314 {
3315   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3316   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
3317 }
```

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

```
3318 \msg_new:nnn { enumext } { list-lengths }
3319 {
3320   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3321   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3322   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3323   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3324   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3325   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3326   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3327   \__enumext_show_length:nnn { skip } { topsep } {#1}
3328   \__enumext_show_length:nnn { skip } { parsep } {#1}
3329   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3330   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3331   ****~
3332 }
```

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

```
3333 \msg_new:nnn { enumext } { list-lengths-not-nested }
3334 {
3335   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3336   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3337   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3338   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3339   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3340   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3341   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3342   \__enumext_show_length:nnn { skip } { topsep } {#1}
3343   \__enumext_show_length:nnn { skip } { parsep } {#1}
3344   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3345   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3346   ****~
3347 }
```

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

```

3348 \msg_new:nnn { enumext } { items-same-answer }
3349 {
3350     *****~Checking~answers~on~'#1'~OK~*****\\
3351     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3352     *****
3353     \prg_replicate:nn { 7 + \str_count:n [#1] } { * }
3354 }
3355 \msg_new:nnn { enumext } { item-different-answer }
3356 {
3357     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3358     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3359 }

```

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

```

3360 \msg_new:nnn { enumext } { missing-starred }
3361 {
3362     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3363 }

```

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

```

3364 \msg_new:nnn { enumext } { list-too-deep }
3365 {
3366     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
3367     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3368 }

```

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

```

3369 \msg_new:nnn { enumext } { anskey-wrong-place }
3370 {
3371     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3372     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3373 }
3374 \msg_new:nnn { enumext } { anspic-wrong-place }
3375 {
3376     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3377     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3378 }
3379 \msg_new:nnn { enumext } { command-wrong-place }
3380 {
3381     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3382     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3383 }

```

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

```

3384 \msg_new:nnn { enumext } { keyans-nested }
3385 {
3386     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3387 }
3388 \msg_new:nnn { enumext } { keyans-wrong-level }
3389 {
3390     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3391     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3392 }
3393 \msg_new:nnn { enumext } { wrong-place }
3394 {
3395     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3396     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3397 }
3398 \msg_new:nnn { enumext } { keyanspic-nested }
3399 {
3400     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
3401 }
3402 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3403 {
3404     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
3405     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3406 }

```

Messages used by `\getkeyans` command.

```

3407 \msg_new:nnn { enumext } { undefined-storage-anskey }
3408 {
3409     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3410 }

```

Messages used by `\miniright` command.

```

3411 \msg_new:nnn { enumext } { missing-miniright }
3412 {
3413   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
3414   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3415 }
3416 \msg_new:nnn { enumext } { wrong-miniright-place }
3417 {
3418   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3419   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3420 }
3421 \msg_new:nnn { enumext } { wrong-miniright-use }
3422 {
3423   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3424   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3425 }

```

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

```

3426 \msg_new:nnn { enumext } { nested }
3427 {
3428   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3429 }
3430 \msg_new:nnn { enumext } { item-joined }
3431 {
3432   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3433 }
3434 \msg_new:nnn { enumext } { item-joined-columns }
3435 {
3436   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3437 }

```

## 10.40 Finish package

Finish package implementation.

```

3438 \file_input_stop:
3439 </package>

```

# 11 Index of Implementation

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

Symbols

\\*

.....

386

\+

.....

196

\-

.....

196

\\

204, 2616, 3297, 3310, 3315, 3320, 3335, 3350, 3351, 3366, 3371, 3376, 3381, 3390, 3395, 3404, 3413, 3418, 3423

A

above

.....

1197

above\*

.....

1197

\addvspace

844, 872, 995, 1074, 1137, 1143, 1171, 1188, 2435, 2455, 2572, 2587, 2811, 2818

after

.....

682

align

.....

340

\Alph

.....

29, 33

\Alph

.....

292, 469, 487, 500, 3182

\alph

.....

29, 33

\alph

.....

293, 467, 3172

\anskey

.....

10, 56, 1636

\anspic

.....

12, 76, 77, 2594

\arabic

.....

29, 30

\arabic

.....

291, 466, 486, 3167, 3187

B

\b

.....

2326, 2339, 2880, 2893

\baselineskip

.....

41

\baselineskip

.....

1596, 1604

before

.....

682

before\*

.....

682

below

.....

1197

below\*

.....

1197

bool commands:

\bool\_gset\_false:N

.....

2478, 2820, 2929, 3132

\bool\_gset\_true:N

786, 2310, 2440, 2803, 2821, 2862, 2926

\bool\_if:NTF

232, 244, 261, 1219, 1233, 1246, 1257, 1268, 1279, 1290, 1301, 1332, 1339, 1350, 1408, 1410, 1558, 1582, 1589, 1617, 1648, 1661, 1663, 1674, 1694, 1819, 1830, 1834, 1896, 1930, 1945, 1956, 2033, 2064, 2138, 2154, 2222, 2232, 2268, 2273, 2317, 2324, 2337, 2369, 2419, 2433, 2446, 2471, 2501, 2557, 2570, 2578, 2596, 2799, 2808, 2812, 2870, 2878, 2891, 2934, 2944, 3027, 3033, 3036, 3050, 3054, 3069, 3098, 3125

\bool\_if:nTF

1172, 1189, 1702, 2076, 2110, 2174, 2617, 3206

\bool\_if\_p:N

.....

2211, 2258

\bool\_lazy\_all:nTF

1388, 1751, 1760, 1773, 1789, 2304, 2856

\bool\_lazy\_and:nnTF

1684, 1727, 1915, 2209, 2257, 2353, 2438, 2924

\bool\_lazy\_or:nnTF

.....

2622

\bool\_new:N

21, 22, 23, 24, 25, 31, 33, 42, 63, 68, 69, 74, 75, 78, 94, 96, 98, 101, 102, 111, 112, 113, 124, 125, 150, 161, 163

\bool\_not\_p:n

1686, 1778, 1793, 2306, 2353, 2438, 2858

\bool\_set\_eq:NN

.....

2042, 2091, 2977

\bool\_set\_false:N

241, 1483, 1486, 1506, 1509, 2460, 2508, 2590, 2654, 2671, 2930, 2974

\bool\_set\_true:N

223, 227, 333, 610, 1203, 1208, 1327, 1346, 1357, 1482, 1485, 1505, 1508, 1518, 1525, 2038,

2069, 2087, 2099, 2303, 2357, 2362, 2388, 2506, 2532, 2788, 2855, 2983, 2990, 2991

box commands:

\box\_dp:N

891, 895, 899, 910, 914, 925, 934, 940, 950, 963, 969, 975, 1006, 1007, 1008, 1011, 1021, 1025, 1034, 1041, 1046, 1054, 1083, 1084, 1087, 1094, 1107, 1115, 1121, 1129, 2683

\box\_new:N

.....

39, 156

\box\_set\_wd:Nn

.....

3090

\box\_use:N

.....

3097

\box\_wd:N

.....

299

C

\c

.....

386, 387, 510, 512, 524, 526

\cB

.....

387

\cE

.....

387

\centering

.....

1174, 1191, 2704, 2814

check-ans

.....

1359

Document class:

article

.....

34

clist commands:

\clist\_const:Nn

.....

168

\clist\_map\_function:nN

.....

2691

\clist\_map\_inline:Nn

339, 552, 615, 681, 696, 777, 1213

\clist\_map\_inline:nn

30, 47, 53, 65, 77, 100, 123, 133, 147, 167, 364, 381, 620, 792, 1369, 1495, 1513, 1534, 1748, 1875, 1991, 2203, 2206, 2239, 2249, 2252, 2278

\columnbreak

.....

57

\columnbreak

.....

1688

columns

.....

761

columns\*

.....

1514

columns-sep

.....

761

columns-sep\*

.....

1514

\columnsep

.....

72, 75

\columnsep

.....

2413, 2554

\columnseprule

.....

72, 75

\columnseprule

.....

2417, 2556

Commands provide by enumext:

\anskey

23, 24, 49–51, 54, 56, 58–60, 62, 71, 84, 88, 89, 92

\anspic\*

.....

23, 60–62, 77–79, 88, 89

\anspic

.....

54, 76–79, 92

\getkeyans

.....

54, 88, 92

\item\*

.....

23, 54, 60–62, 64, 65, 85, 88, 89

\itemwidth

.....

80

\item

.....

64, 65, 80, 84–86

\miniright

.....

22, 38, 46, 47, 72, 73, 75, 76, 93

\printkeyans

.....

23, 54, 88

\setenumext

.....

23, 89–91

Counters defined by enumext:

enumXiii

.....

22, 28

enumXii

.....

22, 28

enumXiv

.....

22, 28

enumXi

.....

22, 28

enumXviii

.....

22, 28

enumXvii

.....

22, 28, 86

enumXvi

.....

22, 28

enumXv

.....

22, 28

©2024 by Pablo González L

94 / 104

## cs commands:

`\cs_generate_variant:Nn` 301, 317, 516, 532, 1548, 1555, 1635, 2193, 2693  
`\cs_if_exist:NTF` ..... 271  
`\cs_new:Nn` ..... 181  
`\cs_new:Npn` ..... 185, 190, 200  
`\cs_new_eq:NN` 208, 209, 210, 214, 215, 246, 247, 250, 251  
`\cs_new_protected:Nn` . 219, 382, 402, 434, 697, 701, 705, 709, 713, 717, 721, 725, 729, 733, 737, 741, 745, 749, 753, 757, 793, 805, 829, 846, 857, 881, 956, 980, 997, 1059, 1076, 1098, 1133, 1139, 1214, 1228, 1242, 1253, 1264, 1275, 1286, 1297, 1337, 1348, 1406, 1418, 1441, 1556, 1580, 1587, 1615, 1622, 1739, 1866, 1996, 2000, 2019, 2072, 2106, 2122, 2132, 2148, 2298, 2351, 2367, 2374, 2397, 2427, 2444, 2499, 2522, 2540, 2565, 2576, 2613, 2656, 2669, 2689, 2694, 2710, 2778, 2797, 2848, 2905, 2911, 2932, 2942, 2959, 3109  
`\cs_new_protected:Npn` 173, 177, 254, 269, 286, 296, 302, 390, 409, 503, 517, 1161, 1180, 1324, 1374, 1539, 1549, 1671, 1816, 1828, 1850, 1901, 1935, 1943, 2029, 2048, 2083, 2095, 2162, 2196, 2242, 2313, 2322, 2518, 2664, 2729, 2865, 2876, 2965, 2972, 2988, 2996, 3001, 3013, 3140, 3153, 3200, 3277, 3289  
`\cs_new_protected_nopar:Nn` ..... 2952, 3085  
`\cs_new_protected_nopar:Npn` ..... 3019  
`\cs_set:Nn` ..... 188, 1821  
`\cs_set:Npn` ..... 1749, 1787, 3146  
`\cs_set_eq:NN` ..... 187, 192, 2838, 2839, 3021  
`\cs_set_protected:Nn` ..... 621, 637, 649, 661  
`\cs_set_protected:Npn` . 26, 40, 48, 60, 66, 90, 118, 129, 141, 148, 318, 340, 369, 450, 470, 533, 553, 597, 616, 673, 682, 761, 778, 1197, 1359, 1467, 1496, 1514, 1741, 1868, 1980, 2194, 2240  
`\cs_to_str:N` ..... 288, 311

## D

`\d` ..... 196  
`\DeclareDocumentEnvironment` ..... 874

## dim commands:

`\dim_abs:n` ..... 2167, 2172  
`\dim_add:Nn` ..... 2674  
`\dim_compare:nNnTF` . 623, 639, 651, 663, 1163, 1182, 2164, 2169, 2175, 2181, 2183, 2185, 2379, 2402, 2526, 2544, 2666, 2712, 2780  
`\dim_compare:nTF` ..... 1712  
`\dim_gset_eq:NN` ..... 2789  
`\dim_gzero:N` ..... 2823  
`\dim_new:N` .. 36, 43, 44, 45, 62, 97, 107, 157, 158, 164  
`\dim_set:Nn` .. 299, 611, 1526, 2062, 2167, 2172, 2174, 2177, 2178, 2182, 2184, 2187, 2188, 2190, 2382, 2405, 2528, 2546, 2696, 2714, 2721, 2764, 2782, 3015  
`\dim_set_eq:NN` 457, 477, 493, 497, 2057, 2205, 2251, 2341, 2413, 2554, 2771, 2774, 2775, 2895, 3006  
`\dim_use:N` 624, 632, 1164, 1170, 1625, 1628, 1633, 2127, 2129, 2380, 2385, 2386, 2393, 2403, 2407, 2408, 2410  
`\dim_zero:N` ..... 2417, 2556, 2675, 2676, 2677  
`\dim_zero_new:N` ..... 2727  
`\c_zero_dim` 626, 640, 652, 664, 1164, 1182, 1714, 2164, 2169, 2175, 2182, 2380, 2403, 2526, 2544, 2712, 2780

## E

`\end` .. 1167, 1185, 1584, 1619, 2432, 2454, 2569, 2586, 2801, 2817, 3210, 3215  
`\endlist` ..... 26  
`\endlist` ..... 209

`\endlrbox` ..... 3088  
`\endminipage` ..... 27  
`\endminipage` ..... 215  
enumext ..... 4, 2279  
enumext internal commands:

`\__enumext_add_pre_parsep:` ... 39, 803, 805, 805  
`\__enumext_after_args_exec:` . 37, 697, 709, 2291  
`\__enumext_after_args_exec_v:` . 37, 38, 713, 725, 2492  
`\__enumext_after_args_exec_vii:` ... 729, 753  
`\__enumext_after_args_exec_viii:` ..... 757  
`\__enumext_after_env:n` ..... 73  
`\__enumext_after_env:nn` .. 73, 87, 177, 177, 2469, 2806, 3123  
`\__enumext_after_hyperref:` ... 27, 217, 219, 219  
`\__enumext_after_list:` . 73, 83, 2296, 2444, 2444  
`\l__enumext_after_list_args_v_tl` ..... 727  
`\l__enumext_after_list_args_vii_tl` 755, 3079  
`\l__enumext_after_list_args_viii_tl` ... 759  
`\__enumext_after_list_v:` .. 76, 2497, 2576, 2576  
`\__enumext_after_list_vii:` ... 2846, 2911, 2911  
`\__enumext_after_star_env:nn` ..... 81  
`\__enumext_after_stop_list:` ... 37, 38, 697, 705, 2458  
`\__enumext_after_stop_list_v:` 37, 713, 721, 2591  
`\l__enumext_after_stop_list_v_tl` ..... 723  
`\__enumext_after_stop_list_vii:` 729, 745, 2915  
`\l__enumext_after_stop_list_vii_tl` ... 747  
`\__enumext_after_stop_list_viii:` ..... 749  
`\l__enumext_after_stop_list_viii_tl` ... 751  
`\l__enumext_align_label_vii_str` .. 3071, 3075  
`\l__enumext_align_label_X_str` ..... 148  
`\c__enumext_all_envs_clist` .. 168, 339, 552, 615, 681, 696, 777, 1213  
`\c__enumext_all_families_seq` .. 90, 3239, 3248, 3270  
`\__enumext_anskey_wrapper:n` ..... 1471, 1826  
`\__enumext_at_begin_document:n` .. 26, 173, 173, 206, 212  
`\__enumext_before_args_exec:` 36, 697, 697, 2377  
`\__enumext_before_args_exec_v:` .. 37, 713, 713, 2525  
`\__enumext_before_args_exec_vii:` .. 729, 729, 2908  
`\__enumext_before_args_exec_viii:` .... 733  
`\__enumext_before_keys_exec:` 37, 697, 701, 2289  
`\__enumext_before_keys_exec_v:` .. 37, 713, 717, 2490  
`\__enumext_before_keys_exec_vii` ..... 729  
`\__enumext_before_keys_exec_vii:` 38, 737, 2834  
`\__enumext_before_keys_exec_viii:` .. 38, 741  
`\__enumext_before_list:` ... 71, 2283, 2374, 2374  
`\__enumext_before_list_v:` . 75, 2485, 2522, 2522  
`\__enumext_before_list_vii:` 83, 2829, 2905, 2905  
`\l__enumext_before_no_starred_key_v_tl` 719  
`\l__enumext_before_no_starred_key_vii-  
tl` ..... 739  
`\l__enumext_before_no_starred_key_viii-  
tl` ..... 743  
`\l__enumext_before_starred_key_v_tl` ... 715  
`\l__enumext_before_starred_key_vii_tl` . 731  
`\l__enumext_before_starred_key_viii_tl` 735  
`\__enumext_calc_hspace:NNNNNNN` 67, 2162, 2162, 2193, 2198, 2244



\\_\_enumext\_check\_ans\_active: . . 52, [1418](#), [1418](#),  
     [2475](#)  
 \\_\_enumext\_check\_ans\_active\_vii: . [1418](#), [1441](#),  
     [3129](#)  
 \l\_\_enumext\_check\_ans\_bool . 64, [111](#), [1332](#), [1363](#),  
     [1408](#), [1663](#), [1930](#), [2033](#), [2064](#), [2438](#), [2924](#), [3033](#)  
 \\_\_enumext\_check\_ans\_count: . 51, 71, [1406](#), [1406](#),  
     [2378](#)  
 \\_\_enumext\_check\_ans\_int:n . . 49, 51, [1334](#), [1374](#),  
     [1374](#)  
 \g\_\_enumext\_check\_ans\_item\_tl . . 62, [111](#), [1929](#),  
     [1937](#), [1941](#)  
 \g\_\_enumext\_check\_ans\_show\_bool 73, [111](#), [2440](#),  
     [2471](#), [2478](#)  
 \g\_\_enumext\_check\_ans\_show\_h\_bool [111](#), [2926](#),  
     [3125](#), [3132](#)  
 \l\_\_enumext\_columns\_sep\_v\_dim 2544, [2546](#), [2554](#)  
 \l\_\_enumext\_columns\_sep\_vii\_dim . . 2712, [2714](#),  
     [2723](#), [2768](#), [2897](#), [3107](#)  
 \l\_\_enumext\_columns\_v\_int 1002, [2542](#), [2550](#), [2562](#),  
     [2567](#)  
 \l\_\_enumext\_columns\_vii\_int . . 2717, [2720](#), [2724](#),  
     [2732](#), [2736](#), [2739](#), [2745](#), [2751](#), [2755](#), [2884](#), [3102](#), [3113](#)  
 \l\_\_enumext\_compare\_items\_ans\_int [111](#), [1420](#),  
     [1426](#), [1443](#), [1452](#)  
 \g\_\_enumext\_count\_item\_all\_int [111](#), [1394](#), [1397](#),  
     [1422](#), [1445](#), [2035](#), [2066](#), [3042](#)  
 \g\_\_enumext\_count\_item\_ans\_int 51, 56, 62, [111](#),  
     [1404](#), [1426](#), [1452](#), [1665](#), [1932](#)  
 \g\_\_enumext\_count\_item\_i\_int . . . . . 1399, [1446](#)  
 \g\_\_enumext\_count\_item\_ii\_int 1400, [1422](#), [1447](#)  
 \g\_\_enumext\_count\_item\_iii\_int 1401, [1423](#), [1448](#)  
 \g\_\_enumext\_count\_item\_iv\_int 1402, [1423](#), [1449](#)  
 \g\_\_enumext\_count\_item\_vii\_int . . . . . 1403  
 \g\_\_enumext\_count\_item\_X\_int . . . . . [111](#)  
 \g\_\_enumext\_count\_level\_i\_int . . . 1435, [1461](#)  
 \g\_\_enumext\_count\_level\_ii\_int . . 1436, [1462](#)  
 \g\_\_enumext\_count\_level\_iii\_int . . 1437, [1463](#)  
 \g\_\_enumext\_count\_level\_iv\_int . . 1438, [1464](#)  
 \g\_\_enumext\_count\_level\_vii\_int . . 1439, [1465](#),  
     [3043](#)  
 \g\_\_enumext\_count\_level\_X\_int . . . . . [111](#)  
 \l\_\_enumext\_counter\_i\_tl . . . . . 26, [278](#)  
 \l\_\_enumext\_counter\_ii\_tl . . . . . 26, [279](#)  
 \l\_\_enumext\_counter\_iii\_tl . . . . . 26, [280](#)  
 \l\_\_enumext\_counter\_iv\_tl . . . . . 26, [281](#)  
 \l\_\_enumext\_counter\_style\_for\_ref\_vii\_  
     tl . . . . . 417, [427](#), [438](#), [440](#)  
 \l\_\_enumext\_counter\_style\_for\_ref\_viii\_  
     tl . . . . . 444, [446](#)  
 \l\_\_enumext\_counter\_style\_for\_ref\_X\_tl [137](#)  
 \c\_\_enumext\_counter\_style\_tl . . . 30, [137](#), [384](#)  
 \g\_\_enumext\_counter\_styles\_tl . 22, 29, [36](#), [289](#),  
     [307](#)  
 \l\_\_enumext\_counter\_v\_tl . . . . . 26, [282](#)  
 \l\_\_enumext\_counter\_vi\_tl . . . . . 26, [283](#)  
 \l\_\_enumext\_counter\_vii\_tl . . . . . 26, [284](#), [414](#)  
 \l\_\_enumext\_counter\_viii\_tl . . . . . 26, [285](#), [424](#)  
 \l\_\_enumext\_current\_widest\_dim 22, [36](#), [313](#), [458](#),  
     [478](#), [494](#), [498](#)  
 \\_\_enumext\_default\_item:n . . . 2029, [2029](#), [2080](#)  
 \\_\_enumext\_define\_counters:Nn 22, [269](#), [269](#), [278](#),  
     [279](#), [280](#), [281](#), [282](#), [283](#), [284](#), [285](#)  
 \\_\_enumext\_endminipage: . 27, [212](#), [215](#), [880](#), [2706](#),  
     [3087](#)  
 \\_\_enumext\_fake\_item: . . . . . 621, [621](#), [2231](#)  
 \l\_\_enumext\_fake\_item\_indent\_v\_dim 640, [645](#)  
 \l\_\_enumext\_fake\_item\_indent\_v\_tl 642, [2088](#),  
     [2092](#), [2100](#)  
 \l\_\_enumext\_fake\_item\_indent\_vii\_dim 652, [657](#)  
 \l\_\_enumext\_fake\_item\_indent\_vii\_tl 654, [3083](#)  
 \l\_\_enumext\_fake\_item\_indent\_viii\_dim . 664,  
     [669](#)  
 \l\_\_enumext\_fake\_item\_indent\_viii\_tl . . 666  
 \l\_\_enumext\_fake\_item\_indent\_X\_tl . . . . . 66  
 \\_\_enumext\_fake\_item\_vii: . . . . 621, [649](#), [2267](#)  
 \\_\_enumext\_fake\_item\_viii: . . . . 621, [661](#), [2272](#)  
 \g\_\_enumext\_footnote\_arg\_seq . [134](#), [2002](#), [2015](#),  
     [2025](#)  
 \g\_\_enumext\_footnote\_int . [134](#), [2009](#), [2012](#), [2014](#),  
     [2016](#)  
 \g\_\_enumext\_footnote\_int\_seq . [134](#), [2003](#), [2016](#),  
     [2021](#), [2024](#)  
 \\_\_enumext\_footnotes\_key\_bool . . . . . 27  
 \l\_\_enumext\_footnotes\_key\_bool 24, 27, 86, [124](#),  
     [227](#), [232](#), [241](#), [3050](#), [3098](#)  
 \\_\_enumext\_footnotetext:nn . . . 1996, [1996](#), [2026](#)  
 \\_\_enumext\_getkeyans:nn . . . 88, [3149](#), [3153](#), [3153](#)  
 \\_\_enumext\_getkeyans\_aux:n . 88, [3137](#), [3140](#), [3140](#)  
 \l\_\_enumext\_hyperref\_bool 24, 27, [124](#), [223](#), [244](#),  
     [261](#), [1729](#), [1917](#), [3027](#)  
 \\_\_enumext\_hypertarget:nn 27, [219](#), [246](#), [250](#), [266](#)  
 \\_\_enumext\_if\_is\_int:n . . . . . 194  
 \\_\_enumext\_if\_is\_int:nTF . . . . . 194, [505](#), [519](#)  
 \l\_\_enumext\_item\_column\_pos\_vii\_int 84, [2739](#),  
     [2745](#), [2751](#), [2755](#), [2762](#), [2955](#), [3102](#), [3105](#)  
 l\_\_enumext\_item\_column\_pos\_X\_int . . . . . [148](#)  
 \g\_\_enumext\_item\_count\_all\_vii\_int 84, [2763](#),  
     [2956](#), [3113](#), [3120](#)  
 \g\_\_enumext\_item\_count\_all\_X\_int . . . . . [148](#)  
 \\_\_enumext\_item\_peek\_args\_vii: 84, [2957](#), [2959](#),  
     [2959](#)  
 \\_\_enumext\_item\_starred: . . 66, [2122](#), [2122](#), [2140](#)  
 \l\_\_enumext\_item\_starred\_vii\_bool 2974, [2990](#),  
     [3054](#)  
 \l\_\_enumext\_item\_starred\_X\_bool . . . . . [148](#)  
 \\_\_enumext\_item\_std:w 26, 64, 65, 79, [206](#), [210](#), [2039](#),  
     [2045](#), [2070](#), [2088](#), [2092](#), [2100](#), [2687](#)  
 \g\_\_enumext\_item\_symbol\_aux\_vii\_tl 2998, [3056](#),  
     [3059](#), [3063](#), [3065](#)  
 \g\_\_enumext\_item\_symbol\_aux\_X\_tl . . . . . [148](#)  
 \l\_\_enumext\_item\_symbol\_sep\_vii\_dim . . 3007,  
     [3015](#), [3062](#), [3064](#)  
 \g\_\_enumext\_item\_symbol\_tl 22, 64, [31](#), [2054](#), [2128](#),  
     [2145](#)  
 \l\_\_enumext\_item\_symbol\_vii\_tl . . . . . 3059  
 \l\_\_enumext\_item\_text\_vii\_box 3049, [3090](#), [3097](#)  
 \l\_\_enumext\_item\_text\_X\_box . . . . . [148](#)  
 \l\_\_enumext\_item\_width\_vii\_dim . . 2721, [2766](#),  
     [2774](#), [2775](#)  
 \l\_\_enumext\_item\_width\_X\_dim . . . . . [148](#)  
 \l\_\_enumext\_itemindent\_X\_dim . . . . . 40  
 \l\_\_enumext\_itemsep\_vii\_skip . . . . . 3119  
 \l\_\_enumext\_joined\_item\_aux\_vii\_int . . 2760,  
     [2761](#), [2762](#), [2763](#), [2769](#)  
 \l\_\_enumext\_joined\_item\_aux\_X\_int . . . . . [148](#)  
 \\_\_enumext\_joined\_item\_vii:w . . 84, [2962](#), [2963](#),  
     [2965](#), [2965](#)  
 \l\_\_enumext\_joined\_item\_vii\_int . . 2731, [2732](#),  
     [2735](#), [2737](#), [2743](#), [2748](#), [2753](#), [2758](#), [2760](#), [2766](#)

```

\l__enumext_joined_item_X_int . . . . . 148
\l__enumext_joined_width_vii_dim . 2764, 2771,
    2774, 3080, 3092
\l__enumext_joined_width_X_dim . . . . . 148
\__enumext_keyans_addto_prop:n 60, 1850, 1850,
    2102, 2619
\__enumext_keyans_addto_seq:n . 61, 1901, 1901,
    2104, 2621
\__enumext_keyans_anspic_code:nnn . 77, 2610,
    2613, 2613
\__enumext_keyans_check_ans:nn 62, 1935, 1935,
    2495, 2651
\__enumext_keyans_default_item:n . 65, 2083,
    2083, 2118
\l__enumext_keyans_env_bool 16, 2353, 2506, 2590
\__enumext_keyans_fake_item: . 621, 637, 2221
\__enumext_keyans_internal_ref: 60, 1866, 1866,
    2103, 2620
\l__enumext_keyans_level_h_int . . . . . 16
\l__enumext_keyans_level_int . . 16, 1155, 1652,
    1965, 2505, 2509, 2604
\__enumext_keyans_make_label: 29, 67, 2148, 2148,
    2220
\__enumext_keyans_mini_addvspace: 44, 75, 1059,
    1059, 2534
\__enumext_keyans_mini_right_cmd:n 47, 1157,
    1180, 1180
\__enumext_keyans_mini_set_vskip: 43, 997, 997,
    1061
\__enumext_keyans_multi_addvspace: . 75, 846,
    857, 2559
\__enumext_keyans_multi_set_vskip: . 40, 846,
    846, 859
\__enumext_keyans_multicols_start: 75, 2538,
    2540, 2540
\__enumext_keyans_multicols_stop: . 76, 1184,
    2565, 2565, 2589
\__enumext_keyans_parse_keys:n 2484, 2518, 2518
\l__enumext_keyans_pic_above_int . 106, 2697,
    2698, 2700
\l__enumext_keyans_pic_above_skip . 79, 106,
    2642, 2681
\__enumext_keyans_pic_arg_two: 78, 2640, 2669,
    2669
\l__enumext_keyans_pic_below_int . 106, 2697,
    2698, 2701
\l__enumext_keyans_pic_body_seq . 77-79, 106,
    2608, 2647, 2705
\__enumext_keyans_pic_do:n 79, 2647, 2649, 2689,
    2689, 2693
\l__enumext_keyans_pic_level_int . 16, 1147,
    1656, 1853, 1876, 1904, 2658, 2659
\__enumext_keyans_pic_row:n 79, 2691, 2694, 2694
\__enumext_keyans_pic_safe_exec: . 78, 2636,
    2656, 2656
\__enumext_keyans_pic_skip_abs:N . 78, 2664,
    2664, 2680
\l__enumext_keyans_pic_width_dim . 106, 2696,
    2703
\__enumext_keyans_redefine_item: . 66, 2106,
    2106, 2219
\__enumext_keyans_safe_exec: . 2483, 2499, 2499
\__enumext_keyans_show_left:n . 65, 1943, 1943,
    2098, 2627
\__enumext_keyans_starred_item:n . 65, 2095,
    2095, 2114
\l__enumext_keyans_tmpa_tl . 23, 78, 2097, 2101
\l__enumext_label_copy_i_tl . 1783, 1879, 1883
\l__enumext_label_copy_v_tl . . . . . 1883
\l__enumext_label_copy_vi_tl . . . . . 1879
\l__enumext_label_copy_vii_tl 1758, 1769, 1800
\l__enumext_label_copy_X_tl . . . . . 126
\l__enumext_label_fill_left_v_tl . . . . 2152
\l__enumext_label_fill_left_X_tl . . . . 66
\l__enumext_label_fill_right_v_tl . . . 2159
\l__enumext_label_fill_right_X_tl . . . . 66
\l__enumext_label_font_style_v_tl 2153, 2631
\l__enumext_label_font_style_vii_tl . . 3068
\l__enumext_label_i_tl . . . . . 450
\l__enumext_label_ii_tl . . . . . 450
\l__enumext_label_iii_tl . . . . . 450
\l__enumext_label_iv_tl . . . . . 450
\__enumext_label_style:Nnn 22, 29, 302, 302, 317,
    455, 475, 491, 495
\l__enumext_label_v_tl . 60, 61, 488, 1858, 1909,
    1947, 1954, 1970, 1977, 2097, 2101, 2487, 2626, 2628
\l__enumext_label_vi_tl . 60, 61, 488, 1855, 1906,
    2626, 2628, 2632
\l__enumext_label_vii_tl . 470, 2985, 3010, 3017
\l__enumext_label_viii_tl . . . . . 470
\l__enumext_label_width_by_box . 36, 298, 299
\__enumext_label_width_by_box:Nn 29, 296, 296,
    301, 313, 529
\l__enumext_labelsep_i_dim . . . . . 1951, 1974
\l__enumext_labelsep_v_dim . . . . . 2549
\l__enumext_labelsep_vii_dim . 2716, 2725, 2767,
    3008, 3078, 3094
\l__enumext_labelwidth_i_dim . . . . 1950, 1973
\l__enumext_labelwidth_v_dim . . . . . 2549
\l__enumext_labelwidth_vii_dim . . 2716, 2724,
    2767, 3071, 3075, 3093
\l__enumext_leftmargin_tmp_v_bool . 78, 2671
\l__enumext_leftmargin_tmp_X_bool . . . . 40
\l__enumext_leftmargin_tmp_X_dim . . . . 40
\l__enumext_leftmargin_X_dim . . . . . 40
\__enumext_level: 181, 181, 187, 188, 192, 393, 395,
    396, 404, 406, 624, 628, 632, 699, 703, 707, 711, 795,
    797, 799, 801, 834, 836, 838, 840, 844, 884, 887, 906,
    915, 921, 926, 930, 941, 945, 946, 951, 987, 991, 1164,
    1170, 1217, 1219, 1221, 1224, 1231, 1233, 1235, 1238,
    1412, 1413, 1415, 1560, 1568, 1572, 1576, 1821, 1824,
    1825, 2036, 2038, 2039, 2043, 2044, 2045, 2052, 2054,
    2058, 2059, 2062, 2067, 2069, 2070, 2124, 2127, 2129,
    2136, 2137, 2138, 2141, 2144, 2286, 2288, 2324, 2329,
    2330, 2331, 2333, 2337, 2342, 2343, 2344, 2346, 2357,
    2362, 2369, 2380, 2382, 2385, 2386, 2388, 2393, 2400,
    2403, 2405, 2407, 2408, 2409, 2410, 2413, 2419, 2424,
    2430, 2433, 2435, 2446, 3038, 3039
\__enumext_level_ . . . . . 187
\__enumext_level_end:n . . . . . 185, 190
\l__enumext_level_h_int 16, 412, 436, 1777, 1794,
    2307, 2355, 2850, 2851
\l__enumext_level_int 16, 183, 807, 958, 1151, 1391,
    1754, 1764, 1770, 1776, 1784, 1792, 1799, 2234, 2300,
    2301, 2316, 2360, 2415, 2473, 2513, 2600, 2859, 2936,
    2946, 3127
\__enumext_level_set:n . . . . . 185, 185
\__enumext_list_arg_two_i: . . . . . 2194

```

\\_\_enumext\_list\_arg\_two\_ii: ..... 2194  
 \\_\_enumext\_list\_arg\_two\_iii: ..... 2194  
 \\_\_enumext\_list\_arg\_two\_iv: ..... 2194  
 \\_\_enumext\_list\_arg\_two\_v: . 66, 2194, 2489, 2672  
 \\_\_enumext\_list\_arg\_two\_vii: ..... 2240, 2833  
 \\_\_enumext\_list\_arg\_two\_viii: ..... 2240  
 \l\_\_enumext\_listoffset\_v\_dim ..... 2551  
 \l\_\_enumext\_listparindent\_vii\_dim .... 3081  
 \\_\_enumext\_make\_label: 29, 64, 66, 2132, 2132, 2229  
 \l\_\_enumext\_mark\_answer\_sym\_tl ... 56, 62, 101, 1474, 1630, 1836, 1958  
 \l\_\_enumext\_mark\_position\_str 101, 1478, 1479, 1501, 1502, 1628  
 \l\_\_enumext\_mark\_ref\_sym\_tl .. 101, 1488, 1734, 1925  
 \\_\_enumext\_mini\_addvspace: 43, 72, 980, 980, 2390  
 \\_\_enumext\_mini\_addvspace\_vii: 45, 1133, 1133, 2792  
 \\_\_enumext\_mini\_addvspace\_viii: 45, 1133, 1139  
 \\_\_enumext\_mini\_env\* ..... 874  
 \\_\_enumext\_mini\_right\_cmd:n . 46, 47, 1159, 1161, 1161  
 \\_\_enumext\_mini\_set\_vskip: ... 41, 881, 881, 982  
 \\_\_enumext\_mini\_set\_vskip\_vii: 45, 1076, 1076, 1135  
 \\_\_enumext\_mini\_set\_vskip\_viii: 45, 1076, 1098, 1141  
 \\_\_enumext\_minipage:w 27, 212, 214, 876, 2703, 3080  
 \l\_\_enumext\_minipage\_active\_v\_bool .. 75, 76, 2532, 2557, 2570, 2578  
 \g\_\_enumext\_minipage\_active\_vii\_bool ... 81, 2803, 2808, 2820  
 \l\_\_enumext\_minipage\_active\_vii\_bool . 2788, 2799  
 \g\_\_enumext\_minipage\_active\_X\_bool ... 148  
 \l\_\_enumext\_minipage\_active\_X\_bool .... 54  
 \g\_\_enumext\_minipage\_after\_skip 54, 1080, 1092, 2818  
 \l\_\_enumext\_minipage\_after\_skip 41, 42, 73, 76, 54, 897, 912, 932, 948, 963, 969, 975, 989, 999, 1008, 1011, 1023, 1041, 1052, 1068, 1100, 1113, 1127, 2455, 2587  
 \g\_\_enumext\_minipage\_center\_vii\_bool . 2812, 2821  
 \g\_\_enumext\_minipage\_center\_X\_bool ... 148  
 \l\_\_enumext\_minipage\_hsep\_v\_dim ... 75, 2530  
 \l\_\_enumext\_minipage\_hsep\_vii\_dim .... 2786  
 \l\_\_enumext\_minipage\_left\_skip 41, 75, 54, 889, 904, 923, 938, 985, 995, 1000, 1006, 1015, 1032, 1044, 1064, 1074, 1078, 1083, 1087, 1101, 1105, 1119, 1137, 1143  
 \l\_\_enumext\_minipage\_left\_v\_dim 75, 2528, 2536  
 \l\_\_enumext\_minipage\_left\_vii\_dim 2782, 2794  
 \l\_\_enumext\_minipage\_left\_X\_dim ..... 54  
 \g\_\_enumext\_minipage\_right\_skip 54, 1079, 1084, 1088, 2811  
 \l\_\_enumext\_minipage\_right\_skip .. 41, 54, 893, 908, 928, 943, 1001, 1007, 1019, 1037, 1048, 1102, 1109, 1123, 1171, 1188  
 \l\_\_enumext\_minipage\_right\_v\_dim .. 75, 1182, 1187, 2526, 2530  
 \g\_\_enumext\_minipage\_right\_vii\_dim 81, 2790, 2810, 2823  
 \l\_\_enumext\_minipage\_right\_vii\_dim 81, 2780, 2785, 2791  
 \g\_\_enumext\_minipage\_right\_X\_dim .... 148  
 \g\_\_enumext\_minipage\_right\_X\_skip .... 148  
 \g\_\_enumext\_minipage\_stat\_int . 72, 75, 54, 1176, 1193, 2389, 2448, 2453, 2533, 2580, 2585  
 \g\_\_enumext\_miniright\_code\_vii\_tl . 81, 2816, 2822  
 \g\_\_enumext\_miniright\_code\_X\_tl ..... 148  
 \\_\_enumext\_multi\_addvspace: ... 40, 72, 829, 829, 2421  
 \\_\_enumext\_multi\_set\_vskip: .. 39, 793, 793, 831  
 \l\_\_enumext\_multicols\_above\_ii\_skip ... 812  
 \l\_\_enumext\_multicols\_above\_iii\_skip .. 818  
 \l\_\_enumext\_multicols\_above\_iv\_skip ... 824  
 \l\_\_enumext\_multicols\_above\_v\_skip 848, 862, 872  
 \l\_\_enumext\_multicols\_above\_X\_skip .... 48  
 \l\_\_enumext\_multicols\_below\_v\_skip 852, 866, 2572  
 \l\_\_enumext\_multicols\_below\_X\_skip .... 48  
 \\_\_enumext\_multicols\_start: 72, 2395, 2397, 2397  
 \\_\_enumext\_multicols\_stop: 73, 1166, 2427, 2427, 2457  
 \\_\_enumext\_newlabel:nn .. 24, 27, 59, 61, 254, 254, 1810, 1892  
 \l\_\_enumext\_newlabel\_arg\_one\_tl 24, 27, 59, 61, 126, 1733, 1803, 1811, 1885, 1893, 1923  
 \l\_\_enumext\_newlabel\_arg\_two\_tl 24, 27, 58, 60, 126, 1757, 1767, 1781, 1797, 1812, 1878, 1882, 1894  
 \\_\_enumext\_parse\_keys:n ..... 2282, 2313, 2313  
 \\_\_enumext\_parse\_keys\_vii:n .. 2828, 2865, 2865  
 \\_\_enumext\_parse\_store\_keys:n . 70, 2319, 2322, 2322  
 \\_\_enumext\_parse\_store\_keys\_vii:n 82, 83, 2872, 2876, 2876  
 \l\_\_enumext\_parsep\_i\_skip . 810, 812, 961, 1009  
 \l\_\_enumext\_parsep\_ii\_skip ..... 816, 818, 967  
 \l\_\_enumext\_parsep\_iii\_skip .... 822, 824, 973  
 \l\_\_enumext\_parsep\_vii\_skip ..... 3082  
 \l\_\_enumext\_partopsep\_v\_skip .. 864, 868, 1035, 1039, 1046, 1050, 1066, 1070  
 \l\_\_enumext\_partopsep\_viii\_skip ..... 1111  
 \\_\_enumext\_phantomsection: 27, 219, 247, 251, 267  
 \\_\_enumext\_print\_footnote: ... 1996, 2019, 3100  
 \\_\_enumext\_print\_keyans\_box:NN 56, 1622, 1622, 1635, 1823, 1949, 1972  
 \l\_\_enumext\_print\_keyans\_i\_tl .... 3163, 3192  
 \l\_\_enumext\_print\_keyans\_ii\_tl ... 3168, 3193  
 \l\_\_enumext\_print\_keyans\_iii\_tl .. 3173, 3194  
 \l\_\_enumext\_print\_keyans\_iv\_tl ... 3178, 3195  
 \l\_\_enumext\_print\_keyans\_vii\_tl .. 3183, 3196  
 \l\_\_enumext\_print\_keyans\_X\_tl ..... 90  
 \\_\_enumext\_printkeyans:nnn . 89, 3197, 3200, 3200  
 \\_\_enumext\_redefine\_item: . 65, 2072, 2072, 2228  
 \l\_\_enumext\_ref\_aux\_tl 137, 393, 395, 398, 414, 416, 419, 424, 426, 429  
 \l\_\_enumext\_ref\_key\_arg\_tl .. 137, 387, 392, 399, 411, 420, 430  
 \\_\_enumext\_regex\_label\_ref\_key: .. 30, 31, 382, 382, 394, 415, 425  
 \\_\_enumext\_register\_counter\_style:Nn .. 286, 286, 291, 292, 293, 294, 295  
 \\_\_enumext\_remove\_extra\_parsep\_vii: .. 2843, 3109, 3109

```

\__enumext_renew_footnote: ... 1996, 2000, 3052
\l__enumext_resume_bool ... 22, 31, 1346, 2211
\__enumext_resume_counter: . 50, 1312, 1337, 1337
\__enumext_resume_counter_star: ..... 1314
\__enumext_resume_counter_vii: 50, 1321, 1337,
    1348
\g__enumext_resume_int 22, 73, 31, 1341, 1352, 2212,
    2461
\l__enumext_resume_vii_bool ... 31, 1357, 2258
\g__enumext_resume_vii_int .. 83, 31, 2259, 2917
\__enumext_safe_exec: ..... 2281, 2298, 2298
\__enumext_safe_exec_vii: ... 2827, 2848, 2848
\__enumext_set_error:nn ..... 3277, 3287, 3289
\__enumext_set_label_ref:n ... 30, 390, 390, 462
\__enumext_set_label_ref_h:n . 31, 409, 409, 482
\__enumext_set_parse:n ..... 3260, 3277, 3277
\l__enumext_setkey_tmpa_int ... 85, 3253, 3257
\l__enumext_setkey_tmpa_seq 85, 3251, 3261, 3267,
    3269, 3271, 3284
\l__enumext_setkey_tmpa_tl ... 85, 3259, 3263
\l__enumext_setkey_tmpb_seq 85, 3252, 3255, 3259,
    3260
\l__enumext_setkey_tmpb_tl 85, 3279, 3281, 3282
\l__enumext_show_answer_bool . 101, 1482, 1486,
    1505, 1509, 1830, 1945, 2623
\__enumext_show_length:nnn .. 36, 200, 200, 3321,
    3322, 3323, 3324, 3325, 3326, 3327, 3328, 3329, 3330,
    3336, 3337, 3338, 3339, 3340, 3341, 3342, 3343, 3344,
    3345
\l__enumext_show_position_bool 101, 1483, 1485,
    1506, 1508, 1834, 1956, 2624
\g__enumext_standar_bool ..... 16, 2310
\l__enumext_standar_bool . 16, 1762, 1775, 1791,
    2303, 2460, 2858
\__enumext_standard_item_vii:w .. 84, 85, 2970,
    2972, 2972
\g__enumext_starred_bool . 82, 83, 16, 1390, 1753,
    1763, 1793, 2438, 2862, 2924, 2929
\l__enumext_starred_bool . 82, 83, 16, 1686, 1694,
    1778, 1819, 2306, 2855, 2930
\__enumext_starred_columns_set_vii: .. 2710,
    2710, 2836
\__enumext_starred_item:nn ... 2048, 2048, 2078
\__enumext_starred_item_vii:w 84, 85, 2969, 2988,
    2988
\__enumext_starred_item_vii_aux_i:w .. 2988,
    2993, 2996
\__enumext_starred_item_vii_aux_ii:w . 2988,
    2994, 2999, 3001
\__enumext_starred_item_vii_aux_iii:w 2988,
    3004, 3013
\__enumext_starred_joined_item_vii:n . 80, 84,
    2729, 2729, 2967
\__enumext_start_from:NNn 33, 503, 503, 516, 538
\__enumext_start_item_tmp_vii: 82, 2839, 2952,
    2952
\__enumext_start_item_vii:w . 85, 86, 2980, 2985,
    3010, 3017, 3019, 3019
\__enumext_start_list:nn 26, 68, 78, 206, 208, 2285,
    2486, 2637, 2831
\__enumext_start_mini_vii: . 83, 2778, 2778, 2909
\__enumext_start_store_level: . 71, 2284, 2351,
    2351
\__enumext_start_store_level_vii: . 84, 2830,
    2932, 2932
\l__enumext_start_X_int ..... 66, 533
\__enumext_stop_item_tmp_vii: . 82, 84, 86, 2838,
    2842, 2954, 3021
\__enumext_stop_item_vii: 86, 87, 3021, 3085, 3085
\__enumext_stop_list: .. 26, 206, 209, 2294, 2496,
    2650, 2844
\__enumext_stop_mini_vii: 81, 83, 2797, 2797, 2914
\__enumext_stop_store_level: .. 71, 2295, 2351,
    2367
\__enumext_stop_store_level_vii: .. 84, 2845,
    2932, 2942
\l__enumext_store_active_bool 23, 49, 70, 82, 78,
    1327, 1339, 1350, 1648, 2317, 2353, 2501, 2508, 2596,
    2654, 2870, 2934, 2944
\__enumext_store_addto_prop:n 54, 60, 1539, 1539,
    1548, 1673, 1864
\__enumext_store_addto_seq:n 54, 61, 1549, 1549,
    1555, 1562, 1576, 1584, 1593, 1611, 1619, 1737, 1928
\l__enumext_store_ans_bool ... 1365, 1410, 1558,
    1582, 1589, 1617, 1661, 3036
\l__enumext_store_anskey_arg_tl .. 23, 57, 78,
    1679, 1688, 1690, 1696, 1704, 1707, 1717, 1722, 1725,
    1731, 1737
\__enumext_store_anskey_code:nnnn 56, 57, 1667,
    1671, 1671
\__enumext_store_anskey_show_left:n 59, 1678,
    1828, 1828
\__enumext_store_anskey_show_wrap:n 59, 1816,
    1816, 1832, 1847
\l__enumext_store_columns_break_bool . 1642,
    1685
\l__enumext_store_columns_join_int 78, 1693,
    1698
\l__enumext_store_columns_sep_vii_bool 2891
\l__enumext_store_columns_sep_vii_dim 2896,
    2900
\l__enumext_store_columns_sep_X_bool ... 90
\__enumext_store_columns_sep_X_dim .... 90
\l__enumext_store_columns_vii_bool ... 2878
\l__enumext_store_columns_vii_int 2883, 2887
\l__enumext_store_columns_X_bool ..... 90
\l__enumext_store_columns_X_int ..... 90
\__enumext_store_internal_ref: 58, 1676, 1739,
    1739
\l__enumext_store_item_symbol_sep_dim 1640,
    1714, 1719
\l__enumext_store_item_symbol_tl . 1638, 1705,
    1709
\l__enumext_store_keyans_label_tl 23, 60-62,
    78, 1852, 1855, 1858, 1864, 1903, 1906, 1909,
    1913, 1919, 1928, 1929
\__enumext_store_level_close: . 54, 1556, 1580,
    2371
\__enumext_store_level_close_vii: 1587, 1615,
    2948
\__enumext_store_level_open: .. 53, 54, 70, 1556,
    1556, 2358, 2363
\__enumext_store_level_open_vii: .. 83, 1587,
    1587, 2938
\g__enumext_store_name_tl 23, 73, 78, 1429, 1433,
    1455, 1459, 2441, 2479, 2927, 3133
\l__enumext_store_name_tl 23, 49, 78, 1326, 1343,

```



1354, 1541, 1542, 1543, 1545, 1551, 1552, 1553, 1805, 1806, 1842, 1887, 1888, 1964, 2441, 2462, 2465, 2918, 2921, 2927	\l__enumext_vspace_below_viii_skip 1299, 1303, 1305
\l__enumext_store_opt_vii_tl . 1591, 1601, 1607, 1611, 2885, 2898	\__enumext_widest_from:nnn . . 33, 517, 517, 532, 544
\l__enumext_store_opt_X_tl . . . . . 90	\g__enumext_widest_label_tl 22, 29, 36, 306, 310, 314
\l__enumext_store_ref_key_bool 57, 1491, 1674, 1728, 1896, 1916	\l__enumext_wrap_label_opt_v_bool . . . . 2091
\l__enumext_store_upper_level_X_bool . . . 90	\l__enumext_wrap_label_opt_vii_bool 85, 2979
\l__enumext_store_write_aux_file_tl 24, 59, 61, 126, 1808, 1814, 1890, 1898	\l__enumext_wrap_label_opt_X_bool . . . . 66
\__enumext_storing_set:n 49, 50, 1310, 1319, 1324, 1324	\l__enumext_wrap_label_v_bool 2087, 2091, 2099, 2154
\l__enumext_the_counter_vii_tl . . . . . 416	\l__enumext_wrap_label_vii_bool 85, 2978, 2983, 2991, 3069
\l__enumext_the_counter_viii_tl . . . . . 426	\l__enumext_wrap_label_X_bool . . . . . 66
\l__enumext_the_counter_X_tl . . . . . 137	\__enumext_wrapper_label_v:n . . . . 2156, 2632
\__enumext_tmp:n 26, 30, 40, 47, 48, 53, 60, 65, 66, 77, 90, 100, 118, 123, 129, 133, 141, 147, 148, 167, 616, 620, 1359, 1373, 1467, 1495, 1496, 1513, 1741, 1748, 1749, 1770, 1784, 1787, 1799, 1868, 1875, 2194, 2239, 2240, 2278	\__enumext_wrapper_label_vii:n . . . . . 3072
\__enumext_tmp:nn 318, 339, 340, 368, 369, 381, 533, 552, 597, 615, 673, 681, 682, 696, 761, 777, 778, 792, 1197, 1213, 1514, 1538, 1980, 1995	\__enumext_zero_parsep: . . . . 42, 901, 956, 956
\__enumext_tmp:nnn 450, 466, 467, 468, 469, 470, 486, 487	enumext* . . . . . 4, 2825
\__enumext_tmp:nnnnn 553, 578, 581, 584, 586, 588, 591, 594	enumXi . . . . . 278
\__enumext_tmp:w . . . . . 3146, 3149	enumXii . . . . . 278
\l__enumext_tmpa_vii_int . . . . . 2720, 2723	enumXiii . . . . . 278
\l__enumext_tmpa_X_int . . . . . 148	enumXiv . . . . . 278
\l__enumext_topsep_v_skip 850, 854, 1004, 1017, 1025, 1030, 1050, 1054, 2653, 2684	enumXv . . . . . 278
\l__enumext_topsep_vii_skip . . 1081, 1090, 1094	enumXvi . . . . . 278
\l__enumext_topsep_viii_skip . 1103, 1125, 1129	enumXvii . . . . . 278
\__enumext_use_key_ref: . . . 31, 402, 402, 2230	enumXviii . . . . . 278
\__enumext_use_key_ref_h: . . 31, 434, 434, 2264	Environments provide by enumext:
\l__enumext_vspace_a_star_v_bool . . . . 1246	enumext* 21–23, 25, 26, 28, 30–32, 35, 36, 38, 45, 48–51, 53–58, 63, 71, 82–84, 86–88, 91, 93
\l__enumext_vspace_a_star_vii_bool . . . 1268	enumext 21–23, 25, 26, 28, 29, 31, 32, 34–44, 46–51, 53–58, 63–68, 70, 71, 73, 74, 78, 79, 81, 84, 88, 91, 92
\l__enumext_vspace_a_star_viii_bool . . . 1279	keyans* 21–23, 25, 26, 28, 30–32, 35, 36, 38, 45, 48–50, 53, 54, 63, 91, 93
\l__enumext_vspace_a_star_X_bool . . . . 66	keyanspic 21–24, 28, 29, 32, 46, 49, 50, 54, 60–62, 76–79, 92
\__enumext_vspace_above: . . 47, 1214, 1214, 2376	keyans 21–24, 26, 28, 29, 32, 34–38, 40, 43, 44, 46–50, 53, 54, 60–62, 66–68, 74, 76–78, 81, 91, 92
\__enumext_vspace_above_v: . 48, 1242, 1242, 2524	Environments:
\l__enumext_vspace_above_v_skip . . 1244, 1248, 1250	enumext* . . . . . 69
\__enumext_vspace_above_vii: . . 48, 1264, 1264, 2907	keyans* . . . . . 69
\l__enumext_vspace_above_vii_skip 1266, 1270, 1272	list . . . . . 25, 26, 67, 68, 70
\__enumext_vspace_above_viii: . 48, 1264, 1275	lrbox . . . . . 79, 86, 87
\l__enumext_vspace_above_viii_skip 1277, 1281, 1283	minipage . . . . . 25–27, 38, 41, 76, 78, 79, 86, 87
\l__enumext_vspace_b_star_v_bool . . . . 1257	multicols . . . . . 39–41, 46, 72, 73, 75, 76
\l__enumext_vspace_b_star_vii_bool . . . 1290	exp commands:
\l__enumext_vspace_b_star_viii_bool . . . 1301	\exp_after:wN . . . . . 3149
\l__enumext_vspace_b_star_X_bool . . . . 66	\exp_args:Ne . . . . . 2315, 3137
\__enumext_vspace_below: . . 48, 1228, 1228, 2459	\exp_not:N 145, 309, 398, 419, 429, 630, 644, 645, 656, 657, 668, 669, 1733, 1839, 1840, 1921, 1961, 1962, 3146
\__enumext_vspace_below_v: . 48, 1253, 1253, 2592	\exp_not:n 398, 399, 419, 420, 429, 430, 631, 1522, 1529, 1698, 1709, 1719, 1733, 1734, 1811, 1893, 1923, 1925, 2333, 2346, 2887, 2900
\l__enumext_vspace_below_v_skip . . 1255, 1259, 1261	F
\__enumext_vspace_below_vii: . . 49, 1286, 1286, 2916	\fbox . . . . . 1472
\l__enumext_vspace_below_vii_skip 1288, 1292, 1294	file commands:
\__enumext_vspace_below_viii: . 49, 1286, 1297	\file_input_stop: . . . . . 3438
	first . . . . . 682
	font . . . . . 318
	\footnote . . . . . 63
	\footnote . . . . . 63, 2004
	\footnotemark . . . . . 2014
	\footnotesize . . . . . 1840, 1962
	\footnotetext . . . . . 1998

G

\getkeyans ..... 13, 88, 3135

group commands:

  \group\_begin: 1660, 1838, 1960, 3048, 3067, 3157, 3191

  \group\_end: 1669, 1845, 1968, 3077, 3089, 3159, 3198

H

\hbadness ..... 3096

hbox commands:

  \hbox\_set:Nn ..... 298

\hfill ... 348, 352, 357, 358, 1168, 1186, 1733, 1921, 2802

hook commands:

  \hook\_gput\_code:nnn ..... 175, 179, 217

  \hook\_gset\_rule:nnnn ..... 218

\hspace ..... 3107

\hyperlink ..... 58, 61

\hyperlink ..... 1733, 1921

\hypertarget ..... 27

\hypertarget ..... 246

I

\IfHyperBoolean ..... 224

\IfPackageLoadedTF ..... 4, 221, 234

\ignorespaces ..... 633

int commands:

  \int\_add:Nn ..... 2762

  \int\_case:nn ..... 807, 958

  \int\_compare:nNnTF 412, 436, 883, 1002, 1147, 1151, 1155, 1425, 1451, 1652, 1656, 1853, 1876, 1904, 2301, 2355, 2360, 2399, 2415, 2429, 2448, 2473, 2509, 2513, 2542, 2567, 2580, 2600, 2604, 2659, 2732, 2742, 2758, 2851, 2936, 2946, 3102, 3111, 3127, 3257

  \int\_compare\_p:nNn .. 1391, 1754, 1764, 1776, 1777, 1792, 1794, 2307, 2859

  \int\_decr:N ..... 2761

  \int\_eval:n 1545, 1806, 1840, 1888, 1962, 2212, 2215, 2259, 2262, 2750

  \int\_from\_alph:n ..... 511, 525

  \int\_from\_roman:n ..... 513, 527

  \int\_gadd:Nn ..... 1412, 2763, 3038

  \int\_gincr:N 1415, 1665, 1932, 2035, 2036, 2066, 2067, 2389, 2533, 2956, 3042, 3043

  \int\_gset:Nn ..... 1341, 1352, 2012

  \int\_gset\_eq:NN 1394, 1397, 1399, 1400, 1401, 1402, 1403, 1404, 2009, 2461, 2464, 2917, 2920

  \int\_gzero:N 1176, 1193, 1435, 1436, 1437, 1438, 1439, 1461, 1462, 1463, 1464, 1465, 2453, 2585, 3120

  \int\_if\_exist:NTF 1328, 1376, 1378, 1380, 1382, 1384, 1386, 2462, 2918

  \int\_incr:N ..... 2300, 2505, 2658, 2850, 2955

  \int\_mod:nn ..... 3113

  \int\_new:N 16, 17, 18, 19, 20, 32, 34, 54, 70, 82, 87, 95, 108, 109, 115, 116, 117, 120, 121, 134, 151, 152, 153, 154, 155, 1330, 1377, 1379, 1381, 1383, 1385, 1387

  \int\_set:Nn 507, 511, 513, 1420, 1443, 1519, 1693, 2697, 2698, 2720, 2731, 2737, 2753, 3096, 3253

  \int\_set\_eq:NN ..... 2328, 2760, 2882

  \int\_step\_function:nnN ..... 1770, 1784, 1799

  \int\_step\_inline:nnn ..... 2699, 3280

  \int\_to\_roman:n ..... 183, 1750, 1788

  \int\_use:N .. 884, 1343, 1354, 1413, 2215, 2234, 2262, 2316, 2400, 2409, 2424, 2430, 2735, 2736, 2748, 3039

  \int\_zero:N ..... 3105

\c\_one\_int ..... 2720, 2739, 2745, 2751, 2755, 2758

\c\_zero\_int 1391, 1754, 1764, 1776, 1777, 1792, 1794, 2307, 2859, 2936, 2946, 3116

\item ..... 26, 37, 38, 55, 64, 76, 77, 79, 82

\item . 64, 65, 84, 86, 210, 1564, 1570, 1595, 1603, 1690, 1906, 1909, 2074, 2108, 2837, 2839

\item\* ..... 5, 11, 2106

item-pos\* ..... 1980

item-sym\* ..... 1980

\itemindent ..... 22, 68

\itemindent ..... 67

itemindent ..... 597

\itemsep ..... 78, 79

\itemsep ..... 2673, 2679

\itemwidth ..... 2727, 2771, 2775

K

keyans ..... 11, 2481

keyans\* ..... 11

keyanspic ..... 12, 2634

Keys for environments provide by enumext:

  above\* ..... 23, 47, 48

  above ..... 23, 47, 48, 71, 75, 83

  after ..... 36-38, 73, 76, 83

  align ..... 23, 30, 66, 86

  before\* ..... 36, 37, 71, 83

  before ..... 36-38, 75

  below\* ..... 23, 47-49

  below ..... 23, 47-49, 73, 76, 83

  check-ans .. 23-25, 49, 51, 56, 62, 64, 65, 71, 73, 87, 92

  columns-sep\* ..... 23, 53, 70, 83

  columns-sep ..... 38, 54, 70, 72, 75, 83

  columns\* ..... 23, 53, 70, 83

  columns ..... 22, 38, 41, 47, 54, 70, 72, 75, 83

  first ..... 36-38, 86

  font ..... 29, 66, 86

  item-pos\* ..... 56, 57, 63

  item-sym\* ..... 22, 56, 57, 63, 64

  item\*-sep ..... 64

  itemindent ..... 23, 35, 66, 86

  itemsep ..... 34, 69

  labelsep ..... 29, 64, 68, 86

  labelwidth ..... 28, 29, 32, 33, 68

  label ..... 22, 28, 29, 32, 33, 79

  lisparindent ..... 69

  list-indent ..... 22, 35, 78

  list-offset ..... 35

  listparindent ..... 35, 86

  mark-ans ..... 24, 53, 59

  mark-pos ..... 53

  mark-ref ..... 24, 53, 58, 60

  mini-env ..... 22, 38, 41, 46, 47, 63, 71, 75, 81, 83

  mini-sep ..... 22, 38, 71, 75

  miniright\* ..... 22, 38

  miniright ..... 22, 38, 45, 81

  minirigth\* ..... 25

  minirigth ..... 25

  no-store ..... 24, 50, 51

  noitemsep ..... 34, 42

  nosep ..... 34, 42

  parindent ..... 69

  parsep ..... 34, 69, 86

  partopsep ..... 34

  ref ..... 25, 30, 31

  resume ..... 22, 49, 50, 68, 73, 83

  rightmargin ..... 35

  save-ans 23, 49, 50, 54, 56, 60, 61, 65, 73, 74, 77, 83, 88

  save-key ..... 24

show-ans	24, 53, 56, 57, 59, 65
show-length	26, 36, 68, 91
show-pos	24, 53, 56, 57, 59, 65
start	23, 26, 33, 34, 68
store-brk	56, 57
store-ref	24, 27, 53, 57, 58, 60, 61, 65
topsep	34
widest	22, 26, 33, 34
wrap-ans	53, 56, 59
wrap-label*	29, 64, 66, 85, 86
wrap-label	29, 66, 85, 86
keys commands:	
\keys_define:nn	320, 342, 371, 452, 472, 488, 535, 555, 599, 618, 675, 684, 763, 780, 1199, 1308, 1317, 1361, 1469, 1498, 1516, 1636, 1982, 3161, 3223
\l_keys_key_str	3306
\keys_set:nn	334, 787, 1204, 1209, 1682, 2315, 2520, 2869, 3225, 3226, 3227, 3228, 3229, 3230, 3231, 3232, 3233, 3234, 3235, 3236, 3274

L

l internal commands:	
\l_enumext_store_internal_ref:	57
label	450, 470, 488
Labels provide by enumext:	
\Alph*	28, 29
\Roman*	28, 29
\alph*	28, 29
\arabic*	28-30
\roman*	28, 29
\labelsep	79
\labelsep	2674, 2677
labelsep	318
\labelwidth	29, 79
\labelwidth	2674, 2675
labelwidth	318
\leftmargin	22, 68
\leftmargin	67, 2674
legacy commands:	
\legacy_if:nTF	3022, 3025
\legacy_if_gset_false:n	877
\legacy_if_set_false:n	3024
\legacy_if_set_true:n	2984, 3009, 3016, 3029
\linewidth	71, 75
\linewidth	2384, 2530, 2696, 2723, 2784
\list	26
\list	208
list-indent	597
list-offset	597
\listparindent	2676
listparindent	597
\lrbox	3049

M

\makebox	79
\makebox	1626, 1628, 2128, 3063, 3071, 3075
\makelabel	64, 66, 67, 79
\makelabel	66, 67, 2134, 2150
\makesavenoteenv	240
mark-ans	1467
mark-pos	1467, 1496
mark-ref	1467
mini-env	761
mini-sep	761
\minipage	27

\minipage	214
\miniright	9, 46, 1145, 2451, 2583
\miniright*	9
mode commands:	
\mode_if_vertical:TF	832, 860, 983, 1062
\mode_leave_vertical:	630, 644, 656, 668, 1595, 1603, 1624, 2126, 3061
msg commands:	
\msg_error:nn	2511, 2515, 2602, 2661, 2853, 3237
\msg_error:nnn	1149, 1153, 1178, 1195, 3151, 3156, 3220, 3290
\msg_error:nnnn	1650, 1654, 1658, 2503, 2598, 2606
\msg_fatal:nn	2302
\msg_fatal:nnn	272
\msg_info:nnn	6
\msg_line_context:	3310, 3315, 3320, 3335, 3358, 3362, 3366, 3371, 3376, 3381, 3386, 3390, 3395, 3400, 3404, 3409, 3413, 3418, 3423, 3428, 3432, 3436
\msg_new:nnn	3291, 3295, 3300, 3304, 3308, 3313, 3318, 3333, 3348, 3355, 3360, 3364, 3369, 3374, 3379, 3384, 3388, 3393, 3398, 3402, 3407, 3411, 3416, 3421, 3426, 3430, 3434
\msg_term:nnn	1428, 1454
\msg_term:nnnn	2224, 2234, 2269, 2274
\msg_warning:nn	2450, 2582
\msg_warning:nnn	1432, 1458
\msg_warning:nnnn	1939, 2166, 2171, 2734, 2747
\multicolsep	72, 75
\multicolsep	2414, 2555

N

\NeedsTeXFormat	3
\newcounter	275
\NewDocumentCommand	1145, 1646, 2594, 3135, 3189, 3244
\NewDocumentEnvironment	2279, 2481, 2634, 2825
\newlabel	28
\newlabel	258
no-store	1359
\noindent	82
\noindent	2391, 2535, 2793, 2838, 3104
\nointerlineskip	2391, 2535, 2793
noitemsep	553
\nopagebreak	843, 871, 994, 1073, 1136, 1142
\normalfont	1839, 1961
nosep	553

P

Packages:	
enumext	21, 49, 67, 77, 91
enumitem	28
expl3	79
footnotehyper	27
hyperref	24, 25, 27, 28, 31, 58, 61, 86
lua-visual-debug	41
multicol	21, 91
shortlst	79
\par	843, 871, 994, 1073, 1136, 1142, 1171, 1188, 1818, 2435, 2455, 2572, 2587, 2708, 2811, 2818, 3104, 3118
\parindent	3081
\parsep	39, 42, 78, 79
\parsep	1596, 1604, 2254, 2673, 2680, 2685
parsep	553
\parskip	3082
\partopsep	79
\partopsep	2255, 2678



partopsep ..... 553

peek commands:

  \peek\_meaning:NTF ..... 2961, 2975, 2992, 3003

  \peek\_meaning\_remove:NTF ..... 2968

  \peek\_remove\_spaces:n ..... 2112

\phantomsection ..... 27

\phantomsection ..... 247

prg commands:

  \prg\_do\_nothing: ..... 251

  \prg\_new\_protected\_conditional:Npnn ... 194

  \prg\_replicate:nn ..... 203, 3353

  \prg\_return\_false: ..... 198

  \prg\_return\_true: ..... 197

\printkeyans ..... 13, 88, 3189

prop commands:

  \prop\_count:N ..... 1545, 1806, 1842, 1888, 1964

  \prop\_gput:Nnn ..... 1543

  \prop\_if\_exist:NTF ..... 1541, 3155

  \prop\_item:Nn ..... 3158

  \prop\_new:N ..... 1542

\ProvidesExplPackage ..... 11

R

\raggedcolumns ..... 2423, 2561

\ref ..... 58, 60

ref ..... 450, 470

\refstepcounter ..... 3031

regex commands:

  \regex\_match:nnTF 196, 510, 512, 524, 526, 2326, 2339, 2880, 2893

  \regex\_replace\_once:nnN ..... 386

\renewcommand ..... 398, 419, 429

\RenewDocumentCommand ... 2004, 2074, 2108, 2134, 2150

\RequirePackage ..... 9

resume ..... 1308

resume\* ..... 1308

rightmargin ..... 597

\Roman ..... 29, 33

\Roman ..... 294

\roman ..... 29, 33

\roman ..... 295, 468, 3177

S

save-ans ..... 1308

scan commands:

  \scan\_stop: ..... 79, 2687, 2837, 3146, 3149

seq commands:

  \seq\_clear:N ..... 3251

  \seq\_const\_from\_clist:Nn ..... 3239

  \seq\_count:N ..... 2647, 3255

  \seq\_gclear:N ..... 2002, 2003

  \seq\_gput\_right:Nn ..... 1553, 2015, 2016

  \seq\_if\_empty:NTF ..... 2021, 3204, 3269

  \seq\_if\_exist:NTF ..... 1551, 3202

  \seq\_item:Nn ..... 2705

  \seq\_map\_function:NN ..... 3260

  \seq\_map\_inline:Nn .. 3209, 3214, 3248, 3270, 3271

  \seq\_map\_pairwise\_function:NNN ..... 2023

  \seq\_new:N ..... 88, 89, 106, 135, 136, 1552

  \seq\_pop\_left:NN ..... 3259

  \seq\_put\_right:Nn ..... 2608, 3267, 3284

  \seq\_set\_from\_clist:Nn ..... 3252

  \seq\_set\_map\_e:NNn ..... 3261

\setcounter .. 521, 525, 527, 2212, 2214, 2259, 2261, 2652

\setenumext .. 5-8, 89, 3165, 3170, 3175, 3180, 3185, 3244

\setlength ..... 1597, 1605

show-ans ..... 1467, 1496

show-length ..... 673

skip commands:

  \skip\_add:Nn . 812, 818, 824, 834, 838, 862, 866, 963, 969, 975, 985, 989, 1011, 1064, 1068, 2673

  \skip\_eval:n ..... 1596, 1604

  \skip\_gset:Nn ..... 1084, 1088, 1092

  \skip\_gzero\_new:N ..... 1079, 1080

  \skip\_horizontal:N .... 645, 657, 669, 3064, 3078

  \skip\_horizontal:n 631, 1625, 1633, 2127, 2129, 3062

  \skip\_if\_eq:nnTF . 810, 816, 822, 886, 920, 961, 967, 973, 1004, 1009, 1030, 1081, 1103, 1216, 1230, 1244, 1255, 1266, 1277, 1288, 1299

  \skip\_new:N ..... 50, 51, 55, 56, 57, 58, 59, 110, 165

  \skip\_set:Nn . 795, 799, 848, 852, 889, 893, 897, 904, 908, 912, 923, 928, 932, 938, 943, 948, 1006, 1007, 1008, 1015, 1019, 1023, 1032, 1037, 1041, 1044, 1048, 1052, 1083, 1087, 1105, 1109, 1113, 1119, 1123, 1127, 2667, 2681

  \skip\_set\_eq:NN .... 2207, 2253, 2254, 3081, 3082

  \skip\_use:N 797, 801, 836, 840, 844, 864, 868, 887, 906, 915, 921, 926, 930, 941, 945, 946, 951, 987, 991, 1017, 1217, 1221, 1224, 1231, 1235, 1238, 2435

  \skip\_zero:N ..... 2255, 2414, 2555, 2678, 2679

  \skip\_zero\_new:N 999, 1000, 1001, 1078, 1100, 1101, 1102

  \c\_zero\_skip . 810, 816, 822, 887, 921, 961, 967, 973, 1004, 1009, 1030, 1081, 1103, 1217, 1231, 1244, 1255, 1266, 1277, 1288, 1299

\small ..... 3167, 3172, 3177, 3182, 3187

\star ..... 1986

start ..... 533

\stepcounter ..... 2008, 2615

store-ref ..... 1467

str commands:

  \c\_backslash\_str 3362, 3371, 3372, 3376, 3377, 3381, 3382, 3413, 3414, 3418, 3423, 3424

  \c\_colon\_str ..... 1805, 1887, 3146

  \str\_count:n ..... 203, 3353

  \str\_if\_eq:nnTF ..... 2217, 2265

  \str\_if\_eq\_p:nn ..... 2210, 2258

  \str\_if\_in:nnTF ..... 3142

  \str\_new:N ..... 105, 160

  \str\_set:Nn ... 374, 375, 376, 1478, 1479, 1501, 1502

\string ..... 240

\strutbox 891, 895, 899, 910, 914, 925, 934, 940, 950, 963, 969, 975, 1006, 1007, 1008, 1011, 1021, 1025, 1034, 1041, 1046, 1054, 1083, 1084, 1087, 1094, 1107, 1115, 1121, 1129, 2683

T

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

  \@auxout ..... 256

  \protected@write ..... 256

text commands:

  \text\_expand:n ..... 3138

\textasteriskcentered ..... 1475, 1489

\thepage ..... 262

tl commands:

  \c\_space\_tl .... 1862, 1913, 1954, 1977, 3320, 3335

  \tl\_clear:N ..... 347, 353, 1679, 1852, 1903

  \tl\_clear\_new:N ..... 304

  \tl\_const:Nn ..... 137, 288

  \tl\_gclear:N ... 1941, 2145, 2479, 2822, 3065, 3133

<code>\tl_gput_right:Nn</code> .....	289
<code>\tl_greplace_all:Nnn</code> .....	310
<code>\tl_gset:Nn</code> .....	1929, 2441, 2927, 2998
<code>\tl_gset_eq:NN</code> .....	306, 2054, 3058
<code>\tl_if_blank:nTF</code> .....	3056
<code>\tl_if_empty:nTF</code> ..	404, 438, 444, 1560, 1591, 1705, 1937, 2124, 3282
<code>\tl_if_novalue:nTF</code> ..	1680, 1691, 1860, 1911, 1953, 1976, 2006, 2031, 2050, 2055, 2085, 2645, 2867, 3246
<code>\tl_map_inline:Nn</code> .....	307, 384
<code>\tl_new:N</code> 28, 35, 37, 38, 71, 72, 73, 79, 80, 81, 83, 84, 85, 86, 92, 93, 103, 104, 114, 126, 127, 128, 131, 139, 140, 143, 144, 159, 162	
<code>\tl_put_left:Nn</code> ....	1568, 1601, 1688, 1947, 1970
<code>\tl_put_right:Nn</code> 305, 396, 417, 427, 1520, 1527, 1572, 1607, 1690, 1696, 1704, 1707, 1717, 1722, 1725, 1731, 1757, 1767, 1781, 1797, 1803, 1808, 1855, 1858, 1862, 1878, 1882, 1885, 1890, 1906, 1909, 1913, 1919, 1954, 1977, 2331, 2344, 2885, 2898, 3163, 3168, 3173, 3178, 3183	
<code>\tl_remove_all:Nn</code> .....	3281
<code>\tl_remove_once:Nn</code> .....	1745, 1872
<code>\tl_replace_all:Nnn</code> .....	309
<code>\tl_reverse:N</code> .....	1744, 1746, 1871, 1873
<code>\tl_set:Nn</code> 145, 274, 348, 352, 357, 358, 392, 411, 628, 642, 654, 666, 1326, 1474, 1488, 1836, 1958, 2052, 3279	
<code>\tl_set_eq:NN</code> 315, 393, 395, 414, 416, 424, 426, 1743, 1870, 2097, 2101, 2626, 2628	
<code>\tl_to_str:n</code> .....	3138
<code>\tl_trim_spaces:n</code> .....	305, 3267, 3279, 3285
<code>\tl_use:N</code> ..	311, 314, 406, 440, 446, 699, 703, 707, 711, 715, 719, 723, 727, 731, 735, 739, 743, 747, 751, 755, 759, 1630, 1750, 1758, 1769, 1783, 1788, 1800, 2039, 2045, 2070, 2088, 2092, 2100, 2128, 2136, 2137, 2144, 2152, 2153, 2159, 2286, 2487, 2631, 2816, 3068, 3079, 3083, 3192, 3193, 3194, 3195, 3196, 3263
token commands:	
<code>\token_to_str:N</code> .....	258
<code>\topsep</code> .....	1597, 1605
<code>topsep</code> .....	<u>553</u>
<code>\typeout</code> .....	226, 229, 236, 239, 240
<b>U</b>	
<code>\u</code> .....	387
use commands:	
<code>\use:N</code> .....	204, 2141, 2288
<code>\use:n</code> .....	3144
<code>\use_none:nn</code> .....	250
<code>\usecounter</code> .....	2208, 2256
<b>V</b>	
<code>\value</code> .....	2461, 2466, 2917, 2922
<code>\vspace</code> 878, 1221, 1224, 1235, 1238, 1248, 1250, 1259, 1261, 1270, 1272, 1281, 1283, 1292, 1294, 1303, 1305, 1596, 1604, 2642, 2653, 3119	
<b>W</b>	
<code>widest</code> .....	<u>533</u>
<code>wrap-ans</code> .....	<u>1467</u>
<code>wrap-label</code> .....	<u>318</u>
<code>wrap-label*</code> .....	<u>318</u>