

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-29<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>1</b>	<b>4.3</b>	<b>Keys for add code</b>	<b>8</b>
1.1	Description and usage	2	4.4	Keys for start, series and resume	9
1.2	The concept of left margin	3	4.5	Keys for multicol	9
1.3	User interface	3	4.6	Keys for minipage	9
1.3.1	Internal counters	3	4.6.1	The command <code>\miniright</code>	10
1.3.2	Support for multicol	3	4.6.2	The key <code>mini-right</code>	10
1.3.3	Support for minipage	4	<b>5</b>	<b>The storage system</b>	<b>10</b>
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.1	Keys for storage system	10
1.3.5	Support for <code>\footnote</code>	4	5.1.1	Keys for label and ref	11
<b>2</b>	<b>The environments provided</b>	<b>4</b>	5.1.2	Keys for wrap and display	11
2.1	The environment <code>enumext</code>	4	5.1.3	Keys for debug and checking	11
2.2	The environment <code>enumext*</code>	5	5.2	The command <code>\anskey</code>	12
2.3	The command <code>\item*</code>	5	5.3	The environment <code>keyans</code>	12
2.3.1	Keys for <code>\item*</code>	5	5.3.1	The <code>\item*</code> in <code>keyans</code>	12
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.4	The environment <code>keyanspic</code>	13
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>6</b>	5.4.1	The command <code>\anspic</code>	13
<b>4</b>	<b>The <code>keyval</code> system</b>	<b>6</b>	5.5	Printing stored content	14
4.1	Keys for <code>label</code> and <code>ref</code>	6	5.5.1	The command <code>\getkeyans</code>	14
4.2	Keys for spaces	7	5.5.2	The command <code>\printkeyans</code>	14
4.2.1	Vertical spaces	7	<b>6</b>	<b>Full examples</b>	<b>15</b>
4.2.2	Horizontal spaces	8	<b>7</b>	<b>The way of non-enumerated lists</b>	<b>18</b>
			<b>8</b>	<b>References</b>	<b>19</b>
			<b>9</b>	<b>Change history</b>	<b>20</b>
			<b>10</b>	<b>Index of Documentation</b>	<b>21</b>
			<b>11</b>	<b>Implementation</b>	<b>23</b>
			<b>12</b>	<b>Index of Implementation</b>	<b>114</b>

## Motivation and acknowledgments

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

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

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all  $\text{\TeX}$  team for their great work and to the different members of the `TeX-SX` community who have provided great answers and ideas. Here a note of the main ones:

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

<sup>\*</sup>This file describes a documentation for v1.0, last revised 2024-05-29.

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

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (l<sup>pp</sup>l), version 1.3 or later (<https://www.latex-project.org/l<sup>pp</sup>l.txt>). The software has the status “maintained”.  
The enumext package loads and requires multicol[3] package, need to have a modern T<sub>E</sub>X distribution such as T<sub>E</sub>X Live or MiK<sub>T</sub>E<sub>X</sub>. It has been tested with the standard classes provided by L<sup>A</sup>T<sub>E</sub>X: book, report, article and letter on 10pt, 11pt and 12pt.

1 Introduction

In the L<sup>A</sup>T<sub>E</sub>X 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) L<sup>A</sup>T<sub>E</sub>Xze 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) L<sup>A</sup>T<sub>E</sub>Xze 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

(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


- Where what we are interested in the <label> and a “short note” that we leave as an explanation, and then print them:
- ©2024 by Pablo González L

2 / 126

1. (B),  $x = 5$

2. (D)

3. (C), some note
- \* 4. (E), A duck

\* 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 `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

### 1.3 User interface

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

#### 1.3.1 Internal counters

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

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

#### 1.3.2 Support for multicol

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.

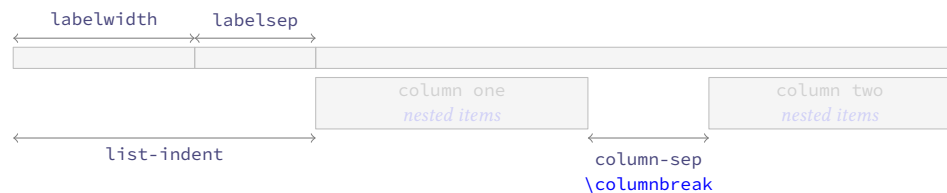


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §4.5).

- 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 §4.6).

#### 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 (§4.1), the standard  $\TeX$  `\label` and `\ref` commands work as usual. It also provides an “*internal reference*” system for the “*stored content*” by means of the key `save-ref` (§5.1.1) when the key `save-ans` (§5.1) is active.

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

#### 1.3.5 Support for \footnote

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

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 environments provided

The package `enumext` provides two main list environments, the *vertical* environment `enumext` and the *horizontal* environment `enumext*`.

<code>enumext</code>	<code>\begin{enumext}[\langle keyval list \rangle]</code>	<code>\begin{enumext*}[\langle keyval list \rangle]</code>
<code>enumext*</code>	<code>\item \langle item content \rangle</code>	<code>\item \langle item content \rangle</code>
	<code>\item [\langle custom \rangle] \langle item content \rangle</code>	<code>\item [\langle custom \rangle] \langle item content \rangle</code>
	<code>\item* [\langle symbol \rangle] [\langle offset \rangle] \langle item content \rangle</code>	<code>\item* [\langle symbol \rangle] [\langle offset \rangle] \langle item content \rangle</code>
	<code>\end{enumext}</code>	<code>\end{enumext*}</code>

2.1 The environment enumext

The `enumext` is an environment that works in the same way as the standard `enumerate` environment provided by  $\text{\LaTeX}$ , `\item` and `\item[\langle custom \rangle]` 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 `[\langle key = val \rangle]` in the environment.

Example with columns=2

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.

2.2 The environment enumext\*

The `enumext*` environment is a horizontal list environment similar to the `enumerate*` environment provided by the `enumitem` package or `task` environment provided by the `task` package , `\item` and `\item[\langle custom \rangle]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself, but it can be nested within `enumext` and can contain it nested within it.
- Each “item” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that includes `labelwidth`, `labelsep` plus the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.

Example with columns=2

1. This text is in the first level.

X This text is in the first level.
2. This text is in the first level.

★ 3. This text is in the first level.

2.3 The command \item\*

```
\item* \item*
\item* [\langle symbol \rangle]
\item* [\langle symbol \rangle] [\langle offset \rangle]
```

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` works like the numbered `\item`, but placing a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the value set by the `labelsep` key and can be `\langle offset \rangle` using the second optional argument. The default values for `\langle symbol \rangle` and `\langle offset \rangle` 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` and `enumext*` environments is NOT the same as in the `keyans` and `keyans*` environments.

### 2.3.1 Keys for `\item*`

`item-sym*` =  $\{\langle symbol \rangle\}$

default:  $\$ \star \$$

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

`item-pos*` =  $\{\langle rigid length \rangle\}$

default: *by levels*

Sets the *offset* between the box containing the current  $\langle label \rangle$  defined by `labelwidth` key and the  $\langle symbol \rangle$  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*.

## 2.4 The command `\item` in `enumext*`

The `\item` command for the `enumext*` environment provides an optional first argument `\item(\langle number \rangle)` which “joins items” between columns. Let’s consider the following examples adapted directly from the `task` package:

```
\begin{enumext*}[widest=10,columns=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item the seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
  \item[Z] The ninth
  \item The tenth
\end{enumext*}
```

- |  |  |              |               |
|--|--|--------------|---------------|
| 1. The first   | * 2. The second  | 3. The third | 4. The fourth |
| * 5. The fifth item is way too long for this and needs three columns | 6. The sixth   |              |               |
| 7. the seventh   | X The eighth item is way too long for this and needs Z | The ninth    |               |
| 8. The tenth   | two columns  |              |               |

## 3 The command `\setenumext`

---

`\setenumext`

`\setenumext{\langle key = val \rangle}`

`\setenumext[\langle enumext, level \rangle]{\langle key = val \rangle}`

`\setenumext[\langle enumext* \rangle]{\langle key = val \rangle}`

`\setenumext[\langle keyans \rangle]{\langle key = val \rangle}`

`\setenumext[\langle keyans* \rangle]{\langle key = val \rangle}`

`\setenumext[\langle print, level \rangle]{\langle key = val \rangle}`

`\setenumext[\langle print, * \rangle]{\langle key = val \rangle}`

`\setenumext[\langle print* \rangle]{\langle key = val \rangle}`

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

The  $\langle keys \rangle$  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.

- The key `save-ans` that activate the “storage system” must NOT be passed through this command and must be passed directly in the optional argument of the “first level” of the environment in which they are executed.

## 4 The keyval system

The  $\langle key = val \rangle$  system used by the `enumext` package is implemented using `l3keys` so it must be taken into consideration that those keys marked as “value forbidden”, that is  $\langle key \rangle$  is different from  $\langle key = \rangle$ .

All  $\langle keys \rangle$  described in this section are available for the `enumext`, `enumext*`, `keyans` and `keyans*` environments with the exception of the keys `series`, `resume`, `resume*` which are only available for the “first level” of the environments `enumext` and `enumext*`; and the keys `mini-right`, `mini-right*` which are only available for the `enumext*` and `keyans*` 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 `\dimeval` or `\dimexpr` to perform calculations.

It should be kept in mind that using any  $\langle key \rangle$  that sets a *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and `keyans`, `keyans*` and `keyanspic` environments.



## 4.1 Keys for label and ref

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

Sets the *⟨label⟩* that will be printed at the *current level*. The default value for the first level of the environments `enumext` and `enumext*` are `\arabic*`, for second level are `(\alph*)`, for third level are `\roman*`, and for fourth level are `\Alph*`. For `keyans` and `keyans*` environments the default value is `\Alph*`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 4.2 Keys for spaces

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

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

### 4.2.1 Vertical spaces

`topsep` = { $\langle$ rubber length | rigid length $\rangle$ } 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 value for the first level of the environments `enumext` and `enumext*` are 8.0pt plus 2.0pt minus 4.0pt, for second level are 4.0pt plus 2.0pt minus 1.0pt, for third and fourth level are 2.0pt plus 1.0pt minus 1.0pt. For `keyans` and `keyans*` environments the default value is 4.0pt plus 2.0pt minus 1.0pt.

`parsep` = { $\langle$ rubber length | rigid length $\rangle$ } default: *by levels*

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

`partopsep` = { $\langle$ rubber length | rigid length $\rangle$ } 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 in environment `enumext` are 2.0pt plus 1.0pt minus 1.0pt, for third and fourth level are 1.0pt minus 1.0pt. For `keyans`, `keyans*` and `enumext*` environments the default value is 2.0pt plus 1.0pt minus 1.0pt.

- The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. 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  $\langle$ vertical mode $\rangle$  and apply this value to the “top” and “bottom” the environment or nested level.

`itemsep` = { $\langle$ rubber length | rigid length $\rangle$ } default: *by levels*

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

`noitemsep`  $\langle$ value forbidden $\rangle$  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`  $\langle$ value forbidden $\rangle$  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  $\langle$ keys $\rangle$  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 ‘\*’  $\langle$ keys $\rangle$  applies `\vspace*` so that T<sub>E</sub>X does *not discard* this space at page break.

`above` = { $\langle$ rubber length | rigid length $\rangle$ } 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*` = { $\langle$ rubber length | rigid length $\rangle$ } 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` = { $\langle$ rubber length | rigid length $\rangle$ } default: *not used*

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

`below*` = { $\langle$ rubber length | rigid length $\rangle$ } default: *not used*

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

### 4.2.2 Horizontal spaces

`itemindent` = { $\langle$ rigid length $\rangle$ } 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` = { $\langle$ rigid length $\rangle$ } 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` is set in the environment `enumext` 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.

- For the `enumext*` and `keyans*` environments the keys `list-indent` and `list-offset` have the same effect.

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

`before*` = {*<code>*} default: not used  
 Execute *<code>* “*before*” the environment starts. The *<code>* must be passed between braces, is executed “*before*” performing all calculations related to the *list parameters* and [*<key = val>*] sets in the environment that is, before the arguments defining the environment are executed: `{<code>}\list{<arg one>}{<arg two>}`.

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

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

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

### 4.4 Keys for start, series and resume

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

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

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

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

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

- For security reasons the `series` key will never save in *<series name>* the keys `series`, `resume`, `resume*`, `save-ans`, `save-key` and `start`. When using the key `resume={<series name>}` it will have hierarchy in the *<keys>* that are saved in *<series name>*, in order to establish the value of a *<key>* already saved in *<series name>* it must be placed to the

“right” of `resume={⟨series name⟩}`, the same thing happens with the `resume*` key, the exception is the `save-ans` key that must be placed on the “left” if you want to start the numbering with its value. The `resume` key passed “without value” must be exactly “without value”, i.e. `resume=` cannot be used and if executed before `resume*` it will affect the start value.

## 4.5 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.

## 4.6 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`.

### 4.6.1 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 L<sup>A</sup>T<sub>E</sub>X justification is maintained in the `minipage` on the “right side”.

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

### 4.6.2 The key mini-right

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

`mini-right = {⟨code for drawing or tabular⟩}`

default: not used

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

`mini-right* = {⟨code for drawing or tabular⟩}`

default: not used

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

## 5 The storage system

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

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{keyans}
      ...
    \end{keyans}
\end{enumext}
```

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{keyanspic}
      ...
    \end{keyanspic}
\end{enumext}
```

By executing the key `save-ans={⟨store name⟩}` the entire structure of the environment (excluding the first level) including the optional arguments passed to the inner levels or the environment nested in it, along with the content passed to `\anskey`, the current `⟨labels⟩` for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be stored in a `⟨sequence⟩` and at the same time will be stored (without the environment structure or optional arguments) in a `⟨prop list⟩`.

The optional arguments of the inner levels or the nested environment are filtered by excluding all `⟨keys⟩` related to the “stored system” along with the keys `series`, `resume` and `resume*` when storing in `⟨sequence⟩`.

## 5.1 Keys for storage system

- The only *⟨keys⟩* available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the *⟨keys⟩* described in this section must be passed directly in the optional argument of the “first level” of the environment in which the key `save-ans` is executed. The key `save-ans` should NOT be passed with the command `\setenumext`.

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

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

`save-key = {⟨key list⟩}` default: *not set*

This key *overrides* the default “stored keys” of the optional arguments of the inner levels or nested environment that will be passed to the *⟨sequence⟩*. The *⟨key list⟩* passed to this key ignores any *⟨keys⟩* in the “stored system” and must be passed between braces. For example, if we execute at a second level:

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{enumext}[nosep, columns=2, save-key={columns=3}]
      ...
    \end{enumext}
\end{enumext}
```

The *⟨keys⟩* that will be stored by default in the *⟨sequence⟩* would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the *⟨sequence⟩* only the key `columns=3` ignoring all the others.

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

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

### 5.1.1 Keys for label and ref

`save-ref = {⟨true | false⟩}` default: *false*

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

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

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

### 5.1.2 Keys for wrap and display

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

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

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

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

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

Displays the *current ⟨argument⟩* passed to the `\anskey` command, the current *⟨label⟩* for `\item*` and `\anspic*` commands at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

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

Sets the *symbol* to be displayed in the left margin for the commands `\anskey`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

`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.

### 5.1.3 Keys for debug and checking

`show-pos = {⟨true | false⟩}` default: *false*  
 Displays the *position* occupied by the “stored content” by commands `\anskey`, `\item*` and `\anspic*` in the *prop list* ⟨*store name*⟩ set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans = {⟨true | false⟩}` default: *false*  
 Enables the *checking answer* mechanism by displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “only one answer” or “only one execution” of the `\anskey` command. It is intended to be used in conjunction with the `no-store` key.

`no-store` ⟨*value forbidden*⟩ default: *not used*  
 This is a *meta-key* that does not receive an argument and disables the environment structure stored in the ⟨*sequence*⟩ at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested environments in which you want to use `enumext` or `enumext*` but without using the `\anskey` command, without interfering with the `check-ans` key and without storing an unwanted environment structure in the ⟨*sequence*⟩.

## 5.2 The command `\anskey`

---

```
\anskey ⟨content⟩
\anskey[⟨keys⟩]{⟨content⟩}
\anskey*[⟨keys⟩]{⟨content⟩}
```

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

### Example

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

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

## 5.3 The environment `keyans`

---

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

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

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

The `<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[<keyans>]{<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*)`.

5.3.1 The `\item*` in `keyans`

`\item*` `\item*`  
`\item*{<content>}`

The `\item*` and `\item*{<content>}` 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` or `enumext*` environments.

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

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

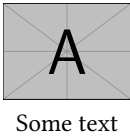
Example

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

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

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

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



5.4 The environment `keyanspic`

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

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

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

5.4.1 The command `\anspic`

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

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

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



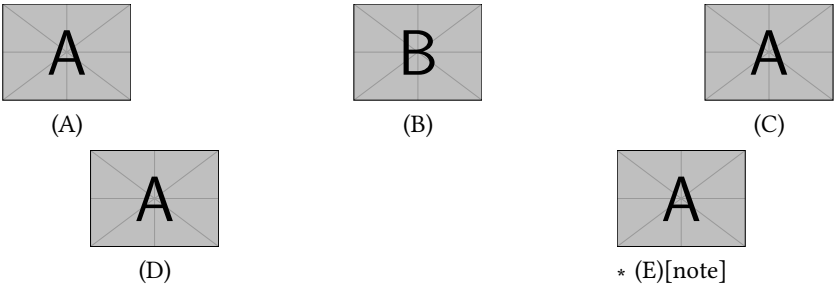


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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.



5.5 Printing stored content

5.5.1 The command `\getkeyans`

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

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

The form taken by the argument `{<store name> : <position>}` is the same as that used to generate the “internal label and ref” system when `save-ref` key are active, so to refer to a “stored content”. For example `\getkeyans{test:4}` will return the “stored content” at position `4` of the environment in which the key `save-ans=test` was set.

5.5.2 The command `\printkeyans`

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

The command `\printkeyans` prints “all stored content” in *sequence* `{<store name>}` defined by `save-ans` key placing this inside the `enumext` environment or the `enumext*` environment if the *starred version* ‘`*`’ is used. The “stored content” can only be accessed *after* it is stored in the *sequence*, if `{<store name>}` does not exist the command will return an error.

The optional argument allows to handle the `<keys>` on the “first level” of the environment `enumext` or `enumext*` if the *starred version* ‘`*`’ is used. The default values for the “first level” are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep`, `first=\small`, `font=\small` and `columns=2`.

For the inner levels of the environment `enumext` saved in the *sequence* the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`, if the environment `enumext*` is saved within the *sequence*, it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.



Setting the keys for \printkeyans

The default values for the “first level” of \printkeyans commands and \printkeyans\* are established using \setenumext[⟨print , 1⟩]{⟨keys⟩} and \setenumext[⟨print\*⟩]{⟨keys⟩}. The optional argument allows to handle the ⟨keys⟩ “on the first level” of the enumext environment encapsulated by the command. If need to pass options for nested levels use \setenumext[⟨print , level⟩]{⟨store name⟩}.

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1] 

$3(x + y + z)$

2. True False

(a) 

$\LaTeXe$  is cool?

[2] 

Very True!

3. Related to Linux

(a) You use linux?

[3] 

Yes

(b) Rate the following package and class

i. 

xsim

[4] 

very good

ii. 

exsheets

[5] 

obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- \*

\*

\*

\*

\*


6 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 orara<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:  $1 \text{ \AA} =$

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

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

C  $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$ .
- $1 \times 10^{-10} \text{ m}$ ) e il fermi o femtometro ( $1 \text{ fm} =$

$1 \times 10^{-15} \text{ m}$ ). Qual è la relazione tra queste due

unità di misura?

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

- ☐ D

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

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

☐ A

36 km/h.

☐ B

360 km/h.

☐ C

27,8 km/h.

☐ D

$3,60 \times 10^8 \text{ km/h.}$

4. In fisica nucleare si usa l'angstrom (simbolo:  $1 \text{ \AA} =$ 

☐ A

$1 \times 10^{-10} \text{ m}$

 e il fermi o femtometro ( $1 \text{ fm} = 1 \times 10^{-15} \text{ m}$ ). Qual è la relazione tra queste due unità di misura?

☐ A

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

☐ 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 \(cross mark\)](#) .

- ☐ A

36 km/h.

☒ B

360 km/h.

☐ C

27,8 km/h.

☐ D

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

☒ A

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

☐ 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  $1,00 \times 10^2 \text{ m/s}$  espressa in km/h è:

☐ A

36 km/h.

☒ B

360 km/h.

☐ C

27,8 km/h.

☐ D

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

☒ A

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

☐ 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 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

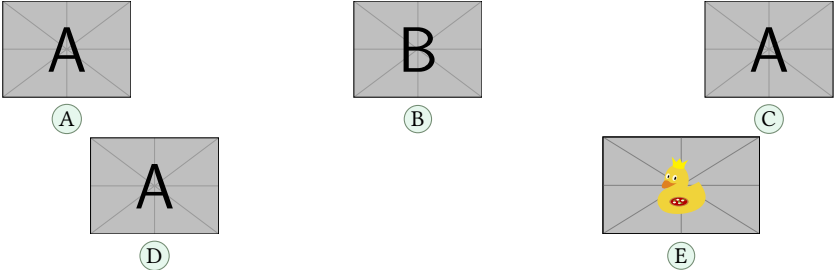
 value

D

 value

E

 value
4. Question with image and label below:



5. Question with image on left side:
- A

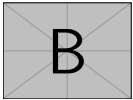
 value
- B

 value
- C

 value
- D

 correct
- E


 value



Test keys

1. B,  $x = 5$
2. D
3. C, some note
4. E, A duck
5. D, other note

Example 4

A “simple worksheet” using ducks :) .

- 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 .

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

1. A)

2. C)

3. B)

4. D)

7 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  $\text{\LaTeX}$  for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

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

Fake description environment

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

- Something** A short one-line description.

This is an entry *without* a label.
- Something** A short *one-line* description text.
- Something long** A much *longer* description text may take more than one line or more than one paragraph.

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

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

- Something** A short one-line description.

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

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

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

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

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

**SoMeThInG** A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, 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.

## 8 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.

## 9 Change history

**v1.0** 2024-05-29 – First public release.



10 Index of Documentation

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

C

Document class:

article

book

exam

letter

report

\columnbreak

\columnsep

Commands provide by enumext:

\anskey

\anspic\*

\anspic

\getkeyans

\item\*

\itemwidth

\item

\miniright

\printkeyans

\setenumext

Counters defined by enumext:

enumXiii

enumXii

enumXiv

enumXi

enumXviii

enumXvii

enumXvi

enumXv

E

Environments provide by enumext:

enumext\*

enumext

keyans\*

keyanspic

keyans

Environments:

enumerate

figure

list

minipage

multicols

table

task

F

\footnote

I

\item

\itemsep

K

Keys for environments provide by enumext:

above\*

above

after

align

before\*

before

below\*

below

check-ans

columns-sep

columns

first

font

item-pos\*

item-sym\*

itemindent

itemsep

labelsep

labelwidth

labelwith

label

list-indent

list-offset

listparindent

mark-ans

mark-pos

mark-ref

mini-env

mini-right\*

mini-right

mini-sep

no-store

noitemsep

nosep

parsep

partopsep

ref

resume\*

resume\*

resume

rightmargin

save-ans

save-key

save-ref

save-sep

series

show-ans

show-length

show-pos

start

topsep

widest

wrap-ans

wrap-label\*

wrap-label

wrap-opt

L

\label

Labels provide by enumext:

\Alph\*

\Roman\*

\alph\*

\arabic\*

\roman\*

\labelsep

\labelwidth

\linewidth

©2024 by Pablo González L

21 / 126

22 / 126

## 11 Implementation

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

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

### 11.1 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.

### 11.2 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>
```

### 11.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-29}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

### 11.4 Definition of variables

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

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

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

```

27 \bool_new:N \l__enumext_starred_first_bool
28 \bool_new:N \l__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_first_bool
31 \bool_new:N \l__enumext_keyans_env_bool

```

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

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

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

The initial values of these variables are set by the function `\__enumext_define_counters:Nn` (§11.8) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§11.11).

```

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

```

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

Internal variables used by `ref` key (§11.11).

```

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

```

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

Internal variables used by `resume`, `resume*` and `series` keys. The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§11.26).

```

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

```

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

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default `<label style>` and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§11.12) and `label` (§11.10) keys.

```

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

```

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

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§11.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `\__enumext_calc_hspace:NNNNNNNNNN` (§11.30.1) which determines the internal values for `\leftmargin` and `\itemindent`.

```

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

```

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

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
```

Internal variables used by `columns` key §11.18).

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

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

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

Internal variables used by `\miniright` command (§11.19.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.17, §11.19).

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

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

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

The integer variable `\l__enumext_start_X_int` are used by the `start` key (§11.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.10). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

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

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

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

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

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

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§11.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§11.28.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§11.33.1) for the `keyanspic` environment.

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

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

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

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

Internal variables used by the command `\setenumext` (§11.38).

```
106 \tl_new:N \l__enumext_setkey_tmpa_tl
107 \tl_new:N \l__enumext_setkey_tmpb_tl
108 \int_new:N \l__enumext_setkey_tmpa_int
109 \seq_new:N \l__enumext_setkey_tmpa_seq
110 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

```
\l__enumext_print_keyans_starred_tl
\l__enumext_store_save_key_X_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_upper_level_X_bool
```

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§11.37) and `save-key` key.

```
111 \tl_new:N \l__enumext_print_keyans_starred_tl
112 \cs_set_protected:Npn \__enumext_tmp:n #1
113 {
114   \tl_new:c { \l__enumext_store_save_key_#1_tl }
115   \bool_new:c { \l__enumext_store_save_key_#1_bool }
116   \tl_new:c { \l__enumext_store_active_keys_#1_tl }
117   \tl_new:c { \l__enumext_print_keyans_#1_tl }
118   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
119 }
120 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_print_keyans_starred_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` (§11.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§11.23).

```
121 \bool_new:N \l__enumext_show_answer_bool
122 \bool_new:N \l__enumext_show_position_bool
123 \tl_new:N \l__enumext_mark_ref_sym_tl
124 \tl_new:N \l__enumext_mark_answer_sym_tl
125 \str_new:N \l__enumext_mark_position_str
```

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

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

Internal variables used by `keyanspic` environment (§11.33.2).

```
126 \seq_new:N \l__enumext_keyans_pic_body_seq
127 \dim_new:N \l__enumext_keyans_pic_width_dim
128 \int_new:N \l__enumext_keyans_pic_above_int
129 \int_new:N \l__enumext_keyans_pic_below_int
130 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

```
\l__enumext_check_answers_bool
\l__enumext_check_ans_key_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\g__enumext_start_line_tl
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int
```

Internal variables used by “*check answer*” mechanism (§11.22.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
131 \bool_new:N \l__enumext_check_answers_bool
132 \bool_new:N \l__enumext_check_ans_key_bool
133 \bool_new:N \g__enumext_check_ans_key_bool
134 \tl_new:N \l__enumext_check_start_line_env_tl
135 \tl_new:N \g__enumext_start_line_tl
136 \tl_new:N \g__enumext_envir_name_tl
137 \int_new:N \g__enumext_check_starred_cmd_int
138 \int_new:N \g__enumext_item_anskey_int
139 \int_new:N \g__enumext_item_number_int
140 \int_new:N \g__enumext_item_answer_diff_int
```

(End of definition for `\l__enumext_check_answers_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 (§11.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
141 \bool_new:N \l__enumext_hyperref_bool
142 \bool_new:N \l__enumext_footnotes_key_bool
```

(End of definition for `\l__enumext_hyperref_bool` and `\l__enumext_footnotes_key_bool`.)



```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_store_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

Internal variables are used when executing the `save-ref` key. The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the labels defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `\__enumext_newlabel:nn` and the variable `\l__enumext_store_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

143 \tl_new:N \l__enumext_newlabel_arg_one_tl
144 \tl_new:N \l__enumext_newlabel_arg_two_tl
145 \tl_new:N \l__enumext_store_write_aux_file_tl
146 \cs_set_protected:Npn \__enumext_tmp:n #1
147 {
148   \tl_new:c { l__enumext_label_copy_#1_tl }
149 }
150 \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`.

```

151 \int_new:N \g__enumext_footnote_int
152 \seq_new:N \g__enumext_footnote_arg_seq
153 \seq_new:N \g__enumext_footnote_int_seq

```

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

```

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

```

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

```

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

```

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

```

\c__enumext_all_envs_clist

```

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

```

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

```

(End of definition for `\c__enumext_all_envs_clist`.)

## 11.5 Some utility functions

```

\__enumext_at_begin_document:n

```

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

```

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

```

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

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

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

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

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

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

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

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

```
191 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
192 {
193   \regex_match:nnTF { ^[\+\\-]?[\d]+\$ } {#1} % $
194   { \prg_return_true: }
195   { \prg_return_false: }
196 }
```

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

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

```
197 \cs_new_protected:Nn \__enumext_regex_counter_style:
198 {
199   \tl_map_inline:Nn \c__enumext_counter_style_tl
200   {
201     \regex_replace_once:nnN { \c{##1}\* }
202     { \c{##1}\cB{\u{\__enumext_ref_the_count_tl}\cE} } \__enumext_ref_key_arg_tl
203   }
204 }
```

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

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

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

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

### 11.5.1 Utilities for environments and levels

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

```
211 \cs_new_protected:Nn \__enumext_is_not_nested:
212 {
213   \str_case:en { \@currentvir }
214   {
215     {enumext}
216     {
217       \bool_lazy_and:nnT
218       { \bool_not_p:n { \g__enumext_standar_bool } }
219       { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
220       {
221         \bool_gset_true:N \g__enumext_standar_bool
```

```

222     }
223   }
224   {enumext*}
225   {
226     \bool_lazy_and:nnT
227     { \bool_not_p:n { \g__enumext_starred_bool } }
228     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
229     {
230       \bool_gset_true:N \g__enumext_starred_bool
231     }
232   }
233 }
234 }

```

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

```

235 \cs_new_protected:Nn \__enumext_is_on_first_level:
236 {
237   \bool_lazy_all:nT
238   {
239     { \bool_if_p:N \g__enumext_standar_bool }
240     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
241     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
242   }
243   {
244     \bool_set_true:N \l__enumext_standar_first_bool
245     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
246     \tl_gset:Ne \g__enumext_start_line_tl
247     {
248       on ~ line ~ \exp_not:V \inputlineno
249     }
250   }
251   \bool_lazy_all:nT
252   {
253     { \bool_if_p:N \g__enumext_starred_bool }
254     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
255     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
256   }
257   {
258     \bool_set_true:N \l__enumext_starred_first_bool
259     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
260     \tl_gset:Ne \g__enumext_start_line_tl
261     {
262       on ~ line ~ \exp_not:V \inputlineno
263     }
264   }
265 }

```

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

`\__enumext_keyans_save_start_line:`

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

```

266 \cs_new_protected:Nn \__enumext_keyans_save_start_line:
267 {
268   \str_case:en { \@currenvir }
269   {
270     {keyans}
271     {
272       \tl_set:Ne \l__enumext_check_start_line_env_tl
273       {
274         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
275       }
276     }
277     {keyans*}
278     {
279       \tl_set:Ne \l__enumext_check_start_line_env_tl
280       {

```

```

281         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
282     }
283 }
284 {keyanspic}
285 {
286     \tl_set:Nc \l__enumext_check_start_line_env_tl
287     {
288         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
289     }
290 }
291 }
292 }

```

(End of definition for `\__enumext_keyans_save_start_line:`)

### 11.5.2 Utilities for log and terminal

The function `\__enumext_reset_global_vars:` will be passed to the function `\__enumext_execute_after_env:` and will return the global variables to their default values after being used.

```

293 \cs_new_protected:Nn \__enumext_reset_global_vars:
294 {
295     \__enumext_reset_global_int:
296     \__enumext_reset_global_bool:
297     \__enumext_reset_global_tl:
298 }
299 \cs_new_protected:Nn \__enumext_reset_global_int:
300 {
301     \int_gzero:N \g__enumext_item_number_int
302     \int_gzero:N \g__enumext_item_anskey_int
303     \int_gzero:N \g__enumext_item_answer_diff_int
304 }
305 \cs_new_protected:Nn \__enumext_reset_global_bool:
306 {
307     \bool_gset_false:N \g__enumext_check_ans_key_bool
308     \bool_gset_false:N \g__enumext_standar_bool
309     \bool_gset_false:N \g__enumext_starred_bool
310 }
311 \cs_new_protected:Nn \__enumext_reset_global_tl:
312 {
313     \tl_gclear:N \g__enumext_store_name_tl
314     \tl_gclear:N \g__enumext_start_line_tl
315     \tl_gclear:N \g__enumext_envir_name_tl
316 }

```

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

The function `\__enumext_log_global_vars:` will be passed to the function `\__enumext_execute_after_env:` and write to the `.log` file the number of elements saved in the *prop list* and *sequence* created by the `save-ans` key along with the value of the integer variable created for the `resume` key.

```

317 \cs_new_protected:Nn \__enumext_log_global_vars:
318 {
319     \msg_log:nneeee { enumext } { prop-seq-int-hook }
320     { \g__enumext_store_name_tl }
321     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
322     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
323     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
324 }

```

The function `\__enumext_log_answer_vars:` will be passed to the function `\__enumext_execute_after_env:` and write to the `.log` file the number of items and answers along with the difference between them.

```

325 \cs_new_protected:Nn \__enumext_log_answer_vars:
326 {
327     \msg_log:nneeee { enumext } { item-answer-hook }
328     { \int_use:N \g__enumext_item_number_int }
329     { \int_use:N \g__enumext_item_anskey_int }
330     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
331 }

```

(End of definition for `\__enumext_log_global_vars:` and `\__enumext_log_answer_vars:`)

## 11.6 Copying list and minipage environments

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

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

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

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

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

```
332 \__enumext_at_begin_document:n
333 {
334   \cs_new_eq:NN \__enumext_start_list:nn \list
335   \cs_new_eq:NN \__enumext_stop_list: \endlist
336   \cs_new_eq:NN \__enumext_item_std:w \item
337 }
```

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

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

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

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

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

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

```
338 \__enumext_at_begin_document:n
339 {
340   \cs_new_eq:NN \__enumext_minipage:w \minipage
341   \cs_new_eq:NN \__enumext_endminipage: \endminipage
342 }
```

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

## 11.7 Compatibility with hyperref and footnotehyper

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

```
343 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
344 \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”.

```
345 \cs_new_protected:Nn \__enumext_after_hyperref:
346 {
347   \IfPackageLoadedTF { hyperref }
348   {
349     \msg_info:nnn { enumext } { package-load } { hyperref }
350     \bool_set_true:N \l__enumext_hyperref_bool
351     \IfHyperBoolean{hyperfootnotes}
352     {
353       \typeout{hyperfootnotes=true}
354       \bool_set_true:N \l__enumext_footnotes_key_bool
355     }
356     { \typeout{hyperfootnotes=false} }
357   }
358   { }
```

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`.

```
359 \bool_if:NT \l__enumext_footnotes_key_bool
360 {
361   \IfPackageLoadedTF { footnotehyper }
362   {
```

```

363         \msg_info:nnn { enumext } { package-load } { footnotehyper }
364     }
365     {
366         \typeout{No ~ footnotehyper ~ load}
367         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
368         \bool_set_false:N \l__enumext_footnotes_key_bool
369     }
370 }

```

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.

```

371     \bool_if:NTF \l__enumext_hyperref_bool
372     {
373         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
374         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
375     }
376     {
377         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
378         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
379     }
380 }

```

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

`\__enumext_newlabel:nn`

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

- #1: `\l__enumext_newlabel_arg_one_tl`
- #2: `\l__enumext_newlabel_arg_two_tl`

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

```

381 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
382 {
383     \protected@write \@auxout { }
384     {
385         \token_to_str:N \newlabel {#1}
386         {
387             {#2}
388             \bool_if:NT \l__enumext_hyperref_bool
389             { { \thepage } {#2} {#1} }
390             { }
391         }
392     }
393     \__enumext_hypertarget:nn {#1} { }
394     \__enumext_phantomsection:
395 }

```

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

## 11.8 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.

```

396 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
397 {
398     \cs_if_exist:cTF { c@ #2 }
399     { \msg_fatal:nnn { enumext } { counters } { #2 } }
400     {
401         \tl_set:Nn #1 { #2 }
402         \newcounter { #2 }
403     }
404 }

```

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



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

```
enumXiv 405 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv 406 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi 407 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii 408 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 409 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
410 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
411 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
412 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
```

(End of definition for enumXi and others.)

## 11.9 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.

```
413 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
414 {
415   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
416   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
417 }
418 \__enumext_register_counter_style:Nn \arabic { 0 }
419 \__enumext_register_counter_style:Nn \Alph { M }
420 \__enumext_register_counter_style:Nn \alph { m }
421 \__enumext_register_counter_style:Nn \Roman { VIII }
422 \__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.

```
423 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
424 {
425   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
426   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
427 }
428 \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.

```
429 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
430 {
431   \tl_clear_new:N #1
432   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
433   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
434   \tl_map_inline:Nn \g__enumext_counter_styles_tl
435   {
436     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
437     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
438     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
439   }
440   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
441   { \tl_use:N \g__enumext_widest_label_tl }
442   \tl_set_eq:cN { the #2 } #1
443 }
444 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

## 11.10 Setting keys associated with label

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

```

445 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
446 {
447   \keys_define:nn { enumext / #1 }
448   {
449     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
450     font      .value_required:n = true,
451     labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
452     labelsep   .initial:n   = {0.3333em},
453     labelsep   .value_required:n = true,
454     labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
455     labelwidth .value_required:n = true,
456     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
457     wrap-label .initial:n   = {##1},
458     wrap-label .value_required:n = true,
459     wrap-label* .code:n = {
460       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
461       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
462     },
463     wrap-label* .value_required:n = true,
464   }
465 }
466 \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.

```

467 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
468 {
469   \keys_define:nn { enumext / #1 }
470   {
471     align .choice:,
472     align / left .code:n =
473       {
474         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
475         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
476       },
477     align / right .code:n =
478       {
479         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
480         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
481       },
482     align / center .code:n =
483       {
484         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
485         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
486       },
487     align .initial:n = left,
488     align .value_required:n = true,
489   }
490 }
491 \clist_map_inline:nn
492 {
493   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
494 }
495 { \__enumext_tmp:nn #1 }

496 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
497 {
498   \keys_define:nn { enumext / #1 }
499   {
500     align .choice:,
501     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
502     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
503     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },

```

```

504         align .initial:n = left,
505         align .value_required:n = true,
506     }
507 }
508 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

### 11.11 Setting label and ref keys

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

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

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
509 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
510 {
511     \keys_define:nn { enumext / #1 }
512     {
513         label .code:n = {
514             \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
515             { \l__enumext_counter_#2_tl } {##1}
516             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
517             \l__enumext_current_widest_dim
518         },
519         label .initial:n = #3,
520         label .value_required:n = true,
521         ref .code:n = \__enumext_standar_ref:n {##1},
522         ref .value_required:n = true,
523     }
524 }
525 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
526 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
527 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
528 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

(End of definition for label and others.)

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

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

```

529 \cs_new_protected:Npn \__enumext_standar_ref:n #1
530 {
531     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
532     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
533     {
534         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
535     }
536     {
537         \tl_set_eq:Nc
538         \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
539         \__enumext_regex_counter_style:
540         \tl_set_eq:Nc
541         \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
542         \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
543         {
544             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
545             { \exp_not:V \l__enumext_ref_key_arg_tl }
546         }
547     }
548 }

```

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

```

549 \cs_new_protected:Nn \__enumext_standar_ref:
550 {

```

```

551 \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
552 {
553   \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
554 }
555 }

```

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

### 11.11.2 Define and set label and ref keys for enumext\* and keyans\* environments

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

```

\__enumext_label_vii_tl
\__enumext_label_viii_tl
556 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
557 {
558   \keys_define:nn { enumext / #1 }
559   {
560     label .code:n = {
561       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
562       { \__enumext_counter_#2_tl } {##1}
563       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
564       \__enumext_current_widest_dim
565     },
566     label .initial:n = #3,
567     label .value_required:n = true,
568     ref .code:n = \__enumext_starred_ref:n {##1},
569     ref .value_required:n = true,
570   }
571 }
572 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
573 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

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

```

574 \cs_new_protected:Npn \__enumext_starred_ref:n #1
575 {
576   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
577   \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
578   {
579     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
580     {
581       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
582     }
583     {
584       \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_vii_tl
585       \__enumext_regex_counter_style:
586       \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_vii_tl
587       \tl_put_right:Ne \__enumext_renew_the_count_vii_tl
588       {
589         \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
590         { \exp_not:V \__enumext_ref_key_arg_tl }
591       }
592     }
593   }
594   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
595   {
596     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
597     {
598       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
599     }
600     {
601       \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_viii_tl
602       \__enumext_regex_counter_style:
603       \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_viii_tl
604       \tl_put_right:Ne \__enumext_renew_the_count_viii_tl
605       {
606         \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
607         { \exp_not:V \__enumext_ref_key_arg_tl }
608       }
609     }
610   }
611 }

```

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

```

612 \cs_new_protected:Nn \__enumext_starred_ref:
613 {
614   \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
615   {
616     \tl_if_empty:NF \__enumext_renew_the_count_vii_tl
617     {
618       \tl_use:N \__enumext_renew_the_count_vii_tl
619     }
620   }
621   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
622   {
623     \tl_if_empty:NF \__enumext_renew_the_count_viii_tl
624     {
625       \tl_use:N \__enumext_renew_the_count_viii_tl
626     }
627   }
628 }

```

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

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

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

```

\__enumext_label_v_tl
\__enumext_label_vi_tl
629 \keys_define:nn { enumext / keyans }
630 {
631   label .code:n = {
632     \__enumext_label_style:cnv { \__enumext_label_v_tl }
633     { \__enumext_counter_v_tl } {#1}
634     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
635     \__enumext_current_widest_dim
636     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
637     { \__enumext_counter_vi_tl } {#1}
638     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
639     \__enumext_current_widest_dim
640   },
641   label .initial:n = (\Alph*),
642   label .value_required:n = true,
643   ref .code:n = \__enumext_keyans_ref:n {#1},
644   ref .value_required:n = true,
645 }

```

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

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

```

646 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
647 {
648   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
649   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
650   {
651     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
652   }
653   {
654     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_v_tl
655     \__enumext_regex_counter_style:
656     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
657     \tl_put_right:Ne \__enumext_renew_the_count_v_tl
658     {
659       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
660       { \exp_not:V \__enumext_ref_key_arg_tl }
661     }
662   }
663 }

```

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

```

664 \cs_new_protected:Nn \__enumext_keyans_ref:
665 {
666   \tl_if_empty:NF \__enumext_renew_the_count_v_tl

```

```

667     {
668         \tl_use:N \l__enumext_renew_the_count_v_tl
669     }
670 }

```

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

## 11.12 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.

```

671 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
672 {
673     \__enumext_if_is_int:nTF { #3 }
674     {
675         \int_set:Nn #2 {#3}
676     }
677     {
678         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
679         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
680         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
681         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
682     }
683 }
684 \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.

```

685 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
686 {
687     \__enumext_if_is_int:nTF {#4}
688     {
689         \setcounter{enumX#1} { #4 }
690     }
691     {
692         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
693         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
694         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
695         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
696     }
697     \__enumext_label_width_by_box:cv
698     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
699 }
700 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

```

start
widest
\l__enumext_start_X_int

```

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

```

701 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
702 {
703     \keys_define:nn { enumext / #1 }
704     {
705         start .code:n = {
706             \__enumext_start_from:ccn
707             { \l__enumext_label_#2_tl }
708             { \l__enumext_start_#2_int } {#1}

```



```

709         },
710         start .initial:n = 1,
711         widest .code:n = {
712             \__enumext_widest_from:nccn {#2}
713             { \__enumext_label_#2_tl }
714             { \__enumext_labelwidth_#2_dim } {##1}
715         },
716         widest .value_required:n = true,
717         start .value_required:n = true,
718     }
719 }
720 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 11.13 Setting keys for vertical spaces

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

```

721 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
722 {
723     \keys_define:nn { enumext / #1 }
724     {
725         topsep .skip_set:c = { \__enumext_topsep_#2_skip },
726         topsep .initial:n = {#3},
727         topsep .value_required:n = true,
728         partopsep .skip_set:c = { \__enumext_partopsep_#2_skip },
729         partopsep .initial:n = {#4},
730         partopsep .value_required:n = true,
731         parsep .skip_set:c = { \__enumext_parsep_#2_skip },
732         parsep .initial:n = {#5},
733         parsep .value_required:n = true,
734         itemsep .skip_set:c = { \__enumext_itemsep_#2_skip },
735         itemsep .initial:n = {#6},
736         itemsep .value_required:n = true,
737         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
738         noitemsep .value_forbidden:n = true,
739         nosep .meta:n = {
740             itemsep = 0pt, parsep = 0pt,
741             topsep = 0pt, partopsep = 0pt,
742         },
743         nosep .value_forbidden:n = true,
744     }
745 }

```

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

```

746 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
747 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
748 { 4.0pt plus 2.0pt minus 1.0pt }
749 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
750 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
751 { 2.0pt plus 1.0pt minus 1.0pt }
752 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
753 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
754 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
755 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
756 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
757 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
758 { 2.0pt plus 1.0pt minus 1.0pt }
759 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
760 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
761 { 4.0pt plus 2.0pt minus 1.0pt }
762 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
763 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
764 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

### 11.14 Setting keys for horizontal spaces

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

```

765 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
766 {
767   \keys_define:nn { enumext / #1 }
768   {
769     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
770     itemindent .value_required:n = true,
771     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
772     rightmargin .value_required:n = true,
773     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
774     listparindent .value_required:n = true,
775     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
776     list-offset .value_required:n = true,
777     list-indent .code:n =
778       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
779       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
780     list-indent .value_required:n = true,
781   }
782 }
783 \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.

```

784 \cs_set_protected:Npn \__enumext_tmp:n #1
785 {
786   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
787 }
788 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

789 \cs_set_protected:Nn \__enumext_fake_item:
790 {
791   \dim_compare:nNnT
792     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
793     >
794     { \c_zero_dim }
795   {
796     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
797     {
798       \exp_not:N \mode_leave_vertical:
799       \exp_not:n { \skip_horizontal:n
800         { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
801       \ignorespaces
802     }
803   }
804 }
805 \cs_set_protected:Nn \__enumext_keyans_fake_item:
806 {
807   \dim_compare:nNnT
808     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
809     {
810     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
811     {
812       \exp_not:N \mode_leave_vertical:
813       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
814     }
815   }
816 }
817 \cs_set_protected:Nn \__enumext_fake_item_vii:
818 {
819   \dim_compare:nNnT
820     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
821     {
822     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
823     {
824       \exp_not:N \mode_leave_vertical:

```

```

825         \exp_not:N \skip_horizontal:N \__enumext_fake_item_indent_vii_dim
826     }
827 }
828 }
829 \cs_set_protected:Nn \__enumext_fake_item_viii:
830 {
831     \dim_compare:nNnT
832     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
833     {
834         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
835         {
836             \exp_not:N \mode_leave_vertical:
837             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
838         }
839     }
840 }

```

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

### 11.15 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.

```

841 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
842 {
843     \keys_define:nn { enumext / #1 }
844     {
845         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
846         show-length .initial:n = false,
847     }
848 }
849 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 11.16 Setting before, after and first keys

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

```

before*
after
first
850 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
851 {
852     \keys_define:nn { enumext / #1 }
853     {
854         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
855         before .value_required:n = true,
856         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
857         before* .value_required:n = true,
858         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
859         after .value_required:n = true,
860         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
861         first .value_required:n = true,
862     }
863 }
864 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

#### 11.16.1 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.

```

865 \cs_new_protected:Nn \__enumext_before_args_exec:
866 {
867     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
868 }

```

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⟩`.

```

869 \cs_new_protected:Nn \__enumext_before_keys_exec:
870 {

```

```

871     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
872 }

```

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

```

873 \cs_new_protected:Nn \__enumext_after_stop_list:
874 {
875     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
876 }

```

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`.

```

877 \cs_new_protected:Nn \__enumext_after_args_exec:
878 {
879     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
880 }

```

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

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

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

```

881 \cs_new_protected:Nn \__enumext_before_args_exec_v:
882 {
883     \tl_use:N \l__enumext_before_starred_key_v_tl
884 }

```

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⟩`.

```

885 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
886 {
887     \tl_use:N \l__enumext_before_no_starred_key_v_tl
888 }

```

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

```

889 \cs_new_protected:Nn \__enumext_after_stop_list_v:
890 {
891     \tl_use:N \l__enumext_after_stop_list_v_tl
892 }

```

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`.

```

893 \cs_new_protected:Nn \__enumext_after_args_exec_v:
894 {
895     \tl_use:N \l__enumext_after_list_args_v_tl
896 }

```

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

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

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

```

897 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
898 {
899     \tl_use:N \l__enumext_before_starred_key_vii_tl
900 }
901 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
902 {
903     \tl_use:N \l__enumext_before_starred_key_viii_tl
904 }

```

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⟩`.

```

905 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
906 {
907     \tl_use:N \l__enumext_before_no_starred_key_vii_tl

```

```

908 }
909 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
910 {
911   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
912 }

```

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

```

913 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
914 {
915   \tl_use:N \l__enumext_after_stop_list_vii_tl
916 }
917 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
918 {
919   \tl_use:N \l__enumext_after_stop_list_viii_tl
920 }

```

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`.

```

921 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
922 {
923   \tl_use:N \l__enumext_after_list_args_vii_tl
924 }
925 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
926 {
927   \tl_use:N \l__enumext_after_list_args_viii_tl
928 }

```

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

## 11.17 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.

```

929 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
930 {
931   \keys_define:nn { enumext / #1 }
932   {
933     mini-env   .dim_set:c = { l__enumext_minipage_right_#2_dim },
934     mini-env   .value_required:n = true,
935     mini-sep   .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
936     mini-sep   .initial:n = 0.3333em,
937     mini-sep   .value_required:n = true,
938     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
939     columns-sep .value_required:n = true,
940     columns    .int_set:c = { l__enumext_columns_#2_int },
941     columns    .initial:n = 1,
942     columns    .value_required:n = true,
943   }
944 }
945 \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 `mini-right` and `mini-right*` to implement support for `minipage`.

```

946 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
947 {
948   \keys_define:nn { enumext / #1 }
949   {
950     columns    .initial:n = 2,
951     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
952     mini-right .value_required:n = true,
953     mini-right* .code:n = {
954       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
955       \keys_set:nn { enumext / #1 } { miniright = {##1} }
956     },
957     mini-right* .value_required:n = true,
958   }
959 }
960 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

### 11.18 Adjustment of vertical spaces for multicol

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

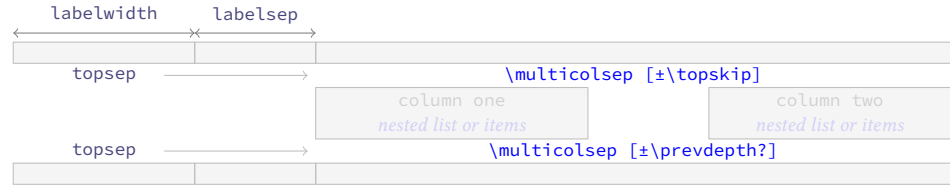


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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

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

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

`\__enumext_multi_set_vskip:`

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

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

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

```

961 \cs_new_protected:Nn \__enumext_multi_set_vskip:
962 {
963   \skip_set:cn { \l__enumext_multicol_above_ \__enumext_level: } _skip {
964     {
965       \skip_use:c { \l__enumext_topsep_ \__enumext_level: } _skip {
966       }
967     }
968   \skip_set:cn { \l__enumext_multicol_below_ \__enumext_level: } _skip {
969     {
970       \skip_use:c { \l__enumext_topsep_ \__enumext_level: } _skip {
971       }
972   }
973   \__enumext_add_pre_parsep:
974 }

```

(End of definition for `\__enumext_multi_set_vskip:`)

`\__enumext_add_pre_parsep:`

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

```

973 \cs_new_protected:Nn \__enumext_add_pre_parsep:
974 {
975   \int_case:nn { \l__enumext_level_int }
976   {
977     { 2 }{
978       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip } {
979         {
980           \skip_add:Nn \l__enumext_multicol_above_ii_skip { \l__enumext_parsep_i_skip }
981         }
982       }
983     { 3 }{
984       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip } {
985         {
986           \skip_add:Nn \l__enumext_multicol_above_iii_skip { \l__enumext_parsep_ii_skip }
987         }
988       }
989     { 4 }{
990       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip } {
991         {
992           \skip_add:Nn \l__enumext_multicol_above_iv_skip { \l__enumext_parsep_iii_skip }

```



```

993         }
994     }
995 }
996 }

```

(End of definition for `\__enumext_add_pre_parse:`)

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

```

997 \cs_new_protected:Nn \__enumext_multi_addvspace:
998 {
999     \__enumext_multi_set_vskip:
1000     \mode_if_vertical:T
1001     {
1002         \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1003         {
1004             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1005         }
1006         \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1007         {
1008             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1009         }
1010     }
1011     \par\nopagebreak
1012     \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1013 }

```

(End of definition for `\__enumext_multi_addvspace:`)

### 11.18.2 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:`

```

1014 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1015 {
1016     \skip_set:Nn \l__enumext_multicols_above_v_skip
1017     {
1018         \l__enumext_topsep_v_skip
1019     }
1020     \skip_set:Nn \l__enumext_multicols_below_v_skip
1021     {
1022         \l__enumext_topsep_v_skip
1023     }
1024 }
1025 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1026 {
1027     \__enumext_keyans_multi_set_vskip:
1028     \mode_if_vertical:T
1029     {
1030         \skip_add:Nn \l__enumext_multicols_above_v_skip
1031         {
1032             \skip_use:N \l__enumext_partopsep_v_skip
1033         }
1034         \skip_add:Nn \l__enumext_multicols_below_v_skip
1035         {
1036             \skip_use:N \l__enumext_partopsep_v_skip
1037         }
1038     }
1039     \par\nopagebreak
1040     \addvspace{ \l__enumext_multicols_above_v_skip }
1041 }

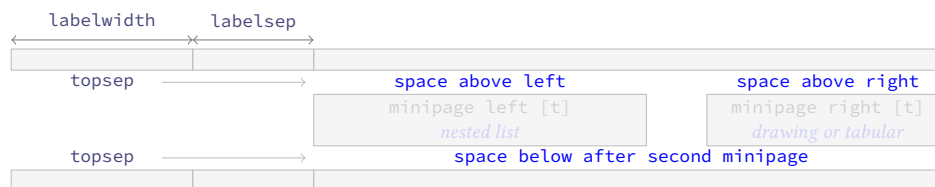
```

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

### 11.19 Adjustment of vertical spaces for minipage

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

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

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

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

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

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

```

1042 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
1043 {
1044     \__enumext_minipage:w [ t ] { #1 }
1045     \legacy_if_gset_false:n { @minipage }
1046     \vspace { 0pt }
1047 }
1048 { \__enumext_endminipage: }

```

(End of definition for `__enumext_mini_env*`.)

### 11.19.1 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`.

```

1049 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1050 {
1051     \int_compare:nNnTF
1052     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1053     {

```

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

```

1054     \skip_if_eq:nnTF
1055     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1056     {
1057         \skip_set:Nn \l__enumext_minipage_left_skip
1058         {
1059             -0.150\box_dp:N \strutbox
1060         }
1061         \skip_set:Nn \l__enumext_minipage_right_skip
1062         {
1063             0.695\box_dp:N \strutbox
1064         }
1065         \skip_set:Nn \l__enumext_minipage_after_skip
1066         {
1067             \box_dp:N \strutbox
1068         }
1069         \__enumext_zero_parsep:
1070     }

```

```

1071     {
1072         \skip_set:Nn \l__enumext_minipage_left_skip
1073         {
1074             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1075         }
1076         \skip_set:Nn \l__enumext_minipage_right_skip
1077         {
1078             0.695\box_dp:N \strutbox
1079         }
1080         \skip_set:Nn \l__enumext_minipage_after_skip
1081         {
1082             1.85\box_dp:N \strutbox
1083             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1084         }
1085     }
1086 }
1087 {

```

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.

```

1088     \skip_if_eq:nnTF
1089     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1090     {
1091         \skip_set:Nn \l__enumext_minipage_left_skip
1092         {
1093             0.5\box_dp:N \strutbox
1094             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1095         }
1096         \skip_set:Nn \l__enumext_minipage_right_skip
1097         {
1098             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1099         }
1100         \skip_set:Nn \l__enumext_minipage_after_skip
1101         {
1102             1.6\box_dp:N \strutbox
1103         }
1104     }
1105     {
1106         \skip_set:Nn \l__enumext_minipage_left_skip
1107         {
1108             0.5875\box_dp:N \strutbox
1109             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1110         }
1111         \skip_set:Nn \l__enumext_minipage_right_skip
1112         {
1113             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1114             + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1115         }
1116         \skip_set:Nn \l__enumext_minipage_after_skip
1117         {
1118             0.325\box_dp:N \strutbox
1119             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1120         }
1121     }
1122 }
1123 }

```

(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.

```

1124 \cs_new_protected:Nn \__enumext_zero_parsep:
1125 {
1126     \int_case:nn { \l__enumext_level_int }
1127     {
1128         { 2 }{
1129             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1130             {
1131                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }

```

```

1132         }
1133     }
1134     { 3 }{
1135         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1136         {
1137             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1138         }
1139     }
1140     { 4 }{
1141         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1142         {
1143             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1144         }
1145     }
1146 }
1147 }

```

(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*.

```

1148 \cs_new_protected:Nn \__enumext_mini_addvspace:
1149 {
1150     \__enumext_mini_set_vskip:
1151     \mode_if_vertical:T
1152     {
1153         \skip_add:Nn \l__enumext_minipage_left_skip
1154         {
1155             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1156         }
1157         \skip_add:Nn \l__enumext_minipage_after_skip
1158         {
1159             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1160         }
1161     }
1162     \par\nopagebreak
1163     \addvspace { \l__enumext_minipage_left_skip }
1164 }

```

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

### 11.19.2 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`.

```

1165 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1166 {
1167     \skip_zero_new:N \l__enumext_minipage_after_skip
1168     \skip_zero_new:N \l__enumext_minipage_left_skip
1169     \skip_zero_new:N \l__enumext_minipage_right_skip
1170     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1171     {
1172         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1173         {
1174             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1175             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1176             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1177             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1178             {
1179                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1180             }
1181         }
1182     }
1183     {
1184         \skip_set:Nn \l__enumext_minipage_left_skip
1185         {
1186             \skip_use:N \l__enumext_topsep_v_skip
1187         }
1188         \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1188         {
1189             0.705\box_dp:N \strutbox
1190         }
1191         \skip_set:Nn \l__enumext_minipage_after_skip
1192         {
1193             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1194         }
1195     }
1196 }
1197 {
1198     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1199     {
1200         \skip_set:Nn \l__enumext_minipage_left_skip
1201         {
1202             0.5\box_dp:N \strutbox
1203             + \l__enumext_partopsep_v_skip
1204         }
1205         \skip_set:Nn \l__enumext_minipage_right_skip
1206         {
1207             \l__enumext_partopsep_v_skip
1208         }
1209         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1210     }
1211     {
1212         \skip_set:Nn \l__enumext_minipage_left_skip
1213         {
1214             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1215         }
1216         \skip_set:Nn \l__enumext_minipage_right_skip
1217         {
1218             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1219         }
1220         \skip_set:Nn \l__enumext_minipage_after_skip
1221         {
1222             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1223         }
1224     }
1225 }
1226 }

```

(End of definition for `\l__enumext_keyans_mini_set_vskip:`)

`\l__enumext_keyans_mini_addvspace:`

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

```

1227 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1228 {
1229     \l__enumext_keyans_mini_set_vskip:
1230     \mode_if_vertical:T
1231     {
1232         \skip_add:Nn \l__enumext_minipage_left_skip
1233         {
1234             \l__enumext_partopsep_v_skip
1235         }
1236         \skip_add:Nn \l__enumext_minipage_after_skip
1237         {
1238             \l__enumext_partopsep_v_skip
1239         }
1240     }
1241     \par\nopagebreak
1242     \addvspace { \l__enumext_minipage_left_skip }
1243 }

```

(End of definition for `\l__enumext_keyans_mini_addvspace:`)

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

`\l__enumext_mini_set_vskip_vii:`

`\l__enumext_mini_set_vskip_viii:`

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

```

1244 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1245 {
1246   \skip_zero_new:N \l__enumext_minipage_left_skip
1247   \skip_gzero_new:N \g__enumext_minipage_right_skip
1248   \skip_gzero_new:N \g__enumext_minipage_after_skip
1249   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1250   {
1251     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1252     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1253   }
1254   {
1255     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1256     \skip_gset:Nn \g__enumext_minipage_right_skip
1257     {
1258       \l__enumext_topsep_vii_skip
1259     }
1260     \skip_gset:Nn \g__enumext_minipage_after_skip
1261     {
1262       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1263     }
1264   }
1265 }
1266 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1267 {
1268   \skip_zero_new:N \l__enumext_minipage_after_skip
1269   \skip_zero_new:N \l__enumext_minipage_left_skip
1270   \skip_zero_new:N \l__enumext_minipage_right_skip
1271   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1272   {
1273     \skip_set:Nn \l__enumext_minipage_left_skip
1274     {
1275       0.5\box_dp:N \strutbox
1276     }
1277     \skip_set:Nn \l__enumext_minipage_right_skip
1278     {
1279       \l__enumext_partopsep_viii_skip
1280     }
1281     \skip_set:Nn \l__enumext_minipage_after_skip
1282     {
1283       1.6\box_dp:N \strutbox
1284     }
1285   }
1286   {
1287     \skip_set:Nn \l__enumext_minipage_left_skip
1288     {
1289       0.5875\box_dp:N \strutbox
1290     }
1291     \skip_set:Nn \l__enumext_minipage_right_skip
1292     {
1293       \l__enumext_topsep_viii_skip
1294     }
1295     \skip_set:Nn \l__enumext_minipage_after_skip
1296     {
1297       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1298     }
1299   }
1300 }

```

(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 `mini-right` 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.

```

1301 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1302 {
1303   \__enumext_mini_set_vskip_vii:
1304   \par\nopagebreak
1305   \addvspace { \l__enumext_minipage_left_skip }
1306 }

```

```

1307 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1308 {
1309     \__enumext_mini_set_vskip_viii:
1310     \par\nopagebreak
1311     \addvspace { \__enumext_minipage_left_skip }
1312 }

```

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

#### 11.19.4 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.

```

1313 \NewDocumentCommand \miniright { s }
1314 {
1315     \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1316     {
1317         \msg_error:nnn { enumext } { wrong-miniright-place }
1318     }
1319     \int_compare:nNt { \__enumext_level_int } = { 0 }
1320     {
1321         \msg_error:nnn { enumext } { wrong-miniright-place }
1322     }
1323     \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1324     {
1325         \__enumext_keyans_mini_right_cmd:n {#1}
1326     }
1327     { \__enumext_mini_right_cmd:n {#1} }
1328 }

```

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

`\__enumext_mini_right_cmd:n` The function `\__enumext_mini_right_cmd:n` takes as argument the *starred version* ‘\*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\__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.

```

1329 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1330 {
1331     \dim_compare:nNtF
1332     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1333     {
1334         \__enumext_multicols_stop:
1335         \end{__enumext_mini_env*}
1336         \hfill
1337         \begin{__enumext_mini_env*}
1338         { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1339         \par\addvspace { \__enumext_minipage_right_skip }
1340         \bool_if:nF {#1}
1341         {
1342             \centering
1343         }
1344         \int_gzero:N \g__enumext_minipage_stat_int
1345     }
1346     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1347 }

```

(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.

```

1348 \\cs_new_protected:Npn \\_enumext_keyans_mini_right_cmd:n #1
1349 {
1350   \\dim_compare:nNnTF { \\_enumext_minipage_right_v_dim } > { \\c_zero_dim }
1351   {
1352     \\_enumext_keyans_multicols_stop:
1353     \\end{\\_enumext_mini_env*}
1354     \\hfill
1355     \\begin{\\_enumext_mini_env*}{ \\_enumext_minipage_right_v_dim }
1356     \\par\\addvspace { \\_enumext_minipage_right_skip }
1357     \\bool_if:nF {#1}
1358     {
1359       \\centering
1360     }
1361     \\int_gzero:N \\g__enumext_minipage_stat_int
1362   }
1363   { \\msg_error:nnn { enumext } { wrong-miniright-use } }
1364 }

```

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

## 11.20 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.

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

```

above*
below
below*
1365 \\cs_set_protected:Npn \\_enumext_tmp:nn #1 #2
1366 {
1367   \\keys_define:nn { enumext / #1 }
1368   {
1369     above .skip_set:c = { \\_enumext_vspace_above_#2_skip },
1370     above .value_required:n = true,
1371     above* .code:n = \\bool_set_true:c { \\_enumext_vspace_a_star_#2_bool }
1372     \\keys_set:nn { enumext / #1 } { above = {##1} },
1373     above* .value_required:n = true,
1374     below .skip_set:c = { \\_enumext_vspace_below_#2_skip },
1375     below .value_required:n = true,
1376     below* .code:n = \\bool_set_true:c { \\_enumext_vspace_b_star_#2_bool }
1377     \\keys_set:nn { enumext / #1 } { below = {##1} },
1378     below* .value_required:n = true,
1379   }
1380 }
1381 \\clist_map_inline:Nn \\c__enumext_all_envs_clist { \\_enumext_tmp:nn #1 }

```

(End of definition for above and others.)

### 11.20.1 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.

```

1382 \\cs_new_protected:Nn \\_enumext_vspace_above:
1383 {
1384   \\skip_if_eq:nnF
1385   { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } } { \\c_zero_skip }
1386   {
1387     \\bool_if:cTF { \\_enumext_vspace_a_star_ \\_enumext_level: _bool }
1388     {
1389       \\vspace*{ \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1390     }
1391     {
1392       \\vspace { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1393     }
1394   }
1395 }

```

(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.

```

1396 \cs_new_protected:Nn \__enumext_vspace_below:
1397 {
1398   \skip_if_eq:nnF
1399   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1400   {
1401     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1402     {
1403       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1404     }
1405     {
1406       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1407     }
1408   }
1409 }

```

(End of definition for `\__enumext_vspace_below`.)

### 11.20.2 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.

```

1410 \cs_new_protected:Nn \__enumext_vspace_above_v:
1411 {
1412   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1413   {
1414     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1415     {
1416       \vspace*{ \l__enumext_vspace_above_v_skip }
1417     }
1418     { \vspace { \l__enumext_vspace_above_v_skip } }
1419   }
1420 }

```

(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.

```

1421 \cs_new_protected:Nn \__enumext_vspace_below_v:
1422 {
1423   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1424   {
1425     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1426     {
1427       \vspace*{ \l__enumext_vspace_below_v_skip }
1428     }
1429     { \vspace { \l__enumext_vspace_below_v_skip } }
1430   }
1431 }

```

(End of definition for `\__enumext_vspace_below_v`.)

### 11.20.3 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`:

```

1432 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1433 {
1434   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1435   {
1436     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1437     {
1438       \vspace*{ \l__enumext_vspace_above_vii_skip }
1439     }
1440     { \vspace { \l__enumext_vspace_above_vii_skip } }
1441   }
1442 }
1443 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1444 {
1445   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1446   {

```

```

1447         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1448         {
1449             \vspace*{ \l__enumext_vspace_above_viii_skip }
1450         }
1451         { \vspace { \l__enumext_vspace_above_viii_skip } }
1452     }
1453 }

```

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

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

```

1454 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1455 {
1456     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1457     {
1458         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1459         {
1460             \vspace*{ \l__enumext_vspace_below_vii_skip }
1461         }
1462         { \vspace { \l__enumext_vspace_below_vii_skip } }
1463     }
1464 }
1465 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1466 {
1467     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1468     {
1469         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1470         {
1471             \vspace*{ \l__enumext_vspace_below_viii_skip }
1472         }
1473         { \vspace { \l__enumext_vspace_below_viii_skip } }
1474     }
1475 }

```

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

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

The `series` key is responsible for the whole process of the `resume` and `resume*` keys. The idea behind this is to be able to absorb the *(keys)* passed to the optional argument of the “*first level*” of the environments `enumext` and `enumext*`, but, discarding some specific *(keys)*. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in `chat-Tex-SX`

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1476 \cs_set_protected:Npn \__enumext_tmp:n #1
1477 {
1478     \keys_define:nn { enumext / #1 }
1479     {
1480         series .str_set:N = \l__enumext_series_str,
1481         series .value_required:n = true,
1482         resume .code:n = \__enumext_resume_series:n {##1},
1483         resume* .code:n = \__enumext_resume_starred:,
1484         resume* .value_forbidden:n = true,
1485     }
1486 }
1487 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

#### 11.21.1 Internal functions for series key

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

```

1488 \cs_new:Npn \__enumext_filter_series:n #1
1489 {
1490     \use:e
1491     {
1492         \keyval_parse:NNn
1493         \__enumext_filter_series_key:n
1494         \__enumext_filter_series_pair:nn {#1}
1495     }
1496 }

```

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

```

1497 \cs_new:Npn \__enumext_filter_series_key:n #1
1498 {
1499   \str_case:nnF {#1}
1500   {
1501     { resume } {}
1502     { resume* } {}
1503   }
1504   { , { \exp_not:n {#1} } }
1505 }

```

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

```

1506 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1507 {
1508   \str_case:nnF {#1}
1509   {
1510     { series } {}
1511     { resume } {}
1512     { start } {}
1513     { save-ans } {}
1514     { save-key } {}
1515   }
1516   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1517 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `\__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *⟨keys⟩*. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§11.31) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.34).

```

1518 \cs_new_protected:Npn \__enumext_parse_series:n #1
1519 {
1520   \str_if_empty:NTF \l__enumext_series_str
1521   {
1522     \bool_if:NF \l__enumext_resume_active_bool
1523     {
1524       \__enumext_resume_last:n {#1}
1525     }
1526   }
1527   {
1528     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1529     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1530     { \__enumext_filter_series:n {#1} }
1531     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1532     {
1533       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1534     }
1535   }
1536 }

```

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

```

1537 \cs_new_protected:Npn \__enumext_resume_last:n #1
1538 {
1539   \bool_if:NT \l__enumext_standar_first_bool
1540   {
1541     \tl_gclear:N \g__enumext_standar_series_tl
1542     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }

```

```

1543     }
1544     \bool_if:NT \l__enumext_starred_first_bool
1545     {
1546         \tl_gclear:N \g__enumext_starred_series_tl
1547         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1548     }
1549 }

```

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

### 11.21.2 Internal function to save counter value

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

• The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§11.31) and the `enumext*` environment definition (§11.34).

```

1550 \cs_new_protected:Nn \__enumext_resume_save_counter:
1551 {
1552     \bool_if:NT \g__enumext_standar_bool
1553     {
1554         \tl_if_empty:NF \l__enumext_series_str
1555         {
1556             \int_gset_eq:cN
1557             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1558         }
1559         \tl_if_empty:NTF \l__enumext_resume_name_tl
1560         {
1561             \str_if_empty:NT \l__enumext_series_str
1562             {
1563                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1564             }
1565         }
1566         {
1567             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1568             {
1569                 \int_gset_eq:cN
1570                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1571             }
1572         }
1573         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1574         {
1575             \int_gset_eq:cN
1576             { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1577         }
1578     }
1579     \bool_if:NT \g__enumext_starred_bool
1580     {
1581         \tl_if_empty:NF \l__enumext_series_str
1582         {
1583             \int_gset_eq:cN
1584             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1585         }
1586         \tl_if_empty:NTF \l__enumext_resume_name_tl
1587         {
1588             \str_if_empty:NT \l__enumext_series_str
1589             {
1590                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1591             }
1592         }
1593         {
1594             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1595             {
1596                 \int_gset_eq:cN
1597                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1598             }
1599         }
1600     }

```

```

1599     }
1600     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1601     {
1602         \int_gset_eq:cN
1603         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1604     }
1605 }
1606 }

```

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

### 11.21.3 Internal functions for resume key

\\_\_enumext\_resume\_series:n

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

```

1607 \cs_new_protected:Npn \__enumext_resume_series:n #1
1608 {
1609     \tl_if_empty:NTF {#1}
1610     {
1611         \__enumext_resume_counter:n { }
1612     }
1613     {
1614         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1615         {
1616             \__enumext_resume_counter:n {#1}
1617             \bool_if:NT \g__enumext_standar_bool
1618             {
1619                 \keys_set:nv { enumext / level-1 }
1620                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1621             }
1622             \bool_if:NT \g__enumext_starred_bool
1623             {
1624                 \keys_set:nv { enumext / enumext* }
1625                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1626             }
1627         }
1628         {
1629             \bool_if:NT \g__enumext_standar_bool
1630             {
1631                 \msg_error:nnn { enumext } { unknown-series } {#1}
1632             }
1633             \bool_if:NT \g__enumext_starred_bool
1634             {
1635                 \msg_error:nnn { enumext } { unknown-series } {#1}
1636             }
1637         }
1638     }
1639 }

```

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

\\_\_enumext\_resume\_counter:n

\\_\_enumext\_resume\_counter:

\\_\_enumext\_resume\_counter\_series:

\\_\_enumext\_resume\_counter\_save\_ans:

The function \\_\_enumext\_resume\_counter:n will set the variable \l\_\_enumext\_resume\_active\_bool to true and pass the value of the key `resume` to the variable \l\_\_enumext\_series\_name\_tl which will contain the `{⟨series name⟩}`. If the variable \l\_\_enumext\_series\_name\_tl is empty, that is, we are passing the key `resume` *without value*, we will execute the function \\_\_enumext\_resume\_counter: otherwise, when we pass `resume={⟨series name⟩}` we will execute the function \\_\_enumext\_resume\_counter\_series:, finally we will execute the function \\_\_enumext\_resume\_counter\_save\_ans: which is associated with the key `save-ans`.

```

1640 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1641 {
1642     \bool_set_true:N \l__enumext_resume_active_bool
1643     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1644     \tl_if_empty:NTF \l__enumext_resume_name_tl
1645     {
1646         \__enumext_resume_counter:
1647     }

```

```

1648     {
1649         \__enumext_resume_counter_series:
1650     }
1651     \__enumext_resume_counter_save_ans:
1652 }

```

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

```

1653 \cs_new_protected:Nn \__enumext_resume_counter:
1654 {
1655     \bool_if:NT \g__enumext_standar_bool
1656     {
1657         \int_gincr:N \g__enumext_resume_int
1658         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1659     }
1660     \bool_if:NT \g__enumext_starred_bool
1661     {
1662         \int_gincr:N \g__enumext_resume_vii_int
1663         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1664     }
1665 }

```

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

```

1666 \cs_new_protected:Nn \__enumext_resume_counter_series:
1667 {
1668     \bool_if:NT \g__enumext_standar_bool
1669     {
1670         \int_set:Nn \l__enumext_start_i_int
1671         {
1672             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1673         }
1674     }
1675     \bool_if:NT \g__enumext_starred_bool
1676     {
1677         \int_set:Nn \l__enumext_start_vii_int
1678         {
1679             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1680         }
1681     }
1682 }

```

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

```

1683 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1684 {
1685     \bool_lazy_and:nnT
1686     { \bool_if_p:N \l__enumext_standar_first_bool }
1687     { \bool_if_p:N \l__enumext_store_active_bool }
1688     {
1689         \int_set:Nn \l__enumext_start_i_int
1690         {
1691             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1692         }
1693     }
1694     \bool_lazy_and:nnT
1695     { \bool_if_p:N \l__enumext_starred_first_bool }
1696     { \bool_if_p:N \l__enumext_store_active_bool }
1697     {
1698         \int_set:Nn \l__enumext_start_vii_int
1699         {
1700             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1701         }
1702     }
1703 }

```

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



#### 11.21.4 Internal function for resume\* key

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

```

1704 \cs_new_protected:Nn \__enumext_resume_starred:
1705 {
1706   \bool_if:NT \g__enumext_standar_bool
1707   {
1708     \tl_if_empty:NF \g__enumext_standar_series_tl
1709     {
1710       \__enumext_resume_counter:n { }
1711       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1712     }
1713   }
1714   \bool_if:NT \g__enumext_starred_bool
1715   {
1716     \tl_if_empty:NF \g__enumext_starred_series_tl
1717     {
1718       \__enumext_resume_counter:n { }
1719       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1720     }
1721   }
1722 }

```

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

#### 11.22 Setting save-ans, check-ans and no-store keys

The key `save-ans` is directly associated with the keys `check-ans`, `no-store`, `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

##### 11.22.1 Setting save-ans key

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

```

1723 \cs_set_protected:Npn \__enumext_tmp:n #1
1724 {
1725   \keys_define:nn { enumext / #1 }
1726   {
1727     save-ans .code:n = \__enumext_storing_set:n {##1},
1728     save-ans .value_required:n = true,
1729   }
1730 }
1731 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

##### 11.22.2 Internal functions for save-ans key

`\__enumext_start_save_ans_msg:`  
`\__enumext_stop_save_ans_msg:` The functions `\__enumext_start_save_ans_msg:` and `\__enumext_stop_save_ans_msg:` will display in the terminal and `.log` file the environment in which the `save-ans` key was executed along with the line at the beginning and end of it. The function `\__enumext_start_save_ans_msg:` will be passed to `\__enumext_storing_set:n` and the function `\__enumext_stop_save_ans_msg:` will be passed to the function `\__enumext_execute_after_env:`.

```

1732 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1733 {
1734   \msg_term:nnVV { enumext } { save-ans-log }
1735   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1736 }
1737 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1738 {
1739   \msg_term:nnVV { enumext } { save-ans-log-hook }
1740   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1741 }

```

(End of definition for `\__enumext_start_save_ans_msg:` and `\__enumext_stop_save_ans_msg:`.)

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

```

1742 \cs_new_protected:Npn \__enumext_storing_set:n #1
1743 {
1744   \tl_set:Nx \l__enumext_store_name_tl {#1}
1745   \tl_if_empty:NTF \l__enumext_store_name_tl
1746   {
1747     \bool_lazy_or:nnT
1748     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1749     {
1750       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1751     }
1752   }
1753   {
1754     \bool_lazy_or:nnT
1755     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1756     {
1757       \__enumext_start_save_ans_msg:
1758       \__enumext_storing_exec:
1759     }
1760   }
1761 }

```

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

```

1762 \cs_new_protected:Npn \__enumext_storing_exec:
1763 {
1764   \bool_set_true:N \l__enumext_store_active_bool
1765   \bool_set_true:N \l__enumext_check_answers_bool
1766   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1767   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1768   {
1769     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1770     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1771   }
1772   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1773   {
1774     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1775     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1776   }
1777   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1778   {
1779     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1780     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1781   }
1782 }

```

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

### 11.22.3 The check answer mechanism

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

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

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

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

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

### 3. Keeping track of the number of times the environment nests.

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

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

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

#### 11.22.4 Setting check-ans and no-store keys

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

check-ans

no-store

```

1783 \cs_set_protected:Npn \__enumext_tmp:n #1
1784 {
1785   \keys_define:nn { enumext / #1 }
1786   {
1787     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1788     check-ans .initial:n = false,
1789     check-ans .value_required:n = true,
1790     no-store .code:n = {
1791       \bool_set_false:N \l__enumext_check_answers_bool
1792       \bool_set_false:N \l__enumext_check_ans_key_bool
1793     },
1794     no-store .value_forbidden:n = true,
1795   }
1796 }
1797 \clist_map_inline:nn
1798 {
1799   level-1, level-2, level-3, level-4, enumext*
1800 }
1801 { \__enumext_tmp:n {#1} }
```

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

#### 11.22.5 Set-up check answer mechanism

The function `\__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `\__enumext_check_ans_level:` only if “true”, i.e. the key `no-store` is not active.

```

1802 \cs_new_protected:Nn \__enumext_check_ans_active:
1803 {
1804   \tl_if_empty:NF \l__enumext_store_name_tl
1805   {
1806     \bool_if:NT \l__enumext_check_answers_bool
1807     {
1808       \__enumext_check_ans_level:
1809     }
1810   }
1811 }
```

The function `\__enumext_check_ans_level:` will decrement by “one” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment `enumext`, taking into account whether it is nested within `enumext*` or the opposite.

```

1812 \cs_new_protected:Nn \__enumext_check_ans_level:
1813 {
1814   \int_case:nn { \l__enumext_level_int }
1815   {
1816     { 1 }{
1817       \bool_lazy_all:nT
1818       {
1819         { \bool_if_p:N \g__enumext_starred_bool }
1820         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1821       }
1822       {
1823         \int_gdecr:N \g__enumext_item_number_int
1824       }
1825     }
1826   }
```

```

1826         { 2 }{
1827             \int_gdecr:N \g__enumext_item_number_int
1828         }
1829         { 3 }{
1830             \int_gdecr:N \g__enumext_item_number_int
1831         }
1832         { 4 }{
1833             \int_gdecr:N \g__enumext_item_number_int
1834         }
1835     }

```

We should only execute this if `enumext*` is nested in the first level of `enumext`, for the rest of the cases the value of `\g__enumext_item_number_int` is already decreased.

```

1836     \int_case:nn { \l__enumext_level_h_int }
1837     {
1838         { 1 }{
1839             \bool_lazy_all:nT
1840             {
1841                 { \bool_if_p:N \g__enumext_standar_bool }
1842                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1843             }
1844             {
1845                 \int_gdecr:N \g__enumext_item_number_int
1846             }
1847         }
1848     }
1849 }

```

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

`\__enumext_check_ans_key_hook:`

The function `\__enumext_check_ans_key_hook:` will *export* the status of the local variable `\l__enumext_check_ans_key_bool` to the global variable `\g__enumext_check_ans_key_bool` only if the key `check-ans` is active.

```

1850 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1851 {
1852     \bool_lazy_and:nnT
1853     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1854     { \bool_if_p:N \g__enumext_standar_bool }
1855     {
1856         \bool_gset_true:N \g__enumext_check_ans_key_bool
1857     }
1858     \bool_lazy_and:nnT
1859     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1860     { \bool_if_p:N \g__enumext_starred_bool }
1861     {
1862         \bool_gset_true:N \g__enumext_check_ans_key_bool
1863     }
1864 }

```

(End of definition for `\__enumext_check_ans_key_hook:`)

`\__enumext_item_answer_diff:`

The function `\__enumext_item_answer_diff:` will set the value of the variable `\g__enumext_item_answer_diff_int` which is used by the functions `\__enumext_check_ans_show:` for the key `save-ans` and by the function `\__enumext_check_ans_log:` by the internal “*check answer*” mechanism. This function will be passed to the function `\__enumext_execute_after_env:`.

```

1865 \cs_new_protected:Nn \__enumext_item_answer_diff:
1866 {
1867     \int_gset:Nn \g__enumext_item_answer_diff_int
1868     {
1869         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1870     }
1871 }

```

(End of definition for `\__enumext_item_answer_diff:`)

`\__enumext_check_ans_show:`

`\__enumext_check_ans_msg_less:`  
`\__enumext_check_ans_msg_same_ok:`  
`\__enumext_check_ans_msg_greater:`

The function `\__enumext_check_ans_show:` will be executed within the function `\__enumext_execute_after_env:` when the key `check-ans` is active, that is, when `\g__enumext_check_ans_key_bool` is “*true*” and will return the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `\__enumext_item_answer_diff:`.

```

1872 \cs_new_protected:Nn \__enumext_check_ans_show:
1873 {

```

```

1874     \int_case:nn { \g__enumext_item_answer_diff_int }
1875     {
1876         { -1 } { \__enumext_check_ans_msg_less: }
1877         { 0 } { \__enumext_check_ans_msg_same_ok: }
1878         { 1 } { \__enumext_check_ans_msg_greater: }
1879     }
1880 }
1881 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1882 {
1883     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1884     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1885 }
1886 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1887 {
1888     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1889     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1890 }
1891 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1892 {
1893     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1894     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1895 }

```

(End of definition for \\_\_enumext\_check\_ans\_show: and others.)

\\_\_enumext\_check\_ans\_log: The function \\_\_enumext\_check\_ans\_log: will be executed within the function \\_\_enumext\_execute\_after\_env: when the key `check-ans` is not active, that is, when \g\_\_enumext\_check\_ans\_key\_bool is “false” and write in the log the appropriate message according to the value of \g\_\_enumext\_item\_answer\_diff\_int set by the function \\_\_enumext\_item\_answer\_diff:.

```

1896 \cs_new_protected:Nn \__enumext_check_ans_log:
1897 {
1898     \int_case:nn { \g__enumext_item_answer_diff_int }
1899     {
1900         { -1 } { \__enumext_check_ans_log_msg_less: }
1901         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1902         { 1 } { \__enumext_check_ans_log_msg_greater: }
1903     }
1904 }
1905 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1906 {
1907     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1908     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1909 }
1910 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1911 {
1912     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1913     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1914 }
1915 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1916 {
1917     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1918     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1919 }

```

(End of definition for \\_\_enumext\_check\_ans\_log: and others.)

### 11.22.6 Writing .log and executing the check-ans key

\\_\_enumext\_execute\_after\_env: The \\_\_enumext\_execute\_after\_env: function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the \\_\_enumext\_item\_answer\_diff: function and then will write the values of the global variables used to the .log file. If the key `check-ans` is active it will execute the function \\_\_enumext\_check\_ans\_show: and show the result in the terminal, otherwise it will execute the function \\_\_enumext\_check\_ans\_log: and write the results in the .log file will finally execute the function \\_\_enumext\_reset\_global\_vars: returning the used variables to their original state. As this function is passed to the function \\_\_enumext\_after\_env:nn for the environments `enumext` and `enumext*` we must make sure that we are not nested at any level.

```

1920 \cs_new_protected:Nn \__enumext_execute_after_env:
1921 {
1922     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1923     {
1924         \tl_if_empty:NF \g__enumext_store_name_tl

```

```

1925     {
1926         \__enumext_stop_save_ans_msg:
1927         \__enumext_item_answer_diff:
1928         \__enumext_log_global_vars:
1929         \__enumext_log_answer_vars:
1930         \bool_if:NTF \g__enumext_check_ans_key_bool
1931         {
1932             \__enumext_check_ans_show:
1933         }
1934         { \__enumext_check_ans_log: }
1935     }
1936     \__enumext_reset_global_vars:
1937 }
1938 }

```

(End of definition for \\_\_enumext\_execute\_after\_env:.)

### 11.22.7 Check for \item\* and \anspic\* commands

\\_\_enumext\_check\_starred\_cmd:n

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

```

1939 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1940 {
1941     \int_compare:nNnT
1942     { \g__enumext_check_starred_cmd_int } = { 0 }
1943     {
1944         \msg_warning:nnnV
1945         { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1946     }
1947     \int_compare:nNnT
1948     { \g__enumext_check_starred_cmd_int } > { 1 }
1949     {
1950         \msg_warning:nnnV
1951         { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1952     }
1953     \int_gzero:N \g__enumext_check_starred_cmd_int
1954     \tl_clear:N \l__enumext_check_start_line_env_tl
1955 }

```

(End of definition for \\_\_enumext\_check\_starred\_cmd:n.)

### 11.23 Keys and functions associated with storage

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

```

1956 \cs_set_protected:Npn \__enumext_tmp:n #1
1957 {
1958     \keys_define:nn { enumext / #1 }
1959     {
1960         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1961         wrap-ans .initial:n = \fbox{##1},
1962         wrap-ans .value_required:n = true,
1963         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1964         wrap-opt .initial:n = [{##1}],
1965         wrap-opt .value_required:n = true,
1966         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1967         save-sep .initial:n = {, ~ },
1968         save-sep .value_required:n = true,
1969         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1970         mark-ans .initial:n = \textasteriskcentered,
1971         mark-ans .value_required:n = true,
1972         mark-pos .choice:,
1973         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1974         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1975         mark-pos .initial:n = right,
1976         mark-pos .value_required:n = true,
1977         show-ans .bool_set:N = \l__enumext_show_answer_bool,
1978         show-ans .initial:n = false,
1979         show-ans .value_required:n = true,
1980         show-pos .bool_set:N = \l__enumext_show_position_bool,

```

```

1981     show-pos    .initial:n = false,
1982     show-pos    .value_required:n = true,
1983     mark-ref     .tl_set:N = \__enumext_mark_ref_sym_tl,
1984     mark-ref     .initial:n = \textasteriskcentered,
1985     mark-ref     .value_required:n = true,
1986     save-ref     .bool_set:N = \__enumext_store_ref_key_bool,
1987     save-ref     .initial:n = false,
1988     save-ref     .value_required:n = true,
1989   }
1990 }
1991 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

1992 \cs_set_protected:Npn \__enumext_tmp:n #1
1993 {
1994   \keys_define:nn { enumext / #1 }
1995   {
1996     mark-pos .choice:,
1997     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1998     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1999     mark-pos .initial:n = right,
2000     mark-pos .value_required:n = true,
2001     show-ans .bool_set:N = \__enumext_show_answer_bool,
2002     show-ans .initial:n = false,
2003     show-ans .value_required:n = true,
2004     show-pos .bool_set:N = \__enumext_show_position_bool,
2005     show-pos .initial:n = false,
2006     show-pos .value_required:n = true,
2007   }
2008 }
2009 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

### 11.23.1 Store optional arguments of the environments

The idea behind “*storing*” in the *⟨sequence⟩* is to have a copy of the structure of the environment in which the key **save-ans** is being executed so we must capture the optional arguments passed to the levels of the environment in which it is executed and “*storing*” them.

```

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

```

The functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n` will be responsible for “*storing*” the *⟨keys⟩* filtered from the optional arguments of the environment in which the key **save-ans** is executed and the levels within this for the **enumext** and **enumext\*** environments. We will execute this function only if the variable `\__enumext_store_save_key_X_bool` is false, that is, the key **store-key** is not active, establishing the variable `\__enumext_store_save_key_X_tl` with the filtered *⟨keys⟩*.

```

2010 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2011 {
2012   \bool_if:cF { \__enumext_store_save_key_ \__enumext_level: _bool }
2013   {
2014     \tl_clear:c { \__enumext_save_key_ \__enumext_level: _tl }
2015     \tl_set:ce
2016       { \__enumext_store_save_key_ \__enumext_level: _tl }
2017       { \__enumext_filter_save_key:n {#1} }
2018   }
2019 }
2020 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2021 {
2022   \bool_if:NF \__enumext_store_save_key_vii_bool
2023   {
2024     \tl_clear:N \__enumext_store_save_key_vii_tl
2025     \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2026   }
2027 }

```

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



### 11.23.2 Setting save-key key

Since this list structure will be stored in the *⟨sequence⟩* established by the `save-ans` key when executing `\anskey`, we will not be able to modify it. The best thing here is to have a key that allows you to modify the optional argument of the list stored in the *⟨sequence⟩*.

`save-key` The values set by this key passed in the optional arguments of the `enumext` and `enumext*` environments will override the values of the `\l__enumext_store_save_key_X_tl` variable set by the functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`. Define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2028 \cs_set_protected:Npn \__enumext_tmp:n #1
2029 {
2030   \keys_define:nn { enumext / enumext* }
2031   {
2032     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2033     save-key .value_required:n = true,
2034   }
2035   \keys_define:nn { enumext / #1 }
2036   {
2037     save-key .code:n = \__enumext_parse_save_key:n {##1},
2038     save-key .value_required:n = true,
2039   }
2040 }
2041 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_parse_save_key:n
\__enumext_parse_save_key_vii:n

```

The functions `\__enumext_parse_save_key:n` and `\__enumext_parse_save_key_vii:n` will be responsible for storing the filtered *⟨keys⟩* in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2042 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2043 {
2044   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2045   \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2046   \tl_set:ce
2047   { l__enumext_store_save_key_ \__enumext_level: _tl }
2048   { \__enumext_filter_save_key:n {#1} }
2049 }
2050 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2051 {
2052   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2053   \tl_clear:N \l__enumext_store_save_key_vii_tl
2054   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2055 }

```

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

### 11.23.3 Internal functions to store optional arguments

```

\__enumext_filter_save_key:n
\__enumext_filter_save_key_key:n
\__enumext_filter_save_key_pair:nn

```

The function `\__enumext_filter_save_key:n` will be in charge of filtering the *⟨keys⟩* we want to *store* in *⟨sequence⟩* where `{#1}` represents the optional value passed to the environment.

```

2056 \cs_new:Npn \__enumext_filter_save_key:n #1
2057 {
2058   \use:e
2059   {
2060     \keyval_parse:NNn
2061     \__enumext_filter_save_key_key:n
2062     \__enumext_filter_save_key_pair:nn {#1}
2063   }
2064 }

```

The function `\__enumext_filter_save_key_key:n` will be responsible for filtering the *⟨keys⟩* that are passed “without value” by excluding the `resume`, `resume*` and `no-store` keys.

```

2065 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2066 {
2067   \str_case:nnF {#1}
2068   {
2069     { resume } {} { resume* } {} { no-store } {}
2070   }
2071   { , { \exp_not:n {#1} } }
2072 }

```

The function `\__enumext_filter_save_key_pair:nn` will be responsible for filtering the *⟨keys⟩* that are passed “with value” by excluding the *series*, *resume*, *start*, *save-ans*, *save-ref*, *check-ans*, *show-ans*, *save-pos*, *wrap-ans*, *mark-ans*, *wrap-opt*, *save-sep*, *mark-ref*, *mini-env*, *mini-sep*, *mini-right* and *mini-right\** keys.

```

2073 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2074 {
2075   \str_case:nnF {#1}
2076   {
2077     { series } {} { resume } {} { start } {} { save-ans } {}
2078     { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2079     { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2080     { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2081     { mini-right } {} { mini-right* } {}
2082   }
2083   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2084 }

```

(End of definition for `\__enumext_filter_save_key:n`, `\__enumext_filter_save_key_key:n`, and `\__enumext_filter_save_key_pair:nn`.)

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

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

2085 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2086 {
2087   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2088   {
2089     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2090   }
2091   { #1 }
2092 }
2093 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

#### 11.23.5 Function for storing content in sequence

```

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

```

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

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

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

```

2094 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2095 {
2096   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2097 }
2098 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

#### 11.23.6 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.

```

2099 \cs_new_protected:Nn \__enumext_store_level_open:
2100 {
2101   \bool_if:NT \__enumext_check_answers_bool
2102   {
2103     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2104     {
2105       \__enumext_store_addto_seq:n
2106       {
2107         \item \begin{enumext}
2108       }
2109     }
2110     {
2111       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2112       {

```

```

2113         \item \begin{enumext} [
2114         ]
2115         \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2116         {
2117         ]
2118         }
2119         \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2120     }
2121 }
2122 }
2123 \cs_new_protected:Nn \__enumext_store_level_close:
2124 {
2125     \bool_if:NT \l__enumext_check_answers_bool
2126     {
2127         \__enumext_store_addto_seq:n { \end{enumext} }
2128     }
2129 }

```

(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.

```

2130 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2131 {
2132     \bool_if:NT \l__enumext_check_answers_bool
2133     {
2134         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2135         {
2136             \__enumext_store_addto_seq:n
2137             {
2138                 \item \mode_leave_vertical:
2139                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2140                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2141             }
2142         }
2143         {
2144             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2145             {
2146                 \item \mode_leave_vertical:
2147                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2148                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2149                 ]
2150             }
2151             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2152             {
2153             ]
2154             }
2155             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2156         }
2157     }
2158 }
2159 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2160 {
2161     \bool_if:NT \l__enumext_check_answers_bool
2162     {
2163         \__enumext_store_addto_seq:n { \end{enumext*} }
2164     }
2165 }

```

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

### 11.23.7 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`

```

2165 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2166 {
2167   \mode_leave_vertical:
2168   \skip_horizontal:n { -\dim_use:N #2 }
2169   \makebox[0pt][ r ]
2170   {
2171     \makebox[ \dim_use:N #1 ][ \__enumext_mark_position_str ]
2172     {
2173       \tl_use:N \__enumext_mark_answer_sym_tl
2174     }
2175   }
2176   \skip_horizontal:n { \dim_use:N #2 }
2177 }
2178 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

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

### 11.24 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`.

```

2179 \keys_define:nn { enumext / anskey }
2180 {
2181   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2182   item-sym* .value_required:n = true,
2183   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2184   item-pos* .value_required:n = true,
2185   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2186   store-brk .default:n = true,
2187   store-brk .value_forbidden:n = true,
2188 }

```

This command \anskey will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error. If the `check-ans` key is active, increment `\g__enumext_count_item_with_ans_int`, then call internal function `\__enumext_store_anskey_code:nnnn` will “*store content*” in the *(sequence)* and in the *(prop list)*.

```

2189 \NewDocumentCommand \anskey { d() s o +m }
2190 {
2191   \bool_if:NF \l__enumext_store_active_bool
2192   {
2193     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2194   }
2195   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2196   {
2197     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2198   }
2199   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2200   {
2201     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2202   }
2203   \group_begin:
2204     %\bool_if:NT \l__enumext_check_answers_bool
2205     % {
2206       \int_gincr:N \g__enumext_item_anskey_int
2207     % }
2208     \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2209   \group_end:
2210 }

```

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

\\_\_enumext\_store\_anskey\_code:nnnn

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

```

2211 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2212 {
2213   \__enumext_store_addto_prop:n {#4}
2214   \bool_if:NT \__enumext_store_ref_key_bool
2215   {
2216     \__enumext_store_internal_ref:
2217   }
2218   \__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`.

```

2219   \tl_clear:N \l__enumext_store_anskey_arg_tl
2220   \tl_if_novalue:nF {#3}
2221   {
2222     \keys_set:nn { enumext / anskey } {#3}
2223   }
2224   \bool_lazy_and:nnT
2225   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2226   { \bool_not_p:n { \l__enumext_starred_bool } }
2227   {
2228     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2229   }
2230   \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).

```

2231   \tl_if_novalue:nF {#1}
2232   {
2233     \int_set:Nn \l__enumext_store_columns_join_int {#1}
2234     \bool_if:NT \l__enumext_starred_bool
2235     {
2236       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2237       {
2238         ( \exp_not:V \l__enumext_store_columns_join_int )
2239       }
2240     }
2241   }

```

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`.

```

2242   \bool_if:nTF {#2}
2243   {
2244     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2245     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2246     {
2247       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2248       {
2249         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2250       }
2251     }
2252     \dim_compare:nT
2253     {
2254       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2255     }
2256     {
2257       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2258       {
2259         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2260       }
2261     }
2262     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2263   }
2264   {
2265     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2266   }

```

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

```

2267   \bool_lazy_and:nnT

```

```

2268     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2269     { \bool_if_p:N \l__enumext_hyperref_bool }
2270     {
2271         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2272         {
2273             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2274             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2275         }
2276     }
2277     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2278 }

```

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

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

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

```

2279 \cs_new_protected:Nn \__enumext_store_internal_ref:
2280 {
2281     \cs_set_protected:Npn \__enumext_tmp:n ##1
2282     {
2283         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2284         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2285         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2286         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2287     }
2288     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2289     \cs_set:Npn \__enumext_tmp:n ##1
2290     { . \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.

```

2291     \bool_lazy_all:nT
2292     {
2293         { \bool_if_p:N \g__enumext_starred_bool }
2294         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2295     }
2296     {
2297         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2298         { \tl_use:N \l__enumext_label_copy_vii_tl }
2299     }
2300     \bool_lazy_all:nT
2301     {
2302         { \bool_if_p:N \l__enumext_standar_bool }
2303         { \bool_if_p:N \g__enumext_starred_bool }
2304         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2305     }
2306     {
2307         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2308         {
2309             \tl_use:N \l__enumext_label_copy_vii_tl
2310             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2311         }
2312     }

```

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.

```

2313     \bool_lazy_all:nT
2314     {
2315         { \bool_if_p:N \l__enumext_standar_bool }
2316         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2317         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2318         { \bool_not_p:n { \l__enumext_starred_bool } }
2319     }
2320     {
2321         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2322         {
2323             \tl_use:N \l__enumext_label_copy_i_tl
2324             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n

```

```

2325     }
2326   }
2327   \cs_set:Npn \__enumext_tmp:n ##1
2328   { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2329   \bool_lazy_all:nT
2330   {
2331     { \bool_if:p:N \l__enumext_standar_bool }
2332     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2333     { \bool_not_p:n { \g__enumext_starred_bool } }
2334     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2335   }
2336   {
2337     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2338     {
2339       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2340       . \tl_use:N \l__enumext_label_copy_vii_tl
2341     }
2342   }

```

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

```

2343   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2344   {
2345     \l__enumext_store_name_tl \c_colon_str
2346     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2347   }

```

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.

```

2348   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2349   {
2350     \__enumext_newlabel:nn
2351     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2352     { \l__enumext_newlabel_arg_two_tl }
2353   }
2354   \l__enumext_store_write_aux_file_tl
2355 }

```

(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.

```

2356 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2357 {
2358   \par
2359   \bool_if:NT \l__enumext_starred_bool
2360   {
2361     \cs_set:Nn \__enumext_level: { vii }
2362   }
2363   \__enumext_print_keyans_box:cc
2364   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2365   { \l__enumext_labelsep_ \__enumext_level: _dim }
2366   \__enumext_anskey_wrapper:n { #1 }
2367 }

```

(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.

```

2368 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2369 {
2370   \bool_if:NT \l__enumext_show_answer_bool
2371   {
2372     \__enumext_store_anskey_show_wrap:n { #1 }
2373   }
2374   \bool_if:NT \l__enumext_show_position_bool
2375   {
2376     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2377     {

```



```

2378     \group_begin:
2379     \exp_not:N \normalfont
2380     \exp_not:N \footnotesize [ \int_eval:n
2381     {
2382         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2383     }
2384     ]
2385     \group_end:
2386 }
2387 \__enumext_store_anskey_show_wrap:n { #1 }
2388 }
2389 }

```

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

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

### 11.25.1 Storing content in prop list

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

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

```

2390 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2391 {
2392     \tl_clear:N \l__enumext_store_keyans_label_tl
2393     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2394     {
2395         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2396     }
2397     {
2398         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2399     }
2400     \tl_if_novalue:nF { #1 }
2401     {
2402         % Set save-sep
2403         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2404         {
2405             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2406         }
2407         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2408     }
2409     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2410 }

```

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

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

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

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

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

```

2411 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2412 {
2413     \bool_if:NT \l__enumext_store_ref_key_bool
2414     {
2415         \cs_set_protected:Npn \__enumext_tmp:n ##1
2416         {
2417             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2418             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2419             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2420             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2421         }
2422         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2423         \__enumext_keyans_store_ref_aux_i:

```

```

2424     }
2425 }

```

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

```

2426 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2427 {
2428   \bool_if:NT \g__enumext_starred_bool
2429   {
2430     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2431   }
2432   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2433   {
2434     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2435       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2436   }
2437   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2438   {
2439     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2440       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2441   }
2442   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2443   {
2444     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2445       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2446   }
2447   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2448   {
2449     \l__enumext_store_name_tl \c_colon_str
2450     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2451   }
2452   \l__enumext_keyans_store_ref_aux_ii:
2453 }

```

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

```

2454 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2455 {
2456   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2457   {
2458     \l__enumext_newlabel:nn
2459       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2460       { \l__enumext_newlabel_arg_two_tl }
2461   }
2462   \l__enumext_store_write_aux_file_tl
2463 }

```

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

### 11.25.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the current  $\langle \textit{label} \rangle$  `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the  $\langle \textit{contents} \rangle$  of the optional argument of both commands to the `\l__enumext_store_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

2464 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2465 {
2466   \tl_clear:N \l__enumext_store_keyans_label_tl
2467   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2468   {
2469     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2470   }
2471   {
2472     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2473   }
2474   \tl_if_no_value:nF { #1 }
2475   {
2476     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2477     {

```

```

2478         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2479         {
2480             \l__enumext_store_keyans_item_opt_sep_tl
2481         }
2482     }
2483     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2484 }
2485 \__enumext_keyans_addto_seq_link:
2486 }

```

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

```

2487 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2488 {
2489     \bool_lazy_and:nnT
2490     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2491     { \bool_if_p:N \l__enumext_hyperref_bool }
2492     {
2493         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2494         {
2495             \hfill \exp_not:N \hyperlink
2496             {
2497                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2498             }
2499             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2500         }
2501     }
2502     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2503     \bool_if:NT \l__enumext_check_answers_bool
2504     {
2505         \int_gincr:N \g__enumext_item_anskey_int
2506     }
2507 }

```

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

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

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

```

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

```

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

```

2508 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2509 {
2510     \tl_if_novalue:nF { #1 }
2511     {
2512         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2513     }
2514     \bool_if:NT \l__enumext_show_answer_bool
2515     {
2516         \__enumext_keyans_show_ans:
2517     }
2518     \bool_if:NT \l__enumext_show_position_bool
2519     {
2520         \__enumext_keyans_show_pos:
2521     }
2522 }
2523 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2524 {
2525     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2526     {
2527         \bool_lazy_or:nnT
2528         { \bool_if_p:N \l__enumext_show_answer_bool }
2529         { \bool_if_p:N \l__enumext_show_position_bool }
2530         {
2531             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2532         }

```

```

2533     }
2534   }
2535   \cs_new_protected:Nn \__enumext_keyans_show_ans:
2536   {
2537     \tl_put_left:Nn \l__enumext_label_v_tl
2538     {
2539       \__enumext_print_keyans_box:NN
2540       \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2541     }
2542   }
2543   \cs_new_protected:Nn \__enumext_keyans_show_pos:
2544   {
2545     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2546     {
2547       \tl_set:Ne \l__enumext_mark_answer_sym_tl
2548       {
2549         \group_begin:
2550         \exp_not:N \normalfont
2551         \exp_not:N \footnotesize [ \int_eval:n
2552           {
2553             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2554           }
2555         ]
2556         \group_end:
2557       }
2558     }
2559     {
2560       \tl_set:Ne \l__enumext_mark_answer_sym_tl
2561       {
2562         \group_begin:
2563         \exp_not:N \normalfont
2564         \exp_not:N \footnotesize [ \int_eval:n
2565           {
2566             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2567           }
2568         ]
2569         \group_end:
2570       }
2571     }
2572     \tl_put_left:Nn \l__enumext_label_v_tl
2573     {
2574       \__enumext_print_keyans_box:NN
2575       \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2576     }
2577   }

```

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

## 11.26 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  $\langle symbol \rangle$  and its  $\langle offset \rangle$ .

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

```

item-pos* 2578 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2579 {
2580   \keys_define:nn { enumext / #1 }
2581   {
2582     item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2583     item-sym* .value_required:n = true,
2584     item-sym* .initial:n = { $\star$ },
2585     item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2586     item-pos* .value_required:n = true,
2587   }
2588 }
2589 \clist_map_inline:nn
2590 {
2591   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2592 }
2593 { \__enumext_tmp:nn #1 }

```

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

## 11.27 Redefining \footnote command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

2594 \cs_new_protected:Nn \__enumext_footnotetext:nn
2595 {
2596   \footnotetext[#1]{#2}
2597 }
2598 \cs_new_protected:Nn \__enumext_renew_footnote:
2599 {
2600   \seq_gclear:N \g__enumext_footnote_arg_seq
2601   \seq_gclear:N \g__enumext_footnote_int_seq
2602   \RenewDocumentCommand \footnote { o +m }
2603   {
2604     \tl_if_novalue:nTF {##1}
2605     {
2606       \stepcounter{footnote}
2607       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2608     }
2609     {
2610       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2611     }
2612     \footnotemark [ \g__enumext_footnote_int ]
2613     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2614     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2615   }
2616 }
2617 \cs_new_protected:Nn \__enumext_print_footnote:
2618 {
2619   \seq_if_empty:NF \g__enumext_footnote_int_seq
2620   {
2621     \seq_map_pairwise_function:NNN
2622     \g__enumext_footnote_int_seq
2623     \g__enumext_footnote_arg_seq
2624     \__enumext_footnotetext:nn
2625   }
2626 }

```

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

## 11.28 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.

### 11.28.1 The \item command in enumext

```

\__enumext_default_item:n

```

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

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `\l__enumext_check_ans_key_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:` (§11.29).

```

2627 \cs_new_protected:Npn \__enumext_default_item:n #1
2628 {
2629   \tl_if_novalue:nTF {#1}
2630   {
2631     \bool_if:NT \l__enumext_check_answers_bool
2632     {
2633       \int_gincr:N \g__enumext_item_number_int
2634     }
2635     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2636     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2637   }
2638   {
2639     \bool_set_eq:cc
2640     { l__enumext_wrap_label_ \__enumext_level: _bool }

```

```

2641         { \l__enumext_wrap_label_opt_ \l__enumext_level: _bool }
2642         \l__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \l__enumext_level: _tl
2643     }
2644 }

```

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

\l\_\_enumext\_starred\_item:nn

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

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

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

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

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

In this function the optional argument of `\l__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 `\l__enumext_make_label:` (§11.29).

```

2645 \cs_new_protected:Npn \l__enumext_starred_item:nn #1 #2
2646 {
2647     \tl_if_novalue:nF {#1}
2648     {
2649         \tl_set:cn { \l__enumext_item_symbol_ \l__enumext_level: _tl } {#1}
2650     }
2651     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2652     \tl_if_novalue:nTF {#2}
2653     {
2654         \dim_set_eq:cc
2655         { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim }
2656         { \l__enumext_labelsep_ \l__enumext_level: _dim }
2657     }
2658     {
2659         \dim_set:cn { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } {#2}
2660     }
2661     \bool_if:NT \l__enumext_check_answers_bool
2662     {
2663         \int_gincr:N \g__enumext_item_number_int
2664     }
2665     \bool_set_true:c { \l__enumext_wrap_label_ \l__enumext_level: _bool }
2666     \l__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \l__enumext_level: _tl }
2667 }

```

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

\l\_\_enumext\_redefine\_item:

The function `\l__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 `\l__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.30.2).

```

2668 \cs_new_protected:Npn \l__enumext_redefine_item:
2669 {
2670     \RenewDocumentCommand \item { s o o }
2671     {
2672         \bool_if:nTF {##1}
2673         {
2674             \l__enumext_starred_item:nn {##2} {##3}
2675         }
2676         { \l__enumext_default_item:n {##2} }
2677     }
2678 }

```

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

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

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

`\__enumext_keyans_default_item:n`

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

```

2679 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2680 {
2681   \tl_if_novalue:nTF { #1 }
2682   {
2683     \bool_set_true:N \__enumext_wrap_label_v_bool
2684     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2685   }
2686   {
2687     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2688     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2689   }
2690 }

```

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

`\__enumext_keyans_starred_item:n`

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

```

2691 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2692 {
2693   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2694   \__enumext_keyans_show_left:n { #1 }
2695   \bool_set_true:N \__enumext_wrap_label_v_bool
2696   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item

```

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

```

2697   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_keyans_tmpa_tl
2698   \__enumext_keyans_addto_prop:n { #1 }
2699   \__enumext_keyans_store_ref:
2700   \__enumext_keyans_addto_seq:n { #1 }
2701   \int_gincr:N \g__enumext_check_starred_cmd_int
2702 }

```

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

`\item*`

`\__enumext_keyans_redefine_item:`

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

This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the *keyans* environment (§11.30.2).

```

2703 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2704 {
2705   \RenewDocumentCommand \item { s o }
2706   {
2707     \bool_if:nTF {##1}
2708     {
2709       \peek_remove_spaces:n
2710       {
2711         \__enumext_keyans_starred_item:n {##2}
2712       }
2713     }
2714     {
2715       \__enumext_keyans_default_item:n {##2}
2716     }
2717   }
2718 }

```

(End of definition for `\item*` and `\__enumext_keyans_redefine_item:`. This function is documented on page 12.)



### 11.29 Redefining \makeLabel command

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

#### 11.29.1 Redefining \makeLabel for enumext

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

```
2719 \cs_new_protected:Nn \__enumext_item_starred:
2720 {
2721   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2722   {
2723     \mode_leave_vertical:
2724     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2725     \makebox[0pt][r]{\g__enumext_item_symbol_tl}
2726     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2727   }
2728 }
```

(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 (§11.30.2).

```
2729 \cs_new_protected:Nn \__enumext_make_label:
2730 {
2731   \RenewDocumentCommand \makeLabel { m }
2732   {
2733     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2734     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2735     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2736     {
2737       \__enumext_item_starred:
2738       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2739     }
2740     { ##1 }
2741     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2742     \tl_gclear:N \g__enumext_item_symbol_tl
2743   }
2744 }
```

(End of definition for `\__enumext_make_label:`)

#### 11.29.2 Redefining \makeLabel for keyans

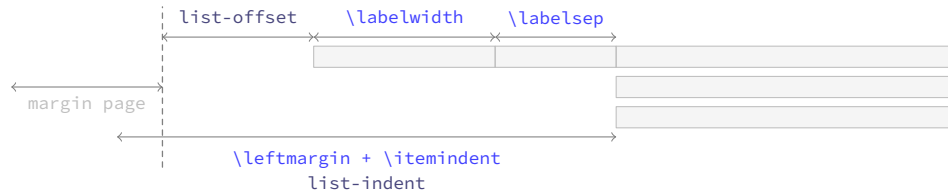
`\__enumext_keyans_make_label:` The function `\__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment. This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.30.2).

```
2745 \cs_new_protected:Nn \__enumext_keyans_make_label:
2746 {
2747   \RenewDocumentCommand \makeLabel { m }
2748   {
2749     \tl_use:N \l__enumext_label_fill_left_v_tl
2750     \tl_use:N \l__enumext_label_font_style_v_tl
2751     \bool_if:NTF \l__enumext_wrap_label_v_bool
2752     {
2753       \__enumext_wrapper_label_v:n { ##1 }
2754     }
2755     { ##1 }
2756     \tl_use:N \l__enumext_label_fill_right_v_tl
2757   }
2758 }
```

(End of definition for `\__enumext_keyans_make_label:`)

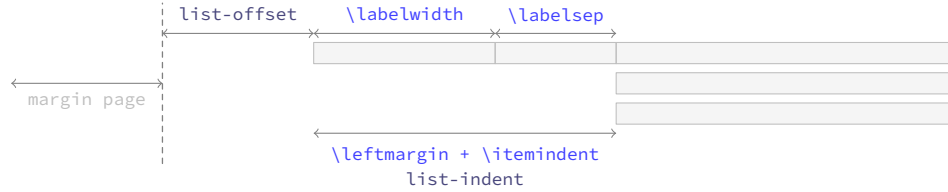
### 11.30 Second argument of the lists

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

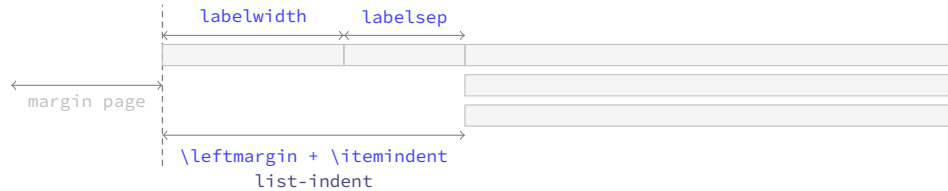
Figure 9: Representation of standard horizontal lengths in `list` environment.

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

Consider the figure 9 where the default margins (on the left) of a list are represented. The idea is to have control over these margins so that our list does not overlap the left margin of the page. The key relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the *label box* is at `\leftmargin + \itemindent` minus `\labelwidth + \labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

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

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

Figure 11: Default horizontal lengths in `enumext`.

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc
```

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

```
#1: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__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 (§11.30.2).

```
2759 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2760 {
2761   \dim_compare:nNt { #1 } < { \c_zero_dim }
2762   {
2763     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2764     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2765   }
2766   \dim_compare:nNt { #2 } < { \c_zero_dim }
2767   {
2768     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2769     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2770   }
2771 }
```

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

```
2771   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2772   \dim_compare:nNtF { #4 } < { \c_zero_dim }
2773   {
2774     \dim_set:Nn #6 { #1 + #2 - #4 }
2775     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2776   }
```

```

2777     {
2778       \dim_compare:nNnT { #4 } = { #1 + #2 }
2779       { \dim_set:Nn #6 { \c_zero_dim } }
2780       \dim_compare:nNnT { #4 } < { #1 + #2 }
2781       { \dim_set:Nn #6 { #1 + #2 - #4 } }
2782       \dim_compare:nNnT { #4 } > { #1 + #2 }
2783       {
2784         \dim_set:Nn #6 { -#1 - #2 + #4 }
2785         \dim_set:Nn #6 { #6*-1 }
2786       }
2787       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2788     }
2789   }
2790   \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(End of definition for \\_\_enumext\_calc\_hspace:NNNNNNN.)

### 11.30.2 Setting second argument of the lists

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

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
2791   \cs_set_protected:Npn \__enumext_tmp:n #1
2792   {
2793     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2794     {
2795       \__enumext_calc_hspace:ccccc
2796       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2797       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2798       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2799       { \__enumext_leftmargin_tmp_#1_bool }
2800       \clist_map_inline:nn
2801       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2802       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2803       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2804       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2805       \usecounter { enumX#1 }
2806       \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2807       \str_if_eq:nnTF {#1} { v }
2808       {
2809         \__enumext_keyans_redefine_item:
2810         \__enumext_keyans_make_label:
2811         \__enumext_keyans_ref:
2812         \__enumext_keyans_fake_item:
2813         \bool_if:cT { \__enumext_show_length_#1_bool }
2814         {
2815           \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2816         }
2817       }
2818       {
2819         \__enumext_redefine_item:
2820         \__enumext_make_label:
2821         \__enumext_standar_ref:
2822         \__enumext_fake_item:
2823         \bool_if:cT { \__enumext_show_length_#1_bool }
2824         {
2825           \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2826         }
2827       }
2828     }
2829   }
2830   \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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.

```

2831   \cs_set_protected:Npn \__enumext_tmp:n #1
2832   {
2833     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2834     {

```

```

2835 \__enumext_calc_hspace:ccccc
2836 { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2837 { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2838 { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2839 { \__enumext_leftmargin_tmp_#1_bool }
2840 \clist_map_inline:nn
2841 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2842 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2843 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2844 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2845 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2846 \skip_zero:N \partopsep
2847 \usecounter { enumX#1 }
2848 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2849 \__enumext_starred_ref:
2850 \str_if_eq:nnTF {#1} { vii }
2851 {
2852   \__enumext_fake_item_vii:
2853   \bool_if:cT { \__enumext_show_length_vii_bool }
2854   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2855 }
2856 {
2857   \__enumext_fake_item_viii:
2858   \bool_if:cT { \__enumext_show_length_#1_bool }
2859   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2860 }
2861 }
2862 }
2863 \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:.)

### 11.31 The environment enumext

**enumext** We create the `enumext` environment based on `list` environment by levels.

```

2864 \NewDocumentEnvironment{enumext}{ 0{ } }
2865 {
2866   \__enumext_safe_exec:
2867   \__enumext_parse_keys:n {#1}
2868   \__enumext_before_list:
2869   \__enumext_start_store_level:
2870   \__enumext_start_list:nn
2871   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2872   {
2873     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2874     \__enumext_before_keys_exec:
2875   }
2876   \__enumext_after_args_exec:
2877 }
2878 {
2879   \__enumext_stop_list:
2880   \__enumext_stop_store_level:
2881   \__enumext_after_list:
2882 }

```

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

`\__enumext_safe_exec:` The `\__enumext_safe_exec:` function first execute the function `\__enumext_is_not_nested:` which will set the variable `\g__enumext_standar_bool` to “true” if the environment is not nested in `enumext*`, we increment the variable `\l__enumext_level_int` for the nesting levels and set the `\l__enumext_standar_bool` variable to “true”. Finally we set the variable `\l__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level” of it using the function `\__enumext_is_on_first_level:`.

```

2883 \cs_new_protected:Nn \__enumext_safe_exec:
2884 {
2885   \__enumext_is_not_nested:
2886   \int_incr:N \l__enumext_level_int
2887   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2888   { \msg_fatal:nn { enumext } { list-too-deep } }
2889   \bool_set_true:N \l__enumext_standar_bool
2890   \__enumext_is_on_first_level:
2891 }

```

(End of definition for `\__enumext_safe_exec:`)

`\__enumext_parse_keys:n`

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

```

2892 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2893 {
2894   \tl_if_novalue:nF {#1}
2895   {
2896     \str_clear:N \l__enumext_series_str
2897     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2898     {
2899       \keys_set:nn { enumext / level-1 } {#1}
2900       \__enumext_parse_series:n {#1}
2901     }
2902     {
2903       \exp_args:Ne \keys_set:nn
2904         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2905     }
2906     \__enumext_store_active_keys:n {#1}
2907   }
2908 }

```

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

`\__enumext_start_store_level:`

The `\__enumext_start_store_level:` and `\__enumext_stop_store_level:` functions activate the level saving mechanism for storage in *⟨sequence⟩* of the `\anskey` command.

If `enumext` are nested in `enumext*` add `\__enumext_store_level_open:` to preserve the stored structure.

```

2909 \cs_new_protected:Nn \__enumext_start_store_level:
2910 {
2911   \bool_lazy_all:nT
2912   {
2913     { \bool_if_p:N \l__enumext_store_active_bool }
2914     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2915     { \bool_not_p:n { \g__enumext_starred_bool } }
2916   }
2917   {
2918     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2919     {
2920       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2921       \__enumext_store_level_open:
2922     }
2923   }
2924   \bool_lazy_all:nT
2925   {
2926     { \bool_if_p:N \l__enumext_store_active_bool }
2927     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2928     { \bool_if_p:N \g__enumext_starred_bool }
2929   }
2930   {
2931     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2932     {
2933       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2934       \__enumext_store_level_open:
2935     }
2936   }
2937 }
2938 \cs_new_protected:Nn \__enumext_stop_store_level:
2939 {
2940   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2941   {
2942     \__enumext_store_level_close:
2943   }
2944 }

```

(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.

```
2945 \cs_new_protected:Nn \__enumext_before_list:
2946 {
2947   \__enumext_vspace_above:
2948   \__enumext_before_args_exec:
```

The function `\__enumext_check_ans_active:` will handle the check answer mechanism, which will be activated with the `check-ans` key.

```
2949   \__enumext_check_ans_active:
```

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.

```
2950   \dim_compare:nNtT
2951   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2952   {
2953     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2954     {
2955       \linewidth
2956       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2957       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2958     }
2959   }
```

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

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

```
2959   \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
2960   \int_gincr:N \g__enumext_minipage_stat_int
2961   \__enumext_mini_addvspace:
2962   \nointerlineskip\noindent
2963   \begin{\__enumext_mini_env*}
2964   { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
2965   }
2966   \__enumext_multicols_start:
2967   }
```

(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.

```
2968 \cs_new_protected:Nn \__enumext_multicols_start:
2969 {
2970   \int_compare:nNtT
2971   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2972   {
2973     \dim_compare:nNtT
2974     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2975     {
2976       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
2977       {
2978         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
2979         + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2980         ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
2981         - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
2982       }
2983     }
2984     \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
2985     \skip_zero:N \multicolsep
2986     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2987     {
2988       \dim_zero:N \columnseprule
2989     }
2990   }
```

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.

```

2990     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2991     {
2992         \__enumext_multi_addvspace:
2993     }
2994     \raggedcolumns
2995     \begin{multicols}{\int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2996 }
2997 }

```

(End of definition for `\__enumext_multicols_start:`)

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

```

2998 \cs_new_protected:Nn \__enumext_multicols_stop:
2999 {
3000     \int_compare:nNt
3001     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3002     {
3003         \end{multicols}
3004         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3005         {
3006             \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3007         }
3008     }
3009 }

```

(End of definition for `\__enumext_multicols_stop:`)

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

```

3010 \cs_new_protected:Nn \__enumext_after_list:
3011 {
3012     \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
3013     {
3014         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3015         {
3016             \msg_warning:nn { enumext } { missing-miniright }
3017             \miniright
3018         }
3019         \int_gzero:N \g__enumext_minipage_stat_int
3020         \end{\__enumext_mini_env*}
3021         \par\addvspace { \__enumext_minipage_after_skip }
3022     }
3023     { \__enumext_multicols_stop: }

```

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

```

3024     \__enumext_check_ans_key_hook:

```

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

```

3025     \__enumext_after_stop_list:
3026     \__enumext_vspace_below:
3027     \bool_set_false:N \__enumext_standar_bool
3028     \__enumext_resume_save_counter:
3029 }

```

(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`.

```

3030 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```



## 11.32 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.

```

3031 \NewDocumentEnvironment{keyans}{0}{ }
3032 {
3033   \__enumext_keyans_safe_exec:
3034   \__enumext_keyans_parse_keys:n {#1}
3035   \__enumext_before_list_v:
3036   \__enumext_start_list:nn
3037   { \tl_use:N \__enumext_label_v_tl }
3038   {
3039     \__enumext_list_arg_two_v:
3040     \__enumext_before_keys_exec_v:
3041   }
3042   \__enumext_after_args_exec_v:
3043 }
3044 {
3045   \__enumext_check_starred_cmd:n { item }
3046   \__enumext_stop_list:
3047   \__enumext_after_list_v:
3048 }
```

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

`\__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.

```

3049 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3050 {
3051   \bool_if:NF \__enumext_store_active_bool
3052   {
3053     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3054   }
3055   \int_incr:N \__enumext_keyans_level_int
3056   \bool_set_true:N \__enumext_keyans_env_bool
3057   \__enumext_keyans_save_start_line:
3058   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3059   \bool_set_false:N \__enumext_store_active_bool
3060   \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
3061   {
3062     \msg_error:nn { enumext } { keyans-nested }
3063   }
3064   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3065   {
3066     \msg_error:nn { enumext } { keyans-wrong-level }
3067   }
3068 }
```

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

`\__enumext_keyans_parse_keys:n`

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

```

3069 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3070 {
3071   \keys_set:nn { enumext / keyans } {#1}
3072 }
```

(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.

```

3073 \cs_new_protected:Nn \__enumext_before_list_v:
3074 {
3075   \__enumext_vspace_above_v:
3076   \__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.

```

3077 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3078 {
3079   \dim_set:Nn \l__enumext_minipage_left_v_dim
3080   {
3081     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3082   }

```

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*.

```

3083 \bool_set_true:N \l__enumext_minipage_active_v_bool
3084 \int_gincr:N \g__enumext_minipage_stat_int
3085 \__enumext_keyans_mini_addvspace:
3086 \nointerlineskip\noindent
3087 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3088 }

```

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

```

3089 \__enumext_keyans_multicols_start:
3090 }

```

(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.

```

3091 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3092 {
3093   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3094   {

```

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

```

3095 \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3096 {
3097   \dim_set:Nn \l__enumext_columns_sep_v_dim
3098   {
3099     (
3100       \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3101     ) / \l__enumext_columns_v_int
3102     - \l__enumext_listoffset_v_dim
3103   }
3104 }
3105 \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).

```

3106 \skip_zero:N \multicolsep
3107 \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.

```

3108 \bool_if:NF \l__enumext_minipage_active_v_bool
3109 {
3110   \__enumext_keyans_multi_addvspace:
3111 }
3112 \raggedcolumns
3113 \begin{multicols}{ \l__enumext_columns_v_int }
3114 }
3115 }

```

(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.

```

3116 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3117 {
3118   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3119   {
3120     \end{multicols}
3121     \bool_if:NF \l__enumext_minipage_active_v_bool
3122     {
3123       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3124     }
3125   }
3126 }

```

(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.

```

3127 \cs_new_protected:Nn \__enumext_after_list_v:
3128 {
3129   \bool_if:NTF \l__enumext_minipage_active_v_bool
3130   {
3131     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3132     {
3133       \msg_warning:nn { enumext } { missing-miniright }
3134       \miniright
3135     }
3136     \int_gzero:N \g__enumext_minipage_stat_int
3137     \end{__enumext_mini_env*}
3138     \par\addvspace{ \l__enumext_minipage_after_skip }
3139   }
3140   { \__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.

```

3141   \bool_set_false:N \l__enumext_keyans_env_bool
3142   \__enumext_after_stop_list_v:
3143   \__enumext_vspace_below_v:
3144 }

```

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

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

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

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

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

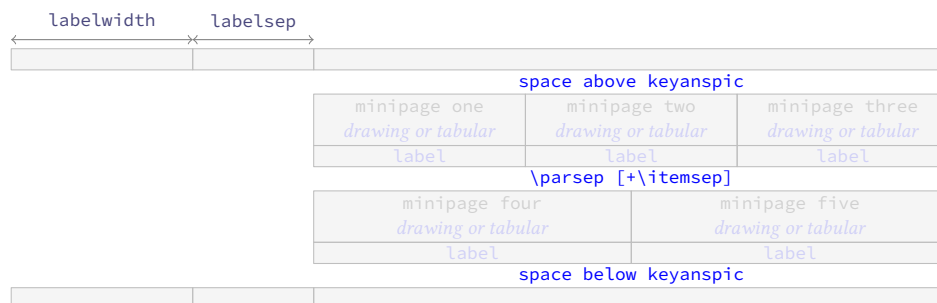


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

### 11.33.1 The command `\anspic`

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

```
3145 \NewDocumentCommand \anspic { s o +m }
3146 {
```

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

```
3147   \bool_if:NF \__enumext_store_active_bool
3148   {
3149     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3150   }
3151   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3152   {
3153     \msg_error:nn { enumext } { keyanspic-wrong-level }
3154   }
3155   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3156   {
3157     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3158   }
```

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

```
3159   \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3160   {
3161     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3162   }
3163 }
```

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

`\__enumext_keyans_anspic_code:nnn`

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

```
3164 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3165 {
3166   \stepcounter { enumXvi }
3167   #3 \\\
3168   \bool_if:nT { #1 }
3169   {
3170     \__enumext_keyans_addto_prop:n { #2 }
3171     \__enumext_keyans_store_ref:
3172     \__enumext_keyans_addto_seq:n { #2 }
3173     \int_gincr:N \__enumext_check_starred_cmd_int
3174     \bool_lazy_or:nnT
3175     { \bool_if_p:N \__enumext_show_answer_bool }
3176     { \bool_if_p:N \__enumext_show_position_bool }
3177     {
3178       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3179       \__enumext_keyans_show_left:n { #2 }
3180       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3181     }
3182   }
3183   \tl_use:N \__enumext_label_font_style_v_tl
3184   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3185 }
```

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

### 11.33.2 The environment `keyanspic`

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

```
3186 \NewDocumentEnvironment{keyanspic}{ o }
3187 {
3188   \__enumext_keyans_pic_safe_exec:
3189   \__enumext_start_list:nn
3190   { }
3191   {
```

```

3192     \__enumext_keyans_pic_arg_two:
3193 }

```

We apply the “adjusted” vertical spacing above the environment

```

3194 \vspace { \__enumext_keyans_pic_above_skip }
3195 }

```

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

```

3196 {
3197   \tl_if_novalue:nTF { #1 }
3198   {
3199     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
3200   }
3201   { \__enumext_keyans_pic_do:n { #1 } }
3202   \__enumext_stop_list:
3203   \__enumext_check_starred_cmd:n { anspic }
3204   \setcounter { enumXvi } { 0 }
3205   \vspace { \__enumext_topsep_v_skip }
3206   %\bool_set_false:N \__enumext_store_active_bool
3207 }

```

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

`\__enumext_keyans_pic_safe_exec:` The function `\__enumext_keyans_pic_safe_exec:` check nested and level position inside the `enumext` environment.

```

3208 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3209 {
3210   \int_incr:N \__enumext_keyans_pic_level_int
3211   \int_compare:nNt { \__enumext_keyans_pic_level_int } > { 1 }
3212   {
3213     \msg_error:nn { enumext } { keyanspic-nested }
3214   }
3215   \__enumext_keyans_save_start_line:
3216 }

```

(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`.

```

3217 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3218 {
3219   \dim_compare:nNt { #1 } < { 0pt }
3220   { \skip_set:Nn #1 { -#1 } }
3221 }

```

(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.

```

3222 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3223 {

```

The first thing to do is to set the boolean variable `\__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.

```

3224   \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
3225   \__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.

```

3226   \skip_add:Nn \parsep { \itemsep }
3227   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3228   \dim_zero:N \labelwidth
3229   \dim_zero:N \listparindent
3230   \dim_zero:N \labelsep
3231   \skip_zero:N \partopsep
3232   \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.

```

3233 \__enumext_keyans_pic_skip_abs:N \parsep
3234 \skip_set:Nn \l__enumext_keyans_pic_above_skip
3235 {
3236   \box_dp:N \strutbox
3237   + \l__enumext_topsep_v_skip
3238   - \parsep
3239 }
3240 \__enumext_item_std:w \scan_stop:
3241 }
```

(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`.

```

3242 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3243 {
3244   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3245 }
3246 \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.

```

3247 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3248 {
3249   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3250   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3251   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3252   \int_step_inline:nnn
3253     { \l__enumext_keyans_pic_above_int + 1 }
3254     { \l__enumext_keyans_pic_below_int }
3255     {
3256       \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3257       \centering
3258       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3259       \__enumext_endminipage:
3260     }
3261   \par
3262 }
```

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

## 11.34 The environment `enumext*`

Generating horizontal list environments is NOT as simple as standard  $\TeX$  list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content passed to `\item`, we will also add the optional argument ( $\langle number \rangle$ ) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an optional first argument ( $\langle number \rangle$ ).

### 11.34.1 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.

```

3263 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3264 {
3265   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3266   {
3267     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3268     {
3269       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )

```

```

3270         / \l__enumext_columns_vii_int
3271     }
3272 }
3273 \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3274 \dim_set:Nn \l__enumext_item_width_vii_dim
3275 {
3276     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3277     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3278     - \l__enumext_labelsep_vii_dim
3279 }
3280 \dim_zero_new:N \itemwidth
3281 }

```

(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`.

```

3282 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3283 {
3284     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3285     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3286     {
3287         \msg_warning:nnee { enumext } { item-joined }
3288         { \int_use:N \l__enumext_joined_item_vii_int }
3289         { \int_use:N \l__enumext_columns_vii_int }
3290         \int_set:Nn \l__enumext_joined_item_vii_int
3291         {
3292             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3293         }
3294     }
3295     \int_compare:nNnT
3296     { \l__enumext_joined_item_vii_int }
3297     >
3298     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3299     {
3300         \msg_warning:nnee { enumext } { item-joined-columns }
3301         { \int_use:N \l__enumext_joined_item_vii_int }
3302         {
3303             \int_eval:n
3304             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3305         }
3306         \int_set:Nn \l__enumext_joined_item_vii_int
3307         {
3308             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3309         }
3310     }

```

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

```

3311     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3312     {
3313         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3314         \int_decr:N \l__enumext_joined_item_aux_vii_int
3315         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3316         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3317         \dim_set:Nn \l__enumext_joined_width_vii_dim
3318         {
3319             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3320             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3321               + \l__enumext_columns_sep_vii_dim
3322             ) * \l__enumext_joined_item_aux_vii_int
3323         }
3324         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3325     }
3326     {
3327         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3328         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3329     }
3330 }

```

(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`.

```

3331 \cs_new_protected:Nn \__enumext_start_mini_vii:
3332 {
3333   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3334   {
3335     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3336     {
3337       \linewidth
3338       - \l__enumext_minipage_right_vii_dim
3339       - \l__enumext_minipage_hsep_vii_dim
3340     }
3341     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3342     \dim_gset_eq:NN
3343       \g__enumext_minipage_right_vii_dim
3344       \l__enumext_minipage_right_vii_dim
3345     \__enumext_mini_addvspace_vii:
3346     \nointerlineskip\noindent
3347     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3348   }
3349 }

```

(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*”.

```

3350 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3351 {
3352   \bool_if:NT \l__enumext_minipage_active_vii_bool
3353   {
3354     \end{__enumext_mini_env*}
3355     \hfill
3356     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3357   }
3358 }

```

Finally we execute code passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `\__enumext_mini_env*` environment on the “*right side*”.

```

3359 \__enumext_after_env:nn {enumext*}
3360 {
3361   \bool_if:NT \g__enumext_minipage_active_vii_bool
3362   {
3363     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3364     \par\addvspace { \g__enumext_minipage_right_skip }
3365     \bool_if:NF \g__enumext_minipage_center_vii_bool
3366     {
3367       \centering
3368     }
3369     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3370     \end{__enumext_mini_env*}
3371     \par\addvspace{ \g__enumext_minipage_after_skip }
3372   }
3373   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3374   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3375   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3376   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3377   \bool_gset_false:N \g__enumext_starred_bool
3378 }

```

(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.

```

3379 \NewDocumentEnvironment{enumext*}{ o }
3380 {

```

```

3381   \__enumext_safe_exec_vii:
3382   \__enumext_parse_keys_vii:n {#1}
3383   \__enumext_before_list_vii:
3384   \__enumext_start_store_level_vii:
3385   \__enumext_start_list:nn { }
3386   {
3387     \__enumext_list_arg_two_vii:
3388     \__enumext_before_keys_exec_vii:
3389   }
3390   \__enumext_starred_columns_set_vii:
3391   \item[] \scan_stop:
3392   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3393   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3394 }
3395 {
3396   \__enumext_stop_item_tmp_vii:
3397   \__enumext_remove_extra_parsep_vii:
3398   \__enumext_stop_list:
3399   \__enumext_stop_store_level_vii:
3400   \__enumext_after_list_vii:
3401 }

```

(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`.

```

3402 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3403 {
3404   \__enumext_is_not_nested:
3405   \int_incr:N \l__enumext_level_h_int
3406   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3407   {
3408     \msg_error:nn { enumext } { nested }
3409   }
3410   \bool_set_true:N \l__enumext_starred_bool
3411   \__enumext_is_on_first_level:
3412 }

```

(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 functions `\__enumext_parse_series:n` and `\__enumext_store_active_keys_vii:n` and reprocess the `keys` to pass them to the storage `⟨sequence⟩`.

```

3413 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3414 {
3415   \tl_if_novalue:nF {#1}
3416   {
3417     \str_clear:N \l__enumext_series_str
3418     \keys_set:nn { enumext / enumext* } {#1}
3419     \__enumext_parse_series:n {#1}
3420     \__enumext_store_active_keys_vii:n {#1}
3421   }
3422 }

```

(End of definition for `\__enumext_parse_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`.

```

3423 \cs_new_protected:Nn \__enumext_before_list_vii:
3424 {
3425   \__enumext_vspace_above_vii:
3426   \__enumext_check_ans_active:
3427   \__enumext_before_args_exec_vii:
3428   \__enumext_start_mini_vii:
3429 }

```

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

`\__enumext_after_list_vii:` The function `\__enumext_after_list:` first call the function `\__enumext_stop_mini_vii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present. Finally set false the vars `\g__enumext_starred_bool` and `\l__enumext_starred_bool`, save the *current value* of the counter in `\g__enumext_resume_vii_int` for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

3430 \cs_new_protected:Nn \__enumext_after_list_vii:
3431 {
3432   \__enumext_stop_mini_vii:
3433   \__enumext_after_stop_list_vii:
3434   \__enumext_check_ans_key_hook:
3435   \__enumext_vspace_below_vii:
3436   \bool_set_false:N \l__enumext_starred_bool
3437   \__enumext_resume_save_counter:
3438 }

```

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

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

`\__enumext_stop_store_level_vii:`

```

3439 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3440 {
3441   \bool_if:NT \l__enumext_store_active_bool
3442   {
3443     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3444     {
3445       \__enumext_store_level_open_vii:
3446     }
3447   }
3448 }
3449 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3450 {
3451   \bool_if:NT \l__enumext_store_active_bool
3452   {
3453     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3454     {
3455       \__enumext_store_level_close_vii:
3456     }
3457   }
3458 }

```

(End of definition for `\__enumext_start_store_level_vii:` and `\__enumext_stop_store_level_vii:`.)

### 11.34.2 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`.

```

3459 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3460 {
3461   \__enumext_stop_item_tmp_vii:
3462   \int_incr:N \l__enumext_item_column_pos_vii_int
3463   \int_gincr:N \g__enumext_item_count_all_vii_int
3464   \__enumext_item_peek_args_vii:
3465 }

```

(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).

```

3466 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3467 {
3468   \peek_meaning:NTF (
3469     { \__enumext_joined_item_vii:w }
3470     { \__enumext_joined_item_vii:w (1) }
3471   )

```

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

`\__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_standar_item_vii:w`.

```

3472 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3473 {
3474     \__enumext_starred_joined_item_vii:n {#1}
3475     \peek_meaning_remove:NTF *
3476     { \__enumext_starred_item_vii:w }
3477     { \__enumext_standar_item_vii:w }
3478 }

```

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

`\__enumext_standar_item_vii:w` The function `\__enumext_standar_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 ]`.

```

3479 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3480 {
3481     \bool_set_false:N \l__enumext_item_starred_vii_bool
3482     \peek_meaning:NTF [
3483     {
3484         \bool_set_eq:NN
3485         \l__enumext_wrap_label_vii_bool
3486         \l__enumext_wrap_label_opt_vii_bool
3487         \__enumext_start_item_vii:w
3488     }
3489     {
3490         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3491         \legacy_if_set_true:n { @noitemarg }
3492         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3493     }
3494 }

```

(End of definition for `\__enumext_standar_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⟩]`.

```

3495 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3496 {
3497     \bool_set_true:N \l__enumext_item_starred_vii_bool
3498     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3499     \peek_meaning:NTF [
3500     { \__enumext_starred_item_vii_aux_i:w }
3501     { \__enumext_starred_item_vii_aux_ii:w }
3502 }
3503 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3504 {
3505     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3506     \__enumext_starred_item_vii_aux_ii:w
3507 }
3508 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3509 {
3510     \peek_meaning:NTF [
3511     { \__enumext_starred_item_vii_aux_iii:w }
3512     {
3513         \dim_set_eq:NN
3514         \l__enumext_item_symbol_sep_vii_dim
3515         \l__enumext_labelsep_vii_dim
3516         \legacy_if_set_true:n { @noitemarg }
3517         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3518     }
3519 }
3520 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]

```

```

3521 {
3522   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3523   \legacy_if_set_true:n { @noitemarg }
3524   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3525 }

```

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

### 11.34.3 Real definition of `\item` in `enumext*`

`\__enumext_start_item_vii:w` The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment.

The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to the value of `\__enumext_stop_item_vii:` which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

3526 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3527 {
3528   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3529   \legacy_if:nT { @noitemarg }
3530   {
3531     \legacy_if_set_false:n { @noitemarg }
3532     \legacy_if:nT { @nmbrlist }
3533     {
3534       \bool_if:NT \l__enumext_hyperref_bool
3535       {
3536         \legacy_if_set_true:n { @hyper@item }
3537       }
3538       \refstepcounter{enumXvii}
3539       \bool_if:NT \l__enumext_check_answers_bool
3540       {
3541         \int_gincr:N \g__enumext_item_number_int
3542       }
3543     }
3544   }

```

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

```

3545   \group_begin:
3546   \lrbox{ \l__enumext_item_text_vii_box }
3547   \bool_if:NF \l__enumext_footnotes_key_bool
3548   {
3549     \__enumext_renew_footnote:
3550   }
3551   \bool_if:NT \l__enumext_item_starred_vii_bool
3552   {
3553     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3554     {
3555       \tl_gset_eq:NN
3556       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3557     }
3558     \mode_leave_vertical:
3559     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3560     \makebox[ 0pt ]{ r }{ \g__enumext_item_symbol_aux_vii_tl }
3561     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3562     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3563   }
3564   \group_begin:
3565   \tl_use:N \l__enumext_label_font_style_vii_tl
3566   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3567   {
3568     \makebox[ \l__enumext_labelwidth_vii_dim ]{ \l__enumext_align_label_vii_str }
3569     { \__enumext_wrapper_label_vii:n {#1} }
3570   }
3571   {

```

```

3572         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3573     }
3574     \group_end:
3575     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3576     \tl_use:N \l__enumext_after_list_args_vii_tl
3577     \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3578     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3579     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3580     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3581 }

```

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

`\l__enumext_stop_item_vii:` The function `\l__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.

```

3582 \cs_new_protected_nopar:Nn \l__enumext_stop_item_vii:
3583 {
3584     \l__enumext_endminipage:
3585     \endlrbox
3586     \group_end:
3587     \box_set_wd:Nn \l__enumext_item_text_vii_box
3588     {
3589         \l__enumext_joined_width_vii_dim
3590         + \l__enumext_labelwidth_vii_dim
3591         + \l__enumext_labelsep_vii_dim
3592     }
3593     \int_set:Nn \hbadness { 10000 }
3594     \box_use:N \l__enumext_item_text_vii_box
3595     \bool_if:NF \l__enumext_footnotes_key_bool
3596     {
3597         \l__enumext_print_footnote:
3598     }
3599     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3600     {
3601         \par\noindent
3602         \int_zero:N \l__enumext_item_column_pos_vii_int
3603     }
3604     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3605 }

```

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

`\l__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.

```

3606 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
3607 {
3608     \int_compare:nNnT
3609     {
3610         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3611     }
3612     =
3613     { \c_zero_int }
3614     {
3615         \par
3616         \vspace{ -\l__enumext_itemsep_vii_skip }
3617         \int_gzero:N \g__enumext_item_count_all_vii_int
3618     }
3619 }

```

(End of definition for `\l__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 `\l__enumext_after_env:nn`.

```

3620 \l__enumext_after_env:nn {enumext*} { \l__enumext_execute_after_env: }

```

## 11.35 The environment keyans\*

### 11.35.1 Functions for item box width

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

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

```

3621 \cs_new_protected:Nn \_enumext_starred_columns_set_viii:
3622 {
3623   \dim_compare:nNnT { \\_enumext_columns_sep_viii_dim } = { \c_zero_dim }
3624   {
3625     \dim_set:Nn \\_enumext_columns_sep_viii_dim
3626     {
3627       ( \\_enumext_labelwidth_viii_dim + \\_enumext_labelsep_viii_dim )
3628       / \\_enumext_columns_viii_int
3629     }
3630   }
3631   \int_set:Nn \\_enumext_tmpa_viii_int { \\_enumext_columns_viii_int - \c_one_int }
3632   \dim_set:Nn \\_enumext_item_width_viii_dim
3633   {
3634     ( \linewidth - \\_enumext_columns_sep_viii_dim * \\_enumext_tmpa_viii_int )
3635     / \\_enumext_columns_viii_int - \\_enumext_labelwidth_viii_dim
3636     - \\_enumext_labelsep_viii_dim
3637   }
3638   \dim_zero_new:N \itemwidth
3639 }

```

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

\\_enumext\_starred\_joined\_item\_viii:n

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

```

3640 \cs_new_protected:Npn \_enumext_starred_joined_item_viii:n #1
3641 {
3642   \int_set:Nn \\_enumext_joined_item_viii_int {#1}
3643   \int_compare:nNnT { \\_enumext_joined_item_viii_int } > { \\_enumext_columns_viii_int }
3644   {
3645     \msg_warning:nnee { enumext } { item-joined }
3646     { \int_use:N \\_enumext_joined_item_viii_int }
3647     { \int_use:N \\_enumext_columns_viii_int }
3648     \int_set:Nn \\_enumext_joined_item_viii_int
3649     {
3650       \\_enumext_columns_viii_int - \\_enumext_item_column_pos_viii_int + \c_one_int
3651     }
3652   }
3653   \int_compare:nNnT
3654   { \\_enumext_joined_item_viii_int }
3655   >
3656   { \\_enumext_columns_viii_int - \\_enumext_item_column_pos_viii_int + \c_one_int }
3657   {
3658     \msg_warning:nnee { enumext } { item-joined-columns }
3659     { \int_use:N \\_enumext_joined_item_viii_int }
3660     {
3661       \int_eval:n
3662       { \\_enumext_columns_viii_int - \\_enumext_item_column_pos_viii_int + \c_one_int }
3663     }
3664     \int_set:Nn \\_enumext_joined_item_viii_int
3665     {
3666       \\_enumext_columns_viii_int - \\_enumext_item_column_pos_viii_int + \c_one_int
3667     }
3668   }
3669   \int_compare:nNnTF { \\_enumext_joined_item_viii_int } > { \c_one_int }
3670   {
3671     \int_set_eq:NN \\_enumext_joined_item_aux_viii_int \\_enumext_joined_item_viii_int
3672     \int_decr:N \\_enumext_joined_item_aux_viii_int
3673     \int_add:Nn \\_enumext_item_column_pos_viii_int { \\_enumext_joined_item_aux_viii_int }
3674     \int_gadd:Nn \_enumext_item_count_all_viii_int { \\_enumext_joined_item_aux_viii_int }
3675     \dim_set:Nn \\_enumext_joined_width_viii_dim
3676     {
3677       \\_enumext_item_width_viii_dim * \\_enumext_joined_item_viii_int
3678       + ( \\_enumext_labelwidth_viii_dim + \\_enumext_labelsep_viii_dim
3679         + \\_enumext_columns_sep_viii_dim
3680       ) * \\_enumext_joined_item_aux_viii_int
3681     }

```



```

3682     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3683   }
3684   {
3685     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3686     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3687   }
3688 }

```

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

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

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:
3689 \cs_new_protected:Nn \__enumext_start_mini_viii:
3690 {
3691   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3692   {
3693     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3694     {
3695       \linewidth
3696       - \l__enumext_minipage_right_viii_dim
3697       - \l__enumext_minipage_hsep_viii_dim
3698     }
3699     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3700     \dim_gset_eq:NN
3701       \g__enumext_minipage_right_viii_dim
3702       \l__enumext_minipage_right_viii_dim
3703     \__enumext_mini_addvspace_viii:
3704     \nointerlineskip\noindent
3705     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3706   }
3707 }
3708 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3709 {
3710   \bool_if:NT \l__enumext_minipage_active_viii_bool
3711   {
3712     \end{__enumext_mini_env*}
3713     \hfill
3714     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3715   }
3716 }
3717 \__enumext_after_env:nn {keyans*}
3718 {
3719   \bool_if:NT \g__enumext_minipage_active_viii_bool
3720   {
3721     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3722     \par\addvspace { \g__enumext_minipage_right_skip }
3723     \bool_if:NF \g__enumext_minipage_center_viii_bool
3724     {
3725       \centering
3726     }
3727     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3728     \end{__enumext_mini_env*}
3729     \par\addvspace{ \g__enumext_minipage_after_skip }
3730   }
3731   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3732   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3733   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3734   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3735 }

```

(End of definition for \\_\_enumext\_start\_mini\_viii: and \\_\_enumext\_stop\_mini\_viii:)

**keyans\*** First we will generate the environment and we will give a temporary definition to \\_\_enumext\_stop\_item\_tmp\_viii: equal to `\noindent` and next to `\item` equal to `\__enumext_start_item_tmp_viii:` which we will redefine later.

```

3736 \NewDocumentEnvironment{keyans*}{ o }
3737 {
3738   \__enumext_safe_exec_viii:
3739   \__enumext_parse_keys_viii:n {#1}
3740   \__enumext_before_list_viii:
3741   \__enumext_start_list:nn { }

```

```

3742     {
3743         \__enumext_list_arg_two_viii:
3744         \__enumext_before_keys_exec_viii:
3745     }
3746     \__enumext_starred_columns_set_viii:
3747     \item[] \scan_stop:
3748     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3749     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3750 }
3751 {
3752     \__enumext_stop_item_tmp_viii:
3753     \__enumext_remove_extra_parsep_viii:
3754     \__enumext_check_starred_cmd:n { item }
3755     \__enumext_stop_list:
3756     \__enumext_after_list_viii:
3757 }

```

(End of definition for `keyans*`. This function is documented on page 12.)

`\__enumext_safe_exec_viii:` First check the maximum nesting level for the `keyans*` environment.

```

3758 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3759 {
3760     \int_incr:N \l__enumext_keyans_level_h_int
3761     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3762     {
3763         \msg_error:nn { enumext } { nested }
3764     }
3765     \__enumext_keyans_save_start_line:
3766     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3767     \bool_set_false:N \l__enumext_store_active_bool
3768     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3769     {
3770         \msg_error:nn { enumext } { keyans-wrong-level }
3771     }
3772 }

```

(End of definition for `\__enumext_safe_exec_viii:`)

`\__enumext_parse_keys_viii:n` Parse [`<key = val>`] for `keyans*`.

```

3773 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3774 {
3775     \tl_if_novalue:nF {#1}
3776     {
3777         \keys_set:nn { enumext / keyans* } {#1}
3778     }
3779 }

```

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

`\__enumext_before_list_viii:` The function `\__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{<code>}` defined by the `before*` key if it is active, the call the function `\__enumext_start_mini_viii:` handle by `mini-env`.

```

3780 \cs_new_protected:Nn \__enumext_before_list_viii:
3781 {
3782     \__enumext_vspace_above_viii:
3783     \__enumext_before_args_exec_viii:
3784     \__enumext_start_mini_viii:
3785 }

```

(End of definition for `\__enumext_before_list_viii:`)

`\__enumext_after_list_viii:` The function `\__enumext_after_list:` first call the function `\__enumext_stop_mini_viii:`, then apply the `{<code>}` handled by the `after` key together with the `vertical space` handled by the `below` key if they are present.

```

3786 \cs_new_protected:Nn \__enumext_after_list_viii:
3787 {
3788     \__enumext_stop_mini_viii:
3789     \__enumext_after_stop_list_viii:
3790     \__enumext_vspace_below_viii:
3791 }

```

(End of definition for `\__enumext_after_list_viii:`)

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

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

`\__enumext_start_item_tmp_viii:`

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

```
3792 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3793 {
3794   \__enumext_stop_item_tmp_viii:
3795   \int_incr:N \l__enumext_item_column_pos_viii_int
3796   \int_gincr:N \g__enumext_item_count_all_viii_int
3797   \__enumext_item_peek_args_viii:
3798 }
```

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

`\__enumext_item_peek_args_viii:`

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

```
3799 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3800 {
3801   \peek_meaning:NTF (
3802     { \__enumext_joined_item_viii:w }
3803     { \__enumext_joined_item_viii:w (1) }
3804 }
```

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

`\__enumext_joined_item_viii:w`

The function `\__enumext_joined_item_viii:w` will first call the function `\__enumext_starred_joined_item_viii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “\*”, if it is present we will call the function `\__enumext_starred_item_viii:w` otherwise we will call the function `\__enumext_standar_item_viii:w`.

```
3805 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3806 {
3807   \__enumext_starred_joined_item_viii:n {#1}
3808   \peek_meaning_remove:NTF *
3809     { \__enumext_starred_item_viii:w }
3810     { \__enumext_standar_item_viii:w }
3811 }
```

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

`\__enumext_standar_item_viii:w`

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

```
3812 \cs_new_protected:Npn \__enumext_standar_item_viii:w
3813 {
3814   \bool_set_false:N \l__enumext_item_starred_viii_bool
3815   \peek_meaning:NTF [
3816     {
3817       \bool_set_eq:NN
3818       \l__enumext_wrap_label_viii_bool
3819       \l__enumext_wrap_label_opt_viii_bool
3820       \__enumext_start_item_viii:w
3821     }
3822     {
3823       \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```

3824         \legacy_if_set_true:n { @noitemarg }
3825         \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3826     }
3827 }

```

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

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w

```

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

```

3828 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3829 {
3830     \bool_set_true:N \__enumext_item_starred_viii_bool
3831     \bool_set_true:N \__enumext_wrap_label_viii_bool
3832     \peek_meaning:NTF [
3833         { \__enumext_starred_item_viii_aux_i:w }
3834         { \__enumext_starred_item_viii_aux_ii:w }
3835     }

```

The function \\_\_enumext\_starred\_item\_viii\_aux\_i:w will save the optional argument to \item\* in \\_\_enumext\_keyans\_item\_opt\_tl and will save this argument along with the spacing set by the key save-sep in variable \\_\_enumext\_store\_keyans\_label\_tl if present, then call the function \\_\_enumext\_starred\_item\_viii\_aux\_ii:w.

```

3836 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3837 {
3838     \tl_clear:N \__enumext_store_keyans_label_tl
3839     \tl_if_no_value:nF { #1 }
3840     {
3841         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3842         {
3843             \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
3844             \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
3845         }
3846         \tl_set:Ne \__enumext_keyans_item_opt_tl { #1 }
3847     }
3848     \__enumext_starred_item_viii_aux_ii:w
3849 }
3850 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3851 {
3852     \legacy_if_set_true:n { @noitemarg }
3853     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3854 }

```

(End of definition for \\_\_enumext\_starred\_item\_viii:w, \\_\_enumext\_starred\_item\_viii\_aux\_i:w, and \\_\_enumext\_starred\_item\_viii\_aux\_ii:w.)

```
\__enumext_starred_item_exec:
```

The function \\_\_enumext\_starred\_item\_exec: will be in charge of storing the current \langle label \rangle for \item\* followed by the [\langle content \rangle] for \item\*[\langle content \rangle] if present in the \langle sequence \rangle and \langle prop list \rangle set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```

3855 \cs_new_protected:Nn \__enumext_starred_item_exec:
3856 {
3857     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3858     \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3859     \__enumext_keyans_store_ref:
3860     \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3861     \__enumext_keyans_addto_seq_link:
3862     \int_gincr:N \g__enumext_check_starred_cmd_int
3863     \bool_if:NT \__enumext_show_answer_bool
3864     {
3865         \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3866     }
3867     \bool_if:NT \__enumext_show_position_bool
3868     {
3869         \tl_set:Ne \__enumext_mark_answer_sym_tl
3870         {
3871             \group_begin:
3872             \exp_not:N \normalfont
3873             \exp_not:N \footnotesize [ \int_eval:n
3874                 {
3875                     \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3876                 }

```

```

3877         ]
3878     \group_end:
3879 }
3880 \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
3881 }
3882 }

```

(End of definition for `\__enumext_starred_item_exec:`)

### Real definition of `\item` in keyans\*

The implementation at this point is very similar to that of the `enumext*` environment.

```

3883 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3884 {
3885     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3886     \legacy_if:nT { @noitemarg }
3887     {
3888         \legacy_if_set_false:n { @noitemarg }
3889         \legacy_if:nT { @nmbrlist }
3890         {
3891             \bool_if:NT \l__enumext_hyperref_bool
3892             {
3893                 \legacy_if_set_true:n { @hyper@item }
3894             }
3895             \refstepcounter{enumXviii}
3896         }
3897     }

```

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

```

3898     \group_begin:
3899     \lrbox{ \l__enumext_item_text_viii_box }
3900     \bool_if:NF \l__enumext_footnotes_key_bool
3901     {
3902         \__enumext_renew_footnote:
3903     }
3904     \bool_if:NT \l__enumext_item_starred_viii_bool
3905     {
3906         \__enumext_starred_item_exec:
3907     }
3908     \group_begin:
3909     \tl_use:N \l__enumext_label_font_style_viii_tl
3910     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3911     {
3912         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3913         { \__enumext_wrapper_label_viii:n {#1} }
3914     }
3915     {
3916         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3917     }
3918     \group_end:
3919     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3920     \tl_use:N \l__enumext_after_list_args_viii_tl
3921     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3922     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3923     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3924     \bool_if:NT \l__enumext_item_starred_viii_bool
3925     {
3926         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3927         \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3928     }
3929     {
3930         \tl_use:N \l__enumext_fake_item_indent_viii_tl
3931     }
3932 }

```

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

The function `\__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its *contents*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

3933 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3934 {

```

```

3935     \__enumext_endminipage:
3936     \endlrbox
3937     \group_end:
3938     \box_set_wd:Nn \l__enumext_item_text_viii_box
3939     {
3940         \l__enumext_joined_width_viii_dim
3941         + \l__enumext_labelwidth_viii_dim
3942         + \l__enumext_labelsep_viii_dim
3943     }
3944     \int_set:Nn \hbadness { 10000 }
3945     \box_use:N \l__enumext_item_text_viii_box
3946     \bool_if:NF \l__enumext_footnotes_key_bool
3947     {
3948         \__enumext_print_footnote:
3949     }
3950     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3951     {
3952         \par\noindent
3953         \int_zero:N \l__enumext_item_column_pos_viii_int
3954     }
3955     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3956 }

```

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

\\_\_enumext\_remove\_extra\_parsep\_viii:

Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

3957 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3958 {
3959     \int_compare:nNnT
3960     {
3961         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3962     }
3963     =
3964     { \c_zero_int }
3965     {
3966         \par
3967         \vspace{ -\l__enumext_itemsep_viii_skip }
3968         \int_gzero:N \g__enumext_item_count_all_viii_int
3969     }
3970 }

```

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

### 11.36 The command \getkeyans

\getkeyans

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

```

3971 \NewDocumentCommand \getkeyans { m }
3972 {
3973     \exp_args:Ne \__enumext_getkeyans_aux:n
3974     { \tl_to_str:e { \text_expand:n {#1} } }
3975 }

```

(End of definition for \getkeyans. This function is documented on page 14.)

\\_\_enumext\_getkeyans\_aux:n

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

```

3976 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3977 {
3978     \str_if_in:nnTF {#1} { : }
3979     {
3980         \use:e
3981         {
3982             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3983             { {##1} {##2} }
3984         }
3985         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3986     }
3987     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3988 }

```

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

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

```

3989 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3990 {
3991   \prop_if_exist:cF { g__enumext_#1_prop }
3992   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3993   \group_begin:
3994   \prop_item:cn { g__enumext_#1_prop }{#2}
3995   \group_end:
3996 }

```

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

## 11.37 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the *sequence* defined by the `save-ans` key. The first thing we will do is define a set of *filtered keys* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\__enumext_print_keyans_X_tl`. The variable `\__enumext_print_keyans_starred_tl` will have the default *keys* for `\printkeyans*` and will be set by `\setenumext[print*]` and the variable `\__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[print,*]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[print,level]`.

- The reason for storing *keys* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its optional argument, except those related to the first opening level.

```

3997 \cs_generate_variant:Nn \keys_precompile:neN { neN }
3998 \keys_define:nn { enumext / print }
3999 {
4000   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4001               { \__enumext_filter_save_key:n {#1} }
4002               \__enumext_print_keyans_starred_tl, % starred cmd
4003   print* .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4004   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4005               { \__enumext_filter_save_key:n {#1} }
4006               \__enumext_print_keyans_i_tl,
4007   print-1 .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4008   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4009               { \__enumext_filter_save_key:n {#1} }
4010               \__enumext_print_keyans_ii_tl,
4011   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4012   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4013               { \__enumext_filter_save_key:n {#1} }
4014               \__enumext_print_keyans_iii_tl,
4015   print-3 .initial:n = { nosep, label=\roman*, first=\small, font=\small },
4016   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4017               { \__enumext_filter_save_key:n {#1} }
4018               \__enumext_print_keyans_iv_tl,
4019   print-4 .initial:n = { nosep, label=\Alph*, first=\small, font=\small },
4020   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4021               { \__enumext_filter_save_key:n {#1} }
4022               \__enumext_print_keyans_vii_tl, % starred nested
4023   print-* .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4024 }

```

`\printkeyans` Create a user command to print “all stored content” in *sequence* for `\anskey`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `\__enumext_printkeyans:nnn`.

```

4025 \NewDocumentCommand \printkeyans { s O{} m }
4026 {
4027   \group_begin:
4028   \tl_use:N \__enumext_print_keyans_i_tl
4029   \tl_use:N \__enumext_print_keyans_ii_tl
4030   \tl_use:N \__enumext_print_keyans_iii_tl
4031   \tl_use:N \__enumext_print_keyans_iv_tl
4032   \tl_use:N \__enumext_print_keyans_vii_tl
4033   \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }

```



```

4034     \group_end:
4035 }

```

(End of definition for `\printkeyans`. This function is documented on page 14.)

```
\__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 check the starred argument, if it is present it will execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default `\keys` for the environment `enumext*` not nested in the `\sequence`, it will open the environment `enumext*` passing the optional argument to the first level and then will map the `\sequence`, otherwise it will open the environment `enumext` passing the optional argument to the first level and then map the `\sequence`.

```

#1:   starred
#2:   key-val
#3:   seq-name

```

```

4036 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4037 {
4038   \seq_if_exist:cTF { g__enumext_#3_seq }
4039   {
4040     \seq_if_empty:cF { g__enumext_#3_seq }
4041     {
4042       %%\seq_show:c { g__enumext_#3_seq }
4043       \bool_if:nTF {#1}
4044       {
4045         \tl_use:N \l__enumext_print_keyans_starred_tl
4046         \begin{enumext*}[#2]
4047           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4048           \end{enumext*}
4049       }
4050       {
4051         \begin{enumext}[#2]
4052           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4053           \end{enumext}
4054       }
4055     }
4056   }
4057   {
4058     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4059   }
4060 }

```

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

### 11.38 The command `\setenumext`

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

```

4061 \keys_define:nn { enumext / meta-families }
4062 {
4063   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
4064   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
4065   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
4066   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
4067   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} },
4068   enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} },
4069   keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} },
4070   print*    .code:n = { \keys_set:nn { enumext / print    } { print* = {#1} } },
4071   print-1   .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } },
4072   print-2   .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } },
4073   print-3   .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } },
4074   print-4   .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } },
4075   print-*   .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } },
4076   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
4077 }

```

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

```

4078 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4079 {
4080   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4081   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4082 }

```

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

```

4083 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4084 {
4085   \tl_if_novalue:nTF {#1}
4086   {
4087     \seq_map_inline:Nn \c__enumext_all_families_seq
4088     }
4089   {
4090     \seq_clear:N \l__enumext_setkey_tmpa_seq
4091     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4092     \int_set:Nn \l__enumext_setkey_tmpa_int
4093     {
4094       \seq_count:N \l__enumext_setkey_tmpb_seq
4095     }
4096     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4097     {
4098       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4099       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4100       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4101       {
4102         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4103       }
4104     }
4105     {
4106       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4107     }
4108     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4109     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4110     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4111   }
4112   {
4113     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4114   }
4115 }

```

(End of definition for `\setenumext`. This function is documented on page 6.)

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

Internal functions used by the `\setenumext` command.

```

4116 \cs_new_protected:Npn \__enumext_set_parse:n #1
4117 {
4118   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4119   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } %<- max level
4120   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4121   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4122   {
4123     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4124     { \tl_trim_spaces:n {#1} }
4125   }
4126   { \__enumext_set_error:nn {#1} { } }
4127 }
4128 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4129 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

### 11.39 Messages

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

```

4130 \msg_new:nnn { enumext } { package-load }
4131 {
4132   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4133 }
4134 \msg_new:nnn { enumext } { package-not-load }
4135 {
4136   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4137 }
4138 \msg_new:nnn { enumext } { package-load-foot }
4139 {
4140   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4141 }

```

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

```
4142 \msg_new:nnn { enumext } { counters }
4143 {
4144   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4145   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4146 }
```

Message used in the creation of `(prop list)` by `enumext` package.

```
4147 \msg_new:nnn { enumext } { store-prop }
4148 {
4149   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4150 }
4151 \msg_new:nnn { enumext } { store-seq }
4152 {
4153   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4154 }
4155 \msg_new:nnn { enumext } { store-int }
4156 {
4157   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4158 }
4159 \msg_new:nnn { enumext } { prop-seq-int-hook }
4160 {
4161   * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4162   * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4163   * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4164 }
4165 \msg_new:nnn { enumext } { item-answer-hook }
4166 {
4167   * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4168   * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4169   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4170 }
```

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

```
4171 \msg_new:nnn { enumext } { invalid-key }
4172 {
4173   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4174 }
4175 \msg_new:nnn { enumext } { unknown-key-family }
4176 {
4177   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4178 }
```

Messages used in length calculation.

```
4179 \msg_new:nnn { enumext } { width-negative }
4180 {
4181   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4182   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4183 }
4184 \msg_new:nnn { enumext } { width-zero }
4185 {
4186   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4187   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4188 }
```

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

```
4189 \msg_new:nnn { enumext } { list-lengths }
4190 {
4191   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4192   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4193   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4194   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4195   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4196   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4197   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4198   \__enumext_show_length:nnn { skip } { topsep } {#1}
4199   \__enumext_show_length:nnn { skip } { parsep } {#1}
4200   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4201   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4202   *****
4203 }
```

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

```
4204 \msg_new:nnn { enumext } { list-lengths-not-nested }
4205 {
4206     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4207     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4208     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4209     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4210     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4211     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4212     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4213     \__enumext_show_length:nnn { skip } { topsep } {#1}
4214     \__enumext_show_length:nnn { skip } { parsep } {#1}
4215     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4216     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4217     *****
4218 }
```

Messages used by `ref` key.

```
4219 \msg_new:nnn { enumext } { key-ref-empty }
4220 {
4221     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4222 }
```

Messages used by `save-ans` key.

```
4223 \msg_new:nnn { enumext } { save-ans-empty }
4224 {
4225     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4226 }
4227 \msg_new:nnn { enumext } { save-ans-log }
4228 {
4229     * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4230 }
4231 \msg_new:nnn { enumext } { save-ans-log-hook }
4232 {
4233     * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4234 }
4235 \msg_new:nnn { enumext } { save-ans-hook }
4236 {
4237     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4238 }
```

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

```
4239 \msg_new:nnn { enumext } { need-save-ans }
4240 {
4241     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4242 }
4243 \msg_new:nnn { enumext } { items-same-answer }
4244 {
4245     *****\\
4246     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4247     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\\
4248     *****
4249 }
4250 \msg_new:nnn { enumext } { item-greater-answer }
4251 {
4252     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4253     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4254     Items ~ > ~ Answers.
4255 }
4256 \msg_new:nnn { enumext } { item-less-answer }
4257 {
4258     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4259     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4260     Items ~ < ~ Answers.
4261 }
```

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

```
4262 \msg_new:nnn { enumext } { missing-starred }
4263 {
4264     Missing ~ '\c_backslash_str #1*' ~ #2.
4265 }
```

```

4266 \msg_new:nnn { enumext } { many-starred }
4267 {
4268   Many ~ '\c_backslash_str #1*' ~ #2.
4269 }

```

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

```

4270 \msg_new:nnn { enumext } { list-too-deep }
4271 {
4272   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4273   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4274 }

```

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

```

4275 \msg_new:nnn { enumext } { anskey-wrong-place }
4276 {
4277   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4278   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4279 }
4280 \msg_new:nnn { enumext } { anspic-wrong-place }
4281 {
4282   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4283   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4284 }
4285 \msg_new:nnn { enumext } { command-wrong-place }
4286 {
4287   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4288   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4289 }

```

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

```

4290 \msg_new:nnn { enumext } { keyans-nested }
4291 {
4292   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4293 }
4294 \msg_new:nnn { enumext } { keyans-wrong-level }
4295 {
4296   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4297   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4298 }
4299 \msg_new:nnn { enumext } { wrong-place }
4300 {
4301   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4302   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4303 }
4304 \msg_new:nnn { enumext } { keyanspic-nested }
4305 {
4306   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4307 }
4308 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4309 {
4310   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4311   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4312 }

```

Messages used by `\getkeyans` command.

```

4313 \msg_new:nnn { enumext } { undefined-storage-anskey }
4314 {
4315   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4316 }

```

Messages used by `\miniright` command.

```

4317 \msg_new:nnn { enumext } { missing-miniright }
4318 {
4319   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4320   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4321 }
4322 \msg_new:nnn { enumext } { wrong-miniright-place }
4323 {
4324   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4325   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4326 }
4327 \msg_new:nnn { enumext } { wrong-miniright-use }
4328 {

```

```

4329     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4330     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4331 }

```

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

```

4332 \msg_new:nnn { enumext } { nested }
4333 {
4334     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4335 }
4336 \msg_new:nnn { enumext } { item-joined }
4337 {
4338     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4339 }
4340 \msg_new:nnn { enumext } { item-joined-columns }
4341 {
4342     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4343 }

```

## 11.40 Finish package

Finish package implementation.

```

4344 \file_input_stop:
4345 </package>

```

## 12 Index of Implementation

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

Symbols	
<code>\*</code>	201
<code>\+</code>	193
<code>\-</code>	193
<code>\\</code>	209, 3167, 4144, 4161, 4162, 4167, 4168, 4181, 4186, 4191, 4206, 4245, 4246, 4247, 4252, 4253, 4258, 4259, 4272, 4277, 4282, 4287, 4296, 4301, 4310, 4319, 4324, 4329
A	
<code>above</code>	<u>1365</u>
<code>above*</code>	<u>1365</u>
<code>\addvspace</code>	1012, 1040, 1163, 1242, 1305, 1311, 1339, 1356, 3006, 3021, 3123, 3138, 3364, 3371, 3722, 3729
<code>after</code>	850
<code>align</code>	<u>467</u>
<code>\Alph</code>	33, <u>38</u>
<code>\Alph</code>	419, 528, 573, 641, 4019
<code>\alph</code>	33, <u>38</u>
<code>\alph</code>	420, 526, 4011, 4023
<code>\anskey</code>	12, 69, <u>2179</u>
<code>\anspic</code>	13, 89, 90, <u>3145</u>
<code>\anspic*</code>	64
<code>\arabic</code>	28, <u>33</u>
<code>\arabic</code>	418, 525, 572, 4003, 4007
B	
<code>\baselineskip</code>	45
<code>\baselineskip</code>	2139, 2147
<code>before</code>	850
<code>before*</code>	850
<code>below</code>	<u>1365</u>
<code>below*</code>	<u>1365</u>
bool commands:	
<code>\bool_gset_false:N</code>	307, 308, 309, 3373, 3377, 3731
<code>\bool_gset_true:N</code>	221, 230, 954, 1856, 1862, 3356, 3374, 3714, 3732
<code>\bool_if:NTF</code>	359, 371, 388, 1387, 1401, 1414, 1425, 1436, 1447, 1458, 1469, 1522, 1539, 1544, 1552, 1579, 1617, 1622, 1629, 1633, 1655, 1660, 1668, 1675, 1706, 1714, 1806, 1930, 2012, 2022, 2101, 2125, 2132, 2160, 2191, 2204, 2214, 2234, 2359, 2370, 2374, 2413, 2428, 2503, 2514, 2518, 2631, 2661, 2735, 2751, 2813, 2823, 2853, 2858, 2940, 2990, 3004, 3012, 3051, 3108, 3121, 3129, 3147, 3352, 3361, 3365, 3441, 3451, 3534, 3539, 3547, 3551, 3566, 3595, 3710, 3719, 3723, 3863, 3867, 3891, 3900, 3904, 3910, 3924, 3946
<code>\bool_if:nTF</code>	1340, 1357, 2242, 2672, 2707, 2771, 3168, 4043
<code>\bool_if_p:N</code>	239, 253, 1686, 1687, 1695, 1696, 1819, 1841, 1853, 1854, 1859, 1860, 2225, 2268, 2269, 2293, 2302, 2303, 2315, 2331, 2490, 2491, 2528, 2529, 2913, 2926, 2928, 3175, 3176
<code>\bool_lazy_all:nTF</code>	237, 251, 1817, 1839, 2291, 2300, 2313, 2329, 2911, 2924
<code>\bool_lazy_and:nnTF</code>	217, 226, 1685, 1694, 1852, 1858, 2224, 2267, 2489
<code>\bool_lazy_or:nnTF</code>	1747, 1754, 2527, 3174
<code>\bool_new:N</code>	25, 26, 27, 28, 29, 30, 31, 51, 61, 82, 87, 88, 93, 94, 97, 115, 118, 121, 122, 131, 132, 133, 141, 142, 156, 167, 169
<code>\bool_not_p:n</code>	218, 227, 2226, 2318, 2333, 2914, 2915, 2927
<code>\bool_set_eq:NN</code>	2639, 2687, 3484, 3817
<code>\bool_set_false:N</code>	368, 1791, 1792, 3027, 3059, 3141, 3206, 3224, 3436, 3481, 3767, 3814
<code>\bool_set_true:N</code>	244, 258, 350, 354, 460, 778, 1371, 1376, 1642, 1764, 1765, 2044, 2052, 2635, 2665, 2683, 2695, 2889, 2920, 2933, 2959, 3056, 3083, 3341, 3410, 3490, 3497, 3498, 3699, 3823, 3830, 3831
box commands:	
<code>\box_dp:N</code>	1059, 1063, 1067, 1078, 1082, 1093, 1102, 1108, 1118, 1131, 1137, 1143, 1174, 1175, 1176, 1179, 1189, 1193, 1202, 1209, 1214, 1222, 1251, 1252, 1255, 1262, 1275, 1283, 1289, 1297, 3236
<code>\box_new:N</code>	58, 162
<code>\box_set_wd:Nn</code>	3587, 3938
<code>\box_use:N</code>	3594, 3945
<code>\box_wd:N</code>	426
C	
<code>\c</code>	201, 202, 678, 680, 692, 694
<code>\cB</code>	202
<code>\cE</code>	202
<code>\centering</code>	1342, 1359, 3257, 3367, 3725
<code>check-ans</code>	<u>1783</u>
Document class:	
<code>article</code>	39
clist commands:	
<code>\clist_const:Nn</code>	174
<code>\clist_map_function:nN</code>	3244
<code>\clist_map_inline:Nn</code>	466, 720, 783, 849, 864, 945, 1381
<code>\clist_map_inline:nn</code>	36, 47, 66, 72, 84, 96, 120, 150, 173, 491, 508, 788, 960, 1487, 1731, 1797, 1991, 2009, 2041, 2288, 2422, 2589, 2800, 2803, 2830, 2840, 2843, 2863, 4119
<code>\columnbreak</code>	70
<code>\columnbreak</code>	2228
<code>columns</code>	<u>929</u>
<code>columns-sep</code>	<u>929</u>
<code>\columnsep</code>	85, 88
<code>\columnsep</code>	2984, 3105
<code>\columnseprule</code>	85, 88
<code>\columnseprule</code>	2988, 3107
Commands provide by <b>enumext</b> :	
<code>\anskey</code>	25, 26, 60, 61, 66, 67, 69, 71–73, 75, 84, 96, 106, 107, 112
<code>\anspic*</code>	25, 26, 64, 73, 74, 90–92, 106, 107
<code>\anspic</code>	67, 89–91, 112
<code>\getkeyans</code>	67, 106, 112
<code>\item*</code>	25, 26, 64, 67, 73, 74, 78, 79, 97, 104, 106, 107
<code>\itemwidth</code>	92, 93, 100
<code>\item</code>	77, 79, 93, 96–98, 100, 103
<code>\miniright</code>	25, 43, 51, 52, 85, 86, 88, 89, 112
<code>\printkeyans*</code>	107
<code>\printkeyans</code>	26, 67, 107
<code>\setenumext</code>	26, 107–110
Counters defined by <b>enumext</b> :	
<code>enumXiii</code>	24, 33
<code>enumXii</code>	24, 33



enumXiv	24, 33	3097, 3249, 3267, 3274, 3317, 3335, 3522, 3625, 3632, 3675, 3693
enumXi	24, 33	\dim_set_eq:NN 516, 563, 634, 638, 2654, 2802, 2842, 2984, 3105, 3324, 3327, 3328, 3513, 3682, 3685, 3686
enumXviii	24, 33	\dim_use:N 792, 800, 1332, 1338, 2168, 2171, 2176, 2724, 2726, 2951, 2956, 2957, 2964, 2974, 2978, 2979, 2981
enumXvii	24, 33, 98	\dim_zero:N 2988, 3107, 3228, 3229, 3230
enumXvi	24, 33	\dim_zero_new:N 3280, 3638
enumXv	24, 33	\c_zero_dim 794, 808, 820, 832, 1332, 1350, 2254, 2761, 2766, 2772, 2779, 2951, 2974, 3077, 3095, 3265, 3333, 3623, 3691
cs commands:		<b>E</b>
\cs_generate_variant:Nn	428, 444, 684, 700, 2093, 2098, 2178, 2790, 3246, 3997	\end .. 1335, 1353, 2127, 2162, 3003, 3020, 3120, 3137, 3354, 3370, 3712, 3728, 4048, 4053
\cs_if_exist:NTF	398	\endlist .. 31
\cs_new:Nn	187	\endlist .. 335
\cs_new:Npn	205, 1488, 1497, 1506, 2056, 2065, 2073	\endlrbox .. 3585, 3936
\cs_new_eq:NN	334, 335, 336, 340, 341, 373, 374, 377, 378	\endminipage .. 31
\cs_new_protected:Nn	197, 211, 235, 266, 293, 299, 305, 311, 317, 325, 345, 549, 612, 664, 865, 869, 873, 877, 881, 885, 889, 893, 897, 901, 905, 909, 913, 917, 921, 925, 961, 973, 997, 1014, 1025, 1049, 1124, 1148, 1165, 1227, 1244, 1266, 1301, 1307, 1382, 1396, 1410, 1421, 1432, 1443, 1454, 1465, 1550, 1653, 1666, 1683, 1704, 1732, 1737, 1762, 1802, 1812, 1850, 1865, 1872, 1881, 1886, 1891, 1896, 1905, 1910, 1915, 1920, 2099, 2123, 2130, 2158, 2165, 2279, 2411, 2426, 2454, 2487, 2523, 2535, 2543, 2594, 2598, 2617, 2668, 2703, 2719, 2729, 2745, 2883, 2909, 2938, 2945, 2968, 2998, 3010, 3049, 3073, 3091, 3116, 3127, 3164, 3208, 3222, 3242, 3247, 3263, 3331, 3350, 3402, 3423, 3430, 3439, 3449, 3466, 3606, 3621, 3689, 3708, 3758, 3780, 3786, 3799, 3855, 3957	\endminipage .. 341
\cs_new_protected:Npn	179, 183, 381, 396, 413, 423, 429, 529, 574, 646, 671, 685, 1329, 1348, 1518, 1537, 1607, 1640, 1742, 1939, 2010, 2020, 2042, 2050, 2085, 2094, 2211, 2356, 2368, 2390, 2464, 2508, 2627, 2645, 2679, 2691, 2759, 2793, 2833, 2892, 3069, 3217, 3282, 3413, 3472, 3479, 3495, 3503, 3508, 3520, 3640, 3773, 3805, 3812, 3828, 3836, 3850, 3976, 3989, 4036, 4116, 4128	enumext .. 5, <u>2864</u>
\cs_new_protected_nopar:Nn	3459, 3582, 3792, 3933	enumext internal commands:
\cs_new_protected_nopar:Npn	3526, 3883	\l__enumext__check_start_line_env_tl ... 29
\cs_set:Nn	2361	\l__enumext__ref_the_count_tl .. 35
\cs_set:Npn	2289, 2327, 3982	\l__enumext__resume_name_tl .. 56
\cs_set_eq:NN	3392, 3393, 3528, 3748, 3749, 3885	\__enumext_add_pre_parsep: ... 44, 971, <u>973</u> , 973
\cs_set_protected:Nn	789, 805, 817, 829	\__enumext_after_args_exec: . 42, <u>865</u> , 877, 2876
\cs_set_protected:Npn	32, 41, 59, 67, 79, 85, 112, 146, 154, 445, 467, 496, 509, 556, 701, 721, 765, 784, 841, 850, 929, 946, 1365, 1476, 1723, 1783, 1956, 1992, 2028, 2281, 2415, 2578, 2791, 2831	\__enumext_after_args_exec_v: . 42, 43, <u>881</u> , 893, 3042
\cs_to_str:N	415, 438	\__enumext_after_args_exec_vii: ... <u>897</u> , 921
<b>D</b>		\__enumext_after_args_exec_viii: .. 925
\d .. 193		\__enumext_after_env:nn 63, 86, 99, <u>183</u> , 183, 3030, 3359, 3620, 3717
\DeclareDocumentEnvironment .. 1042		\__enumext_after_hyperref: ... 31, 343, <u>345</u> , 345
dim commands:		\__enumext_after_list: .. 86, 96, 102, 2881, <u>3010</u> , 3010
\dim_abs:n .. 2764, 2769		\l__enumext_after_list_args_v_tl .. 895
\dim_add:Nn .. 3227		\l__enumext_after_list_args_vii_tl 923, 3576
\dim_compare:nNnTF . 791, 807, 819, 831, 1331, 1350, 2761, 2766, 2772, 2778, 2780, 2782, 2950, 2973, 3077, 3095, 3219, 3265, 3333, 3623, 3691		\l__enumext_after_list_args_viii_tl 927, 3920
\dim_compare:nTF .. 2252		\__enumext_after_list_v: .. 89, 3047, <u>3127</u> , 3127
\dim_gset_eq:NN .. 3342, 3700		\__enumext_after_list_vii: ... 3400, <u>3430</u> , 3430
\dim_gzero:N .. 3376, 3734		\__enumext_after_list_viii: .. 3756, <u>3786</u> , 3786
\dim_new:N .... 55, 62, 63, 64, 81, 127, 163, 164, 170		\__enumext_after_star_env:nn .. 94
\dim_set:Nn .. 426, 779, 2659, 2764, 2769, 2771, 2774, 2775, 2779, 2781, 2784, 2785, 2787, 2953, 2976, 3079,		\__enumext_after_stop_list: ... 42, 43, <u>865</u> , 873, 3025
		\__enumext_after_stop_list_v: 42, <u>881</u> , 889, 3142
		\l__enumext_after_stop_list_v_tl .. 891
		\__enumext_after_stop_list_vii: <u>897</u> , 913, 3433
		\l__enumext_after_stop_list_vii_tl ... 915
		\__enumext_after_stop_list_viii: . 917, 3789
		\l__enumext_after_stop_list_viii_tl ... 919
		\l__enumext_align_label_vii_str .. 3568, 3572
		\l__enumext_align_label_viii_str . 3912, 3916
		\l__enumext_align_label_X_str .. 154
		\c__enumext_all_envs_clist .. <u>174</u> , 466, 720, 783, 849, 864, 945, 1381
		\c__enumext_all_families_seq .. 108, 4078, 4087, 4109
		\__enumext_anskey_wrapper:n .. 1960, 2366
		\__enumext_at_begin_document:n .. 31, <u>179</u> , 179, 332, 338
		\__enumext_before_args_exec: 41, <u>865</u> , 865, 2948
		\__enumext_before_args_exec_v: .. 42, <u>881</u> , 881, 3076

```

\__enumext_before_args_exec_vii: .. 897, 897,
    3427
\__enumext_before_args_exec_viii: 901, 3783
\__enumext_before_keys_exec: 41, 865, 869, 2874
\__enumext_before_keys_exec_v: .. 42, 881, 885,
    3040
\__enumext_before_keys_exec_vii: ..... 897
\__enumext_before_keys_exec_vii: 42, 905, 3388
\__enumext_before_keys_exec_viii: .. 42, 909,
    3744
\__enumext_before_list: ... 85, 2868, 2945, 2945
\__enumext_before_list_v: . 87, 3035, 3073, 3073
\__enumext_before_list_vii: 95, 3383, 3423, 3423
\__enumext_before_list_viii: .. 102, 3740, 3780,
    3780
\l__enumext_before_no_starred_key_v_tl 887
\l__enumext_before_no_starred_key_vii-
    tl ..... 907
\l__enumext_before_no_starred_key_viii-
    tl ..... 911
\l__enumext_before_starred_key_v_tl ... 883
\l__enumext_before_starred_key_vii_tl . 899
\l__enumext_before_starred_key_viii_tl 903
\__enumext_calc_hspace:NNNNNNN 81, 2759, 2759,
    2790, 2795, 2835
\__enumext_check_ans_active: 61, 85, 1802, 1802,
    2949, 3426
\g__enumext_check_ans_item_tl ..... 75
\g__enumext_check_ans_key_bool 62, 63, 131, 307,
    1856, 1862, 1930
\l__enumext_check_ans_key_bool 62, 77, 78, 131,
    1787, 1792, 1853, 1859
\__enumext_check_ans_key_hook: 62, 1850, 1850,
    3024, 3434
\__enumext_check_ans_level: 61, 1802, 1808, 1812
\__enumext_check_ans_log: 62, 63, 1896, 1896, 1934
\__enumext_check_ans_log_msg_greater: 1896,
    1902, 1915
\__enumext_check_ans_log_msg_less: 1896, 1900,
    1905
\__enumext_check_ans_log_msg_same_ok: 1896,
    1901, 1910
\__enumext_check_ans_msg_greater: 1872, 1878,
    1891
\__enumext_check_ans_msg_less: 1872, 1876, 1881
\__enumext_check_ans_msg_same_ok: 1872, 1877,
    1886
\__enumext_check_ans_show: .. 62, 63, 1872, 1872,
    1932
\g__enumext_check_ans_show_bool ..... 86
\l__enumext_check_answers_bool ... 60, 61, 131,
    1765, 1791, 1806, 2101, 2125, 2132, 2160, 2204, 2503,
    2631, 2661, 3539
\__enumext_check_starred_cmd:n 29, 64, 75, 1939,
    1939, 3045, 3203, 3754
\g__enumext_check_starred_cmd_int 131, 1942,
    1948, 1953, 2701, 3173, 3862
\l__enumext_check_start_line_env_tl 131, 272,
    279, 286, 1945, 1951, 1954
\l__enumext_columns_sep_v_dim 3095, 3097, 3105
\l__enumext_columns_sep_vii_dim .. 3265, 3267,
    3276, 3321, 3604
\l__enumext_columns_sep_viii_dim . 3623, 3625,
    3634, 3679, 3955
\l__enumext_columns_v_int 1170, 3093, 3101, 3113,
    3118
\l__enumext_columns_vii_int .. 3270, 3273, 3277,
    3285, 3289, 3292, 3298, 3304, 3308, 3599, 3610
\l__enumext_columns_viii_int . 3628, 3631, 3635,
    3643, 3647, 3650, 3656, 3662, 3666, 3950, 3961
\g__enumext_count_item_with_ans_int .... 69
\l__enumext_counter_i_tl ..... 32, 405
\l__enumext_counter_ii_tl ..... 32, 406
\l__enumext_counter_iii_tl ..... 32, 407
\l__enumext_counter_iv_tl ..... 32, 408
\c__enumext_counter_style_tl ..... 28, 37, 199
\g__enumext_counter_styles_tl . 24, 33, 55, 416,
    434
\l__enumext_counter_v_tl ..... 32, 409, 654
\l__enumext_counter_vi_tl ..... 32, 410
\l__enumext_counter_vii_tl ..... 32, 411, 584
\l__enumext_counter_viii_tl ..... 32, 412, 601
\l__enumext_current_widest_dim 24, 55, 440, 517,
    564, 635, 639
\__enumext_default_item:n ... 2627, 2627, 2676
\__enumext_define_counters:Nn 24, 396, 396, 405,
    406, 407, 408, 409, 410, 411, 412
\__enumext_endminipage: 31, 338, 341, 1048, 3259,
    3584, 3935
\g__enumext_envir_name_tl 29, 136, 245, 259, 315,
    1735, 1740, 1750, 1884, 1889, 1894, 1908, 1913, 1918
\__enumext_execute_after_env: .. 30, 59, 62, 63,
    1920, 1920, 3030, 3620
\__enumext_fake_item: ..... 789, 789, 2822
\l__enumext_fake_item_indent_v_dim 808, 813
\l__enumext_fake_item_indent_v_tl 810, 2684,
    2688, 2696
\l__enumext_fake_item_indent_vii_dim 820, 825
\l__enumext_fake_item_indent_vii_tl 822, 3580
\l__enumext_fake_item_indent_viii_dim . 832,
    837, 3927
\l__enumext_fake_item_indent_viii_tl .. 834,
    3926, 3930
\l__enumext_fake_item_indent_X_tl ..... 85
\__enumext_fake_item_vii: .... 789, 817, 2852
\__enumext_fake_item_viii: .... 789, 829, 2857
\__enumext_filter_save_key:n .. 66, 2017, 2025,
    2048, 2054, 2056, 2056, 4001, 4005, 4009, 4013, 4017,
    4021
\__enumext_filter_save_key_key:n .. 66, 2056,
    2061, 2065
\__enumext_filter_save_key_pair:nn 67, 2056,
    2062, 2073
\__enumext_filter_series:n 54, 1488, 1488, 1530,
    1542, 1547
\__enumext_filter_series_key:n 55, 1488, 1493,
    1497
\__enumext_filter_series_pair:nn .. 55, 1488,
    1494, 1506
\g__enumext_footnote_arg_seq . 151, 2600, 2613,
    2623
\g__enumext_footnote_int . 151, 2607, 2610, 2612,
    2614
\g__enumext_footnote_int_seq . 151, 2601, 2614,
    2619, 2622
\__enumext_footnotes_key_bool ..... 31
\l__enumext_footnotes_key_bool 26, 31, 98, 141,
    354, 359, 368, 3547, 3595, 3900, 3946

```

`\__enumext_footnotetext:nn` ... [2594](#), [2594](#), [2624](#)  
`\__enumext_getkeyans:nn` .. [107](#), [3985](#), [3989](#), [3989](#)  
`\__enumext_getkeyans_aux:n` [106](#), [3973](#), [3976](#), [3976](#)  
`\l__enumext_hyperref_bool` . [26](#), [31](#), [32](#), [141](#), [350](#),  
[371](#), [388](#), [2269](#), [2491](#), [3534](#), [3891](#)  
`\__enumext_hypertarget:nn` [32](#), [345](#), [373](#), [377](#), [393](#)  
`\__enumext_if_is_int:n` ..... [191](#)  
`\__enumext_if_is_int:nTF` ..... [191](#), [673](#), [687](#)  
`\__enumext_is_not_nested:` [28](#), [83](#), [211](#), [211](#), [2885](#),  
[3404](#)  
`\__enumext_is_on_first_level:` . [29](#), [83](#), [211](#), [235](#),  
[2890](#), [3411](#)  
`\g__enumext_item_anskey_int` .. [75](#), [131](#), [302](#), [329](#),  
[330](#), [1869](#), [2206](#), [2505](#)  
`\__enumext_item_answer_diff:` [62](#), [63](#), [1865](#), [1865](#),  
[1927](#)  
`\g__enumext_item_answer_diff_int` . [62](#), [63](#), [140](#),  
[303](#), [1867](#), [1874](#), [1898](#)  
`\l__enumext_item_column_pos_vii_int` [96](#), [3292](#),  
[3298](#), [3304](#), [3308](#), [3315](#), [3462](#), [3599](#), [3602](#)  
`\l__enumext_item_column_pos_viii_int` .. [103](#),  
[3650](#), [3656](#), [3662](#), [3666](#), [3673](#), [3795](#), [3950](#), [3953](#)  
`\l__enumext_item_column_pos_X_int` ..... [154](#)  
`\g__enumext_item_count_all_vii_int` [96](#), [3316](#),  
[3463](#), [3610](#), [3617](#)  
`\g__enumext_item_count_all_viii_int` [103](#), [3674](#),  
[3796](#), [3961](#), [3968](#)  
`\g__enumext_item_count_all_X_int` ..... [154](#)  
`\g__enumext_item_number_int` [61](#), [62](#), [131](#), [301](#), [328](#),  
[330](#), [1823](#), [1827](#), [1830](#), [1833](#), [1845](#), [1869](#), [2633](#), [2663](#),  
[3541](#)  
`\__enumext_item_peek_args_vii:` [96](#), [3464](#), [3466](#),  
[3466](#)  
`\__enumext_item_peek_args_viii:` .. [103](#), [3797](#),  
[3799](#), [3799](#)  
`\__enumext_item_starred:` .. [80](#), [2719](#), [2719](#), [2737](#)  
`\l__enumext_item_starred_vii_bool` [3481](#), [3497](#),  
[3551](#)  
`\l__enumext_item_starred_viii_bool` [3814](#), [3830](#),  
[3904](#), [3924](#)  
`\l__enumext_item_starred_X_bool` ..... [154](#)  
`\__enumext_item_std:w` [31](#), [77–79](#), [92](#), [332](#), [336](#), [2636](#),  
[2642](#), [2666](#), [2684](#), [2688](#), [2696](#), [3240](#)  
`\g__enumext_item_symbol_aux_vii_tl` [3505](#), [3553](#),  
[3556](#), [3560](#), [3562](#)  
`\g__enumext_item_symbol_aux_X_tl` ..... [154](#)  
`\l__enumext_item_symbol_sep_vii_dim` .. [3514](#),  
[3522](#), [3559](#), [3561](#)  
`\g__enumext_item_symbol_tl` [24](#), [78](#), [48](#), [2651](#), [2725](#),  
[2742](#)  
`\l__enumext_item_symbol_vii_tl` ..... [3556](#)  
`\l__enumext_item_text_vii_box` [3546](#), [3587](#), [3594](#)  
`\l__enumext_item_text_viii_box` [3899](#), [3938](#), [3945](#)  
`\l__enumext_item_text_X_box` ..... [154](#)  
`\l__enumext_item_width_vii_dim` ... [3274](#), [3319](#),  
[3327](#), [3328](#)  
`\l__enumext_item_width_viii_dim` .. [3632](#), [3677](#),  
[3685](#), [3686](#)  
`\l__enumext_item_width_X_dim` ..... [154](#)  
`\l__enumext_itemindent_X_dim` ..... [59](#)  
`\l__enumext_itemsep_vii_skip` ..... [3616](#)  
`\l__enumext_itemsep_viii_skip` ..... [3967](#)  
`\l__enumext_joined_item_aux_vii_int` .. [3313](#),  
[3314](#), [3315](#), [3316](#), [3322](#)  
`\l__enumext_joined_item_aux_viii_int` . [3671](#),  
[3672](#), [3673](#), [3674](#), [3680](#)  
`\l__enumext_joined_item_aux_X_int` .... [154](#)  
`\__enumext_joined_item_vii:w` [96](#), [97](#), [3469](#), [3470](#),  
[3472](#), [3472](#)  
`\l__enumext_joined_item_vii_int` .. [3284](#), [3285](#),  
[3288](#), [3290](#), [3296](#), [3301](#), [3306](#), [3311](#), [3313](#), [3319](#)  
`\__enumext_joined_item_viii:w` . [103](#), [3802](#), [3803](#),  
[3805](#), [3805](#)  
`\l__enumext_joined_item_viii_int` . [3642](#), [3643](#),  
[3646](#), [3648](#), [3654](#), [3659](#), [3664](#), [3669](#), [3671](#), [3677](#)  
`\l__enumext_joined_item_X_int` ..... [154](#)  
`\l__enumext_joined_width_vii_dim` . [3317](#), [3324](#),  
[3327](#), [3577](#), [3589](#)  
`\l__enumext_joined_width_viii_dim` [3675](#), [3682](#),  
[3685](#), [3921](#), [3940](#)  
`\l__enumext_joined_width_X_dim` ..... [154](#)  
`\__enumext_keyans_addto_prop:n` [73](#), [2390](#), [2390](#),  
[2698](#), [3170](#)  
`\__enumext_keyans_addto_seq:n` . [74](#), [2464](#), [2464](#),  
[2700](#), [3172](#)  
`\__enumext_keyans_addto_seq_link:` [2464](#), [2485](#),  
[2487](#), [3861](#)  
`\__enumext_keyans_anspic_code:nnn` . [90](#), [3161](#),  
[3164](#), [3164](#)  
`\__enumext_keyans_default_item:n` .. [79](#), [2679](#),  
[2679](#), [2715](#)  
`\l__enumext_keyans_env_bool` [20](#), [2914](#), [2927](#), [3056](#),  
[3141](#)  
`\__enumext_keyans_fake_item:` .. [789](#), [805](#), [2812](#)  
`\l__enumext_keyans_item_opt_tl` . [104](#), [97](#), [2512](#),  
[2525](#), [2531](#), [3846](#)  
`\l__enumext_keyans_level_h_int` .. [20](#), [594](#), [621](#),  
[2442](#), [3760](#), [3761](#)  
`\l__enumext_keyans_level_int` .. [20](#), [1323](#), [2195](#),  
[2437](#), [3055](#), [3060](#), [3155](#)  
`\__enumext_keyans_make_label:` [34](#), [80](#), [2745](#), [2745](#),  
[2810](#)  
`\__enumext_keyans_mini_addvspace:` [49](#), [88](#), [1227](#),  
[1227](#), [3085](#)  
`\__enumext_keyans_mini_right_cmd:n` [52](#), [1325](#),  
[1348](#), [1348](#)  
`\__enumext_keyans_mini_set_vskip:` . [48](#), [1165](#),  
[1165](#), [1229](#)  
`\__enumext_keyans_multi_addvspace:` [88](#), [1014](#),  
[1025](#), [3110](#)  
`\__enumext_keyans_multi_set_vskip:` [45](#), [1014](#),  
[1014](#), [1027](#)  
`\__enumext_keyans_multicols_start:` [88](#), [3089](#),  
[3091](#), [3091](#)  
`\__enumext_keyans_multicols_stop:` . [89](#), [1352](#),  
[3116](#), [3116](#), [3140](#)  
`\__enumext_keyans_parse_keys:n` [3034](#), [3069](#), [3069](#)  
`\l__enumext_keyans_pic_above_int` . [126](#), [3250](#),  
[3251](#), [3253](#)  
`\l__enumext_keyans_pic_above_skip` .. [92](#), [126](#),  
[3194](#), [3234](#)  
`\__enumext_keyans_pic_arg_two:` [91](#), [3192](#), [3222](#),  
[3222](#)  
`\l__enumext_keyans_pic_below_int` . [126](#), [3250](#),  
[3251](#), [3254](#)  
`\l__enumext_keyans_pic_body_seq` .. [90–92](#), [126](#),  
[3159](#), [3199](#), [3258](#)  
`\__enumext_keyans_pic_do:n` [92](#), [3199](#), [3201](#), [3242](#),

3242, 3246  
 \l\_\_enumext\_keyans\_pic\_level\_int .. 20, 1315, 2199, 2393, 2432, 2467, 2545, 3210, 3211  
 \\_\_enumext\_keyans\_pic\_row:n 92, 3244, 3247, 3247  
 \\_\_enumext\_keyans\_pic\_safe\_exec: .. 91, 3188, 3208, 3208  
 \\_\_enumext\_keyans\_pic\_skip\_abs:N .. 91, 3217, 3217, 3233  
 \l\_\_enumext\_keyans\_pic\_width\_dim . 126, 3249, 3256  
 \\_\_enumext\_keyans\_redefine\_item: .. 79, 2703, 2703, 2809  
 \\_\_enumext\_keyans\_ref: ..... 37, 646, 664, 2811  
 \\_\_enumext\_keyans\_ref:n ..... 37, 643, 646, 646  
 \\_\_enumext\_keyans\_safe\_exec: . 3033, 3049, 3049  
 \\_\_enumext\_keyans\_save\_start\_line: . 29, 266, 266, 3057, 3215, 3765  
 \\_\_enumext\_keyans\_show\_ans: .. 2508, 2516, 2535  
 \\_\_enumext\_keyans\_show\_item\_opt: . 2508, 2523, 2696, 3184, 3927  
 \\_\_enumext\_keyans\_show\_left:n . 79, 2508, 2508, 2694, 3179  
 \\_\_enumext\_keyans\_show\_pos: .. 2508, 2520, 2543  
 \\_\_enumext\_keyans\_starred\_item:n .. 79, 2691, 2691, 2711  
 \\_\_enumext\_keyans\_store\_ref: .. 73, 2411, 2411, 2699, 3171, 3859  
 \\_\_enumext\_keyans\_store\_ref\_aux\_i: 74, 2411, 2423, 2426  
 \\_\_enumext\_keyans\_store\_ref\_aux\_ii: 74, 2411, 2452, 2454  
 \l\_\_enumext\_keyans\_tmpa\_tl .. 25, 97, 2693, 2697  
 \\_\_enumext\_keyans\_wrapper\_opt:n .. 1963, 2531  
 \l\_\_enumext\_label\_copy\_i\_tl .. 2323, 2430, 2435, 2440, 2445  
 \l\_\_enumext\_label\_copy\_v\_tl ..... 2440  
 \l\_\_enumext\_label\_copy\_vi\_tl ..... 2435  
 \l\_\_enumext\_label\_copy\_vii\_tl 2298, 2309, 2340, 2430  
 \l\_\_enumext\_label\_copy\_viii\_tl ..... 2445  
 \l\_\_enumext\_label\_copy\_X\_tl ..... 143  
 \l\_\_enumext\_label\_fill\_left\_v\_tl ..... 2749  
 \l\_\_enumext\_label\_fill\_left\_X\_tl ..... 85  
 \l\_\_enumext\_label\_fill\_right\_v\_tl .... 2756  
 \l\_\_enumext\_label\_fill\_right\_X\_tl ..... 85  
 \l\_\_enumext\_label\_font\_style\_v\_tl 2750, 3183  
 \l\_\_enumext\_label\_font\_style\_vii\_tl ... 3565  
 \l\_\_enumext\_label\_font\_style\_viii\_tl .. 3909  
 \l\_\_enumext\_label\_i\_tl ..... 509  
 \l\_\_enumext\_label\_ii\_tl ..... 509  
 \l\_\_enumext\_label\_iii\_tl ..... 509  
 \l\_\_enumext\_label\_iv\_tl ..... 509  
 \\_\_enumext\_label\_style:Nnn 24, 33, 429, 429, 444, 514, 561, 632, 636  
 \l\_\_enumext\_label\_v\_tl .. 73, 74, 629, 2398, 2472, 2537, 2572, 2693, 2697, 3037, 3178, 3180  
 \l\_\_enumext\_label\_vi\_tl . 73, 74, 629, 2395, 2469, 3178, 3180, 3184  
 \l\_\_enumext\_label\_vii\_tl . 556, 3492, 3517, 3524  
 \l\_\_enumext\_label\_viii\_tl 556, 3825, 3853, 3857  
 \l\_\_enumext\_label\_width\_by\_box .. 55, 425, 426  
 \\_\_enumext\_label\_width\_by\_box:Nn 33, 423, 423, 428, 440, 697  
 \l\_\_enumext\_labelsep\_i\_dim ... 2540, 2575, 3865, 3880  
 \l\_\_enumext\_labelsep\_v\_dim ..... 3100  
 \l\_\_enumext\_labelsep\_vii\_dim . 3269, 3278, 3320, 3515, 3575, 3591  
 \l\_\_enumext\_labelsep\_viii\_dim 3627, 3636, 3678, 3919, 3942  
 \l\_\_enumext\_labelwidth\_i\_dim . 2540, 2575, 3865, 3880  
 \l\_\_enumext\_labelwidth\_v\_dim ..... 3100  
 \l\_\_enumext\_labelwidth\_vii\_dim ... 3269, 3277, 3320, 3568, 3572, 3590  
 \l\_\_enumext\_labelwidth\_viii\_dim .. 3627, 3635, 3678, 3912, 3916, 3941  
 \l\_\_enumext\_leftmargin\_tmp\_v\_bool . 91, 3224  
 \l\_\_enumext\_leftmargin\_tmp\_X\_bool ..... 59  
 \l\_\_enumext\_leftmargin\_tmp\_X\_dim ..... 59  
 \l\_\_enumext\_leftmargin\_X\_dim ..... 59  
 \\_\_enumext\_level: 187, 187, 538, 541, 542, 551, 553, 792, 796, 800, 867, 871, 875, 879, 963, 965, 967, 969, 1002, 1004, 1006, 1008, 1012, 1052, 1055, 1074, 1083, 1089, 1094, 1098, 1109, 1113, 1114, 1119, 1155, 1159, 1332, 1338, 1385, 1387, 1389, 1392, 1399, 1401, 1403, 1406, 2012, 2014, 2016, 2044, 2045, 2047, 2103, 2111, 2115, 2119, 2361, 2364, 2365, 2635, 2636, 2640, 2641, 2642, 2649, 2651, 2655, 2656, 2659, 2665, 2666, 2721, 2724, 2726, 2733, 2734, 2735, 2738, 2741, 2871, 2873, 2920, 2933, 2940, 2951, 2953, 2956, 2957, 2959, 2964, 2971, 2974, 2976, 2978, 2979, 2980, 2981, 2984, 2990, 2995, 3001, 3004, 3006, 3012  
 \l\_\_enumext\_level\_h\_int .. 20, 219, 241, 254, 577, 614, 1820, 1836, 2317, 2334, 3405, 3406  
 \l\_\_enumext\_level\_int . 83, 20, 189, 228, 240, 255, 975, 1126, 1319, 1814, 1842, 1922, 2294, 2304, 2310, 2316, 2324, 2332, 2339, 2825, 2886, 2887, 2897, 2904, 2918, 2931, 2986, 3064, 3151, 3443, 3453, 3768  
 \\_\_enumext\_list\_arg\_two\_i: ..... 2791  
 \\_\_enumext\_list\_arg\_two\_ii: ..... 2791  
 \\_\_enumext\_list\_arg\_two\_iii: ..... 2791  
 \\_\_enumext\_list\_arg\_two\_iv: ..... 2791  
 \\_\_enumext\_list\_arg\_two\_v: . 79, 2791, 3039, 3225  
 \\_\_enumext\_list\_arg\_two\_vii: ..... 2831, 3387  
 \\_\_enumext\_list\_arg\_two\_viii: .... 2831, 3743  
 \l\_\_enumext\_listoffset\_v\_dim ..... 3102  
 \l\_\_enumext\_listparindent\_vii\_dim .... 3578  
 \l\_\_enumext\_listparindent\_viii\_dim ... 3922  
 \\_\_enumext\_log\_answer\_vars: . 30, 317, 325, 1929  
 \\_\_enumext\_log\_global\_vars: . 30, 317, 317, 1928  
 \\_\_enumext\_make\_label: 34, 77, 78, 80, 2729, 2729, 2820  
 \l\_\_enumext\_mark\_answer\_sym\_tl . 68, 121, 1969, 2173, 2376, 2547, 2560, 3869  
 \l\_\_enumext\_mark\_position\_str 121, 1973, 1974, 1997, 1998, 2171  
 \l\_\_enumext\_mark\_ref\_sym\_tl .. 121, 1983, 2274, 2499  
 \\_\_enumext\_mini\_addvspace: .. 48, 85, 1148, 1148, 2961  
 \\_\_enumext\_mini\_addvspace\_vii: 50, 1301, 1301, 3345  
 \\_\_enumext\_mini\_addvspace\_viii: 50, 1301, 1307, 3703  
 \\_\_enumext\_mini\_env\* ..... 1042  
 \\_\_enumext\_mini\_right\_cmd:n . 51, 52, 1327, 1329,



1329  
 \\_\_enumext\_mini\_set\_vskip: . 46, 1049, 1049, 1150  
 \\_\_enumext\_mini\_set\_vskip\_vii: 49, 1244, 1244,  
 1303  
 \\_\_enumext\_mini\_set\_vskip\_viii: 49, 1244, 1266,  
 1309  
 \\_\_enumext\_minipage:w . . 31, 338, 340, 1044, 3256,  
 3577, 3921  
 \l\_\_enumext\_minipage\_active\_v\_bool . . 88, 89,  
 3083, 3108, 3121, 3129  
 \g\_\_enumext\_minipage\_active\_vii\_bool . . . 94,  
 3356, 3361, 3373  
 \l\_\_enumext\_minipage\_active\_vii\_bool . 3341,  
 3352  
 \g\_\_enumext\_minipage\_active\_viii\_bool 3714,  
 3719, 3731  
 \l\_\_enumext\_minipage\_active\_viii\_bool 3699,  
 3710  
 \g\_\_enumext\_minipage\_active\_X\_bool . . . 154  
 \l\_\_enumext\_minipage\_active\_X\_bool . . . . 73  
 \g\_\_enumext\_minipage\_after\_skip 73, 1248, 1260,  
 3371, 3729  
 \l\_\_enumext\_minipage\_after\_skip 46, 47, 86, 89,  
 73, 1065, 1080, 1100, 1116, 1131, 1137, 1143, 1157,  
 1167, 1176, 1179, 1191, 1209, 1220, 1236, 1268, 1281,  
 1295, 3021, 3138  
 \g\_\_enumext\_minipage\_center\_vii\_bool . 3365,  
 3374  
 \g\_\_enumext\_minipage\_center\_viii\_bool 3723,  
 3732  
 \g\_\_enumext\_minipage\_center\_X\_bool . . . 154  
 \l\_\_enumext\_minipage\_hsep\_v\_dim . . . 88, 3081  
 \l\_\_enumext\_minipage\_hsep\_vii\_dim . . . . 3339  
 \l\_\_enumext\_minipage\_hsep\_viii\_dim . . . 3697  
 \l\_\_enumext\_minipage\_left\_skip 46, 88, 73, 1057,  
 1072, 1091, 1106, 1153, 1163, 1168, 1174, 1183, 1200,  
 1212, 1232, 1242, 1246, 1251, 1255, 1269, 1273, 1287,  
 1305, 1311  
 \l\_\_enumext\_minipage\_left\_v\_dim 88, 3079, 3087  
 \l\_\_enumext\_minipage\_left\_vii\_dim 3335, 3347  
 \l\_\_enumext\_minipage\_left\_viii\_dim 3693, 3705  
 \l\_\_enumext\_minipage\_left\_X\_dim . . . . . 73  
 \g\_\_enumext\_minipage\_right\_skip 73, 1247, 1252,  
 1256, 3364, 3722  
 \l\_\_enumext\_minipage\_right\_skip . 46, 73, 1061,  
 1076, 1096, 1111, 1169, 1175, 1187, 1205, 1216, 1270,  
 1277, 1291, 1339, 1356  
 \l\_\_enumext\_minipage\_right\_v\_dim . . 88, 1350,  
 1355, 3077, 3081  
 \g\_\_enumext\_minipage\_right\_vii\_dim 94, 3343,  
 3363, 3376  
 \l\_\_enumext\_minipage\_right\_vii\_dim 94, 3333,  
 3338, 3344  
 \g\_\_enumext\_minipage\_right\_viii\_dim . . 3701,  
 3721, 3734  
 \l\_\_enumext\_minipage\_right\_viii\_dim . . 3691,  
 3696, 3702  
 \g\_\_enumext\_minipage\_right\_X\_dim . . . . . 154  
 \g\_\_enumext\_minipage\_right\_X\_skip . . . . 154  
 \g\_\_enumext\_minipage\_stat\_int . 85, 88, 73, 1344,  
 1361, 2960, 3014, 3019, 3084, 3131, 3136  
 \g\_\_enumext\_miniright\_code\_vii\_tl . 94, 3369,  
 3375  
 \g\_\_enumext\_miniright\_code\_viii\_tl 3727, 3733  
 \g\_\_enumext\_miniright\_code\_X\_tl . . . . . 154  
 \\_\_enumext\_multi\_addvspace: . . . 45, 86, 997, 997,  
 2992  
 \\_\_enumext\_multi\_set\_vskip: . . 44, 961, 961, 999  
 \l\_\_enumext\_multicols\_above\_ii\_skip . . . 980  
 \l\_\_enumext\_multicols\_above\_iii\_skip . . . 986  
 \l\_\_enumext\_multicols\_above\_iv\_skip . . . 992  
 \l\_\_enumext\_multicols\_above\_v\_skip 1016, 1030,  
 1040  
 \l\_\_enumext\_multicols\_above\_X\_skip . . . . 67  
 \l\_\_enumext\_multicols\_below\_v\_skip 1020, 1034,  
 3123  
 \l\_\_enumext\_multicols\_below\_X\_skip . . . . 67  
 \\_\_enumext\_multicols\_start: 85, 2966, 2968, 2968  
 \\_\_enumext\_multicols\_stop: 86, 1334, 2998, 2998,  
 3023  
 \\_\_enumext\_newlabel:nn 27, 32, 72, 381, 381, 2350,  
 2458  
 \l\_\_enumext\_newlabel\_arg\_one\_tl 27, 32, 72, 74,  
 143, 2273, 2343, 2351, 2447, 2459, 2497  
 \l\_\_enumext\_newlabel\_arg\_two\_tl 27, 32, 71, 143,  
 2297, 2307, 2321, 2337, 2352, 2434, 2439, 2444, 2460  
 \\_\_enumext\_parse\_keys:n . . . 55, 2867, 2892, 2892  
 \\_\_enumext\_parse\_keys\_vii:n 55, 3382, 3413, 3413  
 \\_\_enumext\_parse\_keys\_viii:n . 3739, 3773, 3773  
 \\_\_enumext\_parse\_save\_key:n 66, 2037, 2042, 2042  
 \\_\_enumext\_parse\_save\_key\_vii:n 66, 2032, 2042,  
 2050  
 \\_\_enumext\_parse\_serie:n . . . . . 95  
 \\_\_enumext\_parse\_series:n . . 55, 84, 1518, 1518,  
 2900, 3419  
 \\_\_enumext\_parse\_store\_keys:n . . . . . 84  
 \l\_\_enumext\_parsep\_i\_skip 978, 980, 1129, 1177  
 \l\_\_enumext\_parsep\_ii\_skip . . . . 984, 986, 1135  
 \l\_\_enumext\_parsep\_iii\_skip . . . 990, 992, 1141  
 \l\_\_enumext\_parsep\_vii\_skip . . . . . 3579  
 \l\_\_enumext\_parsep\_viii\_skip . . . . . 3923  
 \l\_\_enumext\_partopsep\_v\_skip . 1032, 1036, 1203,  
 1207, 1214, 1218, 1234, 1238  
 \l\_\_enumext\_partopsep\_viii\_skip . . . . . 1279  
 \\_\_enumext\_phantomsection: 32, 345, 374, 378, 394  
 \\_\_enumext\_print\_footnote: . . 2594, 2617, 3597,  
 3948  
 \\_\_enumext\_print\_keyans\_box:NN 68, 2165, 2165,  
 2178, 2363, 2539, 2574, 3865, 3880  
 \l\_\_enumext\_print\_keyans\_i\_tl . . . . 4006, 4028  
 \l\_\_enumext\_print\_keyans\_ii\_tl . . . 4010, 4029  
 \l\_\_enumext\_print\_keyans\_iii\_tl . . 4014, 4030  
 \l\_\_enumext\_print\_keyans\_iv\_tl . . . 4018, 4031  
 \l\_\_enumext\_print\_keyans\_starred\_tl 107, 108,  
 111, 4002, 4045  
 \l\_\_enumext\_print\_keyans\_vii\_tl 107, 4022, 4032  
 \l\_\_enumext\_print\_keyans\_X\_tl . . . . . 111  
 \\_\_enumext\_printkeyans:nnn 107, 108, 4033, 4036,  
 4036  
 \\_\_enumext\_redefine\_item: . 78, 2668, 2668, 2819  
 \l\_\_enumext\_ref\_key\_arg\_tl 35, 37, 202, 531, 532,  
 545, 576, 579, 590, 596, 607, 648, 649, 660  
 \l\_\_enumext\_ref\_the\_count\_tl . 35, 37, 538, 541,  
 544, 584, 586, 589, 601, 603, 606, 654, 656, 659  
 \\_\_enumext\_regex\_counter\_style: . . 28, 35, 197,  
 197, 539, 585, 602, 655  
 \\_\_enumext\_register\_counter\_style:Nn . . 413,  
 413, 418, 419, 420, 421, 422  
 \\_\_enumext\_remove\_extra\_parsep\_vii: . . 3397,

[3606](#), [3606](#)  
`\__enumext_remove_extra_parsep_viii:` . [3753](#),  
[3957](#), [3957](#)  
`\__enumext_renew_footnote:` ... [2594](#), [2598](#), [3549](#),  
[3902](#)  
`\l__enumext_renew_the_count_v_tl` [657](#), [666](#), [668](#)  
`\l__enumext_renew_the_count_vii_tl` [587](#), [616](#),  
[618](#)  
`\l__enumext_renew_the_count_viii_tl` [604](#), [623](#),  
[625](#)  
`\l__enumext_renew_the_count_X_tl` ..... [37](#)  
`\__enumext_reset_global_bool:` .. [293](#), [296](#), [305](#)  
`\__enumext_reset_global_int:` ... [293](#), [295](#), [299](#)  
`\__enumext_reset_global_tl:` .... [293](#), [297](#), [311](#)  
`\__enumext_reset_global_vars:` . [30](#), [63](#), [293](#), [293](#),  
[1936](#)  
`\l__enumext_resume_active_bool` [55](#), [57](#), [48](#), [1522](#),  
[1642](#)  
`\__enumext_resume_counter:` .. [57](#), [58](#), [1640](#), [1646](#),  
[1653](#)  
`\__enumext_resume_counter:n` . [55](#), [57](#), [1611](#), [1616](#),  
[1640](#), [1640](#), [1710](#), [1718](#)  
`\__enumext_resume_counter_save_ans:` .. [57](#), [58](#),  
[1640](#), [1651](#), [1683](#)  
`\__enumext_resume_counter_series:` [57](#), [58](#), [1640](#),  
[1649](#), [1666](#)  
`\g__enumext_resume_int` ... [48](#), [1563](#), [1657](#), [1658](#)  
`\__enumext_resume_last:n` .. [55](#), [1518](#), [1524](#), [1537](#)  
`\l__enumext_resume_name_tl` [48](#), [1559](#), [1567](#), [1570](#),  
[1586](#), [1594](#), [1597](#), [1643](#), [1644](#), [1672](#), [1679](#)  
`\__enumext_resume_save_counter:` [56](#), [1550](#), [1550](#),  
[3028](#), [3437](#)  
`\__enumext_resume_series:n` . [57](#), [1482](#), [1607](#), [1607](#)  
`\__enumext_resume_starred:` . [59](#), [1483](#), [1704](#), [1704](#)  
`\g__enumext_resume_vii_int` .. [96](#), [48](#), [1590](#), [1662](#),  
[1663](#)  
`\__enumext_safe_exec:` ..... [83](#), [2866](#), [2883](#), [2883](#)  
`\__enumext_safe_exec_vii:` ... [3381](#), [3402](#), [3402](#)  
`\__enumext_safe_exec_viii:` ... [3738](#), [3758](#), [3758](#)  
`\l__enumext_series_name_tl` ..... [57](#)  
`\l__enumext_series_str` . [56](#), [84](#), [1480](#), [1520](#), [1528](#),  
[1529](#), [1531](#), [1533](#), [1554](#), [1557](#), [1561](#), [1581](#), [1584](#), [1588](#),  
[2896](#), [3417](#)  
`\__enumext_set_error:nn` ..... [4116](#), [4126](#), [4128](#)  
`\__enumext_set_parse:n` ..... [4099](#), [4116](#), [4116](#)  
`\l__enumext_setkey_tmpa_int` ... [106](#), [4092](#), [4096](#)  
`\l__enumext_setkey_tmpa_seq` .. [106](#), [4090](#), [4100](#),  
[4106](#), [4108](#), [4110](#), [4123](#)  
`\l__enumext_setkey_tmpa_tl` .... [106](#), [4098](#), [4102](#)  
`\l__enumext_setkey_tmppb_seq` .. [106](#), [4091](#), [4094](#),  
[4098](#), [4099](#)  
`\l__enumext_setkey_tmppb_tl` [106](#), [4118](#), [4120](#), [4121](#)  
`\l__enumext_show_answer_bool` . [121](#), [1977](#), [2001](#),  
[2370](#), [2514](#), [2528](#), [3175](#), [3863](#)  
`\__enumext_show_length:nnn` .. [41](#), [205](#), [205](#), [4192](#),  
[4193](#), [4194](#), [4195](#), [4196](#), [4197](#), [4198](#), [4199](#), [4200](#), [4201](#),  
[4207](#), [4208](#), [4209](#), [4210](#), [4211](#), [4212](#), [4213](#), [4214](#), [4215](#),  
[4216](#)  
`\l__enumext_show_position_bool` [121](#), [1980](#), [2004](#),  
[2374](#), [2518](#), [2529](#), [3176](#), [3867](#)  
`\g__enumext_standar_bool` [28](#), [83](#), [20](#), [218](#), [221](#), [239](#),  
[308](#), [1552](#), [1617](#), [1629](#), [1655](#), [1668](#), [1706](#), [1841](#), [1854](#)  
`\l__enumext_standar_bool` . [83](#), [86](#), [20](#), [2302](#), [2315](#),  
[2331](#), [2889](#), [3027](#)  
`\l__enumext_standar_first_bool` [29](#), [83](#), [20](#), [244](#),  
[1539](#), [1686](#), [1748](#), [1755](#)  
`\__enumext_standar_item_vii:w` . [97](#), [3477](#), [3479](#),  
[3479](#)  
`\__enumext_standar_item_viii:w` [103](#), [3810](#), [3812](#),  
[3812](#)  
`\__enumext_standar_ref:` .... [35](#), [529](#), [549](#), [2821](#)  
`\__enumext_standar_ref:n` .... [35](#), [521](#), [529](#), [529](#)  
`\g__enumext_standar_series_tl` . [48](#), [1541](#), [1542](#),  
[1708](#), [1711](#)  
`\g__enumext_starred_bool` [28](#), [95](#), [96](#), [20](#), [227](#), [230](#),  
[253](#), [309](#), [1579](#), [1622](#), [1633](#), [1660](#), [1675](#), [1714](#), [1819](#),  
[1860](#), [2293](#), [2303](#), [2333](#), [2428](#), [2915](#), [2928](#), [3377](#)  
`\l__enumext_starred_bool` . [95](#), [96](#), [20](#), [2226](#), [2234](#),  
[2318](#), [2359](#), [3410](#), [3436](#)  
`\__enumext_starred_columns_set_vii:` .. [3263](#),  
[3263](#), [3390](#)  
`\__enumext_starred_columns_set_viii:` . [3621](#),  
[3621](#), [3746](#)  
`\l__enumext_starred_first_bool` ... [29](#), [20](#), [258](#),  
[1544](#), [1695](#), [1748](#), [1755](#)  
`\__enumext_starred_item:nn` ... [2645](#), [2645](#), [2674](#)  
`\__enumext_starred_item_exec:` [104](#), [3855](#), [3855](#),  
[3906](#)  
`\__enumext_starred_item_vii:w` . [97](#), [3476](#), [3495](#),  
[3495](#)  
`\__enumext_starred_item_vii_aux_i:w` .. [3495](#),  
[3500](#), [3503](#)  
`\__enumext_starred_item_vii_aux_ii:w` . [3495](#),  
[3501](#), [3506](#), [3508](#)  
`\__enumext_starred_item_vii_aux_iii:w` [3495](#),  
[3511](#), [3520](#)  
`\__enumext_starred_item_viii:w` [103](#), [104](#), [3809](#),  
[3828](#), [3828](#)  
`\__enumext_starred_item_viii_aux_i:w` .. [104](#),  
[3828](#), [3833](#), [3836](#)  
`\__enumext_starred_item_viii_aux_ii:w` . [104](#),  
[3828](#), [3834](#), [3848](#), [3850](#)  
`\__enumext_starred_joined_item_vii:n` . [93](#), [97](#),  
[3282](#), [3282](#), [3474](#)  
`\__enumext_starred_joined_item_viii:n` . [100](#),  
[103](#), [3640](#), [3640](#), [3807](#)  
`\__enumext_starred_ref:` .... [37](#), [574](#), [612](#), [2849](#)  
`\__enumext_starred_ref:n` .... [36](#), [568](#), [574](#), [574](#)  
`\g__enumext_starred_series_tl` . [48](#), [1546](#), [1547](#),  
[1716](#), [1719](#)  
`\__enumext_start_from:NNn` [38](#), [671](#), [671](#), [684](#), [706](#)  
`\l__enumext_start_i_int` ..... [1658](#), [1670](#), [1689](#)  
`\__enumext_start_item_tmp_vii:` [94](#), [3393](#), [3459](#),  
[3459](#)  
`\__enumext_start_item_tmp_viii:` .. [101](#), [3749](#),  
[3792](#), [3792](#)  
`\__enumext_start_item_vii:w` . [97](#), [98](#), [3487](#), [3492](#),  
[3517](#), [3524](#), [3526](#), [3526](#)  
`\__enumext_start_item_viii:w` .. [103](#), [3820](#), [3825](#),  
[3853](#), [3883](#), [3883](#)  
`\g__enumext_start_line_tl` [29](#), [131](#), [246](#), [260](#), [314](#),  
[1884](#), [1889](#), [1894](#), [1908](#), [1913](#), [1918](#)  
`\__enumext_start_list:nn` [31](#), [80](#), [91](#), [332](#), [334](#), [2870](#),  
[3036](#), [3189](#), [3385](#), [3741](#)  
`\__enumext_start_mini_vii:` . [95](#), [3331](#), [3331](#), [3428](#)  
`\__enumext_start_mini_viii:` ... [102](#), [3689](#), [3689](#),  
[3784](#)  
`\__enumext_start_save_ans_msg:` [59](#), [1732](#), [1732](#),

1757  
 \\_\_enumext\_start\_store\_level: . 84, 2869, 2909, 2909  
 \\_\_enumext\_start\_store\_level\_vii: . 96, 3384, 3439, 3439  
 \l\_\_enumext\_start\_vii\_int . . . 1663, 1677, 1698  
 \l\_\_enumext\_start\_X\_int . . . . . 85, 701  
 \\_\_enumext\_stop\_item\_tmp\_vii: . 94, 96, 98, 3392, 3396, 3461, 3528  
 \\_\_enumext\_stop\_item\_tmp\_viii: 101, 103, 3748, 3752, 3794, 3885  
 \\_\_enumext\_stop\_item\_vii: 98, 99, 3528, 3582, 3582  
 \\_\_enumext\_stop\_item\_viii: 105, 3885, 3933, 3933  
 \\_\_enumext\_stop\_list: . . 31, 332, 335, 2879, 3046, 3202, 3398, 3755  
 \\_\_enumext\_stop\_mini\_vii: 94, 96, 3350, 3350, 3432  
 \\_\_enumext\_stop\_mini\_viii: 102, 3689, 3708, 3788  
 \\_\_enumext\_stop\_save\_ans\_msg: . 59, 1732, 1737, 1926  
 \\_\_enumext\_stop\_store\_level: . . 84, 2880, 2909, 2938  
 \\_\_enumext\_stop\_store\_level\_vii: . . 96, 3399, 3439, 3449  
 \l\_\_enumext\_store\_active\_bool 25, 60, 84, 95, 97, 1687, 1696, 1764, 2191, 2913, 2926, 3051, 3059, 3147, 3206, 3441, 3451, 3767  
 \\_\_enumext\_store\_active\_keys:n . . 65, 66, 2010, 2010, 2906  
 \\_\_enumext\_store\_active\_keys\_vii:n 65, 66, 95, 2010, 2020, 3420  
 \\_\_enumext\_store\_addto\_prop:n 67, 73, 2085, 2085, 2093, 2213, 2409, 3858  
 \\_\_enumext\_store\_addto\_seq:n 67, 75, 2094, 2094, 2098, 2105, 2119, 2127, 2136, 2154, 2162, 2277, 2502  
 \l\_\_enumext\_store\_anskey\_arg\_tl . . 25, 70, 97, 2219, 2228, 2230, 2236, 2244, 2247, 2257, 2262, 2265, 2271, 2277  
 \\_\_enumext\_store\_anskey\_code:nnnn . 69, 2208, 2211, 2211  
 \\_\_enumext\_store\_anskey\_show\_left:n 72, 2218, 2368, 2368  
 \\_\_enumext\_store\_anskey\_show\_wrap:n 72, 2356, 2356, 2372, 2387  
 \l\_\_enumext\_store\_columns\_break\_bool . 2185, 2225  
 \l\_\_enumext\_store\_columns\_join\_int 97, 2233, 2238  
 \\_\_enumext\_store\_internal\_ref: . . 69, 71, 2216, 2279, 2279  
 \l\_\_enumext\_store\_item\_symbol\_sep\_dim 2183, 2254, 2259  
 \l\_\_enumext\_store\_item\_symbol\_tl . 2181, 2245, 2249  
 \l\_\_enumext\_store\_keyans\_item\_opt\_sep\_tl . . . . 1966, 2403, 2405, 2476, 2480, 3841, 3843  
 \l\_\_enumext\_store\_keyans\_item\_opt\_tl . . . 97  
 \l\_\_enumext\_store\_keyans\_label\_tl 25, 73-75, 104, 97, 2392, 2395, 2398, 2405, 2407, 2409, 2466, 2469, 2472, 2478, 2483, 2493, 2502, 3838, 3843, 3844, 3857, 3858, 3860  
 \\_\_enumext\_store\_level\_close: . 67, 2099, 2123, 2942  
 \\_\_enumext\_store\_level\_close\_vii: 2130, 2158, 3455  
 \\_\_enumext\_store\_level\_open: . . 67, 2099, 2099, 2921, 2934  
 \\_\_enumext\_store\_level\_open\_vii: . 2130, 2130, 3445  
 \g\_\_enumext\_store\_name\_tl . 25, 86, 97, 313, 320, 321, 322, 323, 1740, 1766, 1883, 1888, 1893, 1907, 1912, 1917, 1924  
 \l\_\_enumext\_store\_name\_tl . 25, 59, 61, 97, 1573, 1576, 1600, 1603, 1691, 1700, 1735, 1744, 1745, 1766, 1767, 1769, 1770, 1772, 1774, 1775, 1777, 1779, 1780, 1804, 2087, 2089, 2096, 2345, 2346, 2382, 2449, 2450, 2553, 2566, 3875  
 \l\_\_enumext\_store\_ref\_key\_bool 69, 1986, 2214, 2268, 2413, 2490  
 \l\_\_enumext\_store\_save\_key\_vii\_bool . . 2022, 2052  
 \l\_\_enumext\_store\_save\_key\_vii\_tl 2024, 2025, 2053, 2054, 2134, 2144, 2150, 2154  
 \l\_\_enumext\_store\_save\_key\_X\_bool . . . . . 65  
 \l\_\_enumext\_store\_save\_key\_X\_tl . . 65, 66, 111  
 \l\_\_enumext\_store\_upper\_level\_X\_bool . . 111  
 \l\_\_enumext\_store\_write\_aux\_file\_tl 27, 72, 74, 143, 2348, 2354, 2456, 2462  
 \\_\_enumext\_storing\_exec: 59, 60, 1742, 1758, 1762  
 \\_\_enumext\_storing\_set:n . . 59, 1727, 1742, 1742  
 \l\_\_enumext\_the\_counter\_v\_tl . . . . . 656  
 \l\_\_enumext\_the\_counter\_vii\_tl . . . . . 586  
 \l\_\_enumext\_the\_counter\_viii\_tl . . . . . 603  
 \l\_\_enumext\_the\_counter\_X\_tl . . . . . 37  
 \\_\_enumext\_tmp:n 32, 36, 41, 47, 59, 66, 67, 72, 79, 84, 85, 96, 112, 120, 146, 150, 154, 173, 784, 788, 1476, 1487, 1723, 1731, 1783, 1801, 1956, 1991, 1992, 2009, 2028, 2041, 2281, 2288, 2289, 2310, 2324, 2327, 2339, 2415, 2422, 2791, 2830, 2831, 2863  
 \\_\_enumext\_tmp:nn 445, 466, 467, 495, 496, 508, 701, 720, 765, 783, 841, 849, 850, 864, 929, 945, 946, 960, 1365, 1381, 2578, 2593  
 \\_\_enumext\_tmp:nnn 509, 525, 526, 527, 528, 556, 572, 573  
 \\_\_enumext\_tmp:nnnnnn 721, 746, 749, 752, 754, 756, 759, 762  
 \\_\_enumext\_tmp:w . . . . . 3982, 3985  
 \l\_\_enumext\_tmpa\_vii\_int . . . . . 3273, 3276  
 \l\_\_enumext\_tmpa\_viii\_int . . . . . 3631, 3634  
 \l\_\_enumext\_tmpa\_X\_int . . . . . 154  
 \l\_\_enumext\_topsep\_v\_skip 1018, 1022, 1172, 1185, 1193, 1198, 1218, 1222, 3205, 3237  
 \l\_\_enumext\_topsep\_vii\_skip . . 1249, 1258, 1262  
 \l\_\_enumext\_topsep\_viii\_skip . 1271, 1293, 1297  
 \l\_\_enumext\_vspace\_a\_star\_v\_bool . . . . 1414  
 \l\_\_enumext\_vspace\_a\_star\_vii\_bool . . . 1436  
 \l\_\_enumext\_vspace\_a\_star\_viii\_bool . . . 1447  
 \l\_\_enumext\_vspace\_a\_star\_X\_bool . . . . . 85  
 \\_\_enumext\_vspace\_above: . . 52, 1382, 1382, 2947  
 \\_\_enumext\_vspace\_above\_v: . 53, 1410, 1410, 3075  
 \l\_\_enumext\_vspace\_above\_v\_skip . . 1412, 1416, 1418  
 \\_\_enumext\_vspace\_above\_vii: . . 53, 1432, 1432, 3425  
 \l\_\_enumext\_vspace\_above\_vii\_skip 1434, 1438, 1440  
 \\_\_enumext\_vspace\_above\_viii: . 53, 1432, 1443, 3782  
 \l\_\_enumext\_vspace\_above\_viii\_skip 1445, 1449, 1451



`\l_enumext_vspace_b_star_v_bool` . . . . . 1425  
`\l_enumext_vspace_b_star_vii_bool` . . . 1458  
`\l_enumext_vspace_b_star_viii_bool` . . . 1469  
`\l_enumext_vspace_b_star_X_bool` . . . . . 85  
`\_enumext_vspace_below:` . . 53, 1396, 1396, 3026  
`\_enumext_vspace_below_v:` . 53, 1421, 1421, 3143  
`\l_enumext_vspace_below_v_skip` . . 1423, 1427, 1429  
`\_enumext_vspace_below_vii:` . . 54, 1454, 1454, 3435  
`\l_enumext_vspace_below_vii_skip` 1456, 1460, 1462  
`\_enumext_vspace_below_viii:` . 54, 1454, 1465, 3790  
`\l_enumext_vspace_below_viii_skip` 1467, 1471, 1473  
`\_enumext_widest_from:nNn` . . 38, 685, 685, 700, 712  
`\g_enumext_widest_label_tl` 24, 33, 55, 433, 437, 441  
`\l_enumext_wrap_label_opt_v_bool` . . . . . 2687  
`\l_enumext_wrap_label_opt_vii_bool` 97, 3486  
`\l_enumext_wrap_label_opt_viii_bool` . . 103, 3819  
`\l_enumext_wrap_label_opt_X_bool` . . . . . 85  
`\l_enumext_wrap_label_v_bool` 2683, 2687, 2695, 2751  
`\l_enumext_wrap_label_vii_bool` 97, 3485, 3490, 3498, 3566  
`\l_enumext_wrap_label_viii_bool` . 103, 3818, 3823, 3831, 3910  
`\l_enumext_wrap_label_X_bool` . . . . . 85  
`\_enumext_wrapper_label_v:n` . . . . . 2753, 3184  
`\_enumext_wrapper_label_vii:n` . . . . . 3569  
`\_enumext_wrapper_label_viii:n` . . . . . 3913  
`\_enumext_zero_parsep:` . . . 47, 1069, 1124, 1124  
`enumext*` . . . . . 5, 3379  
`enumXi` . . . . . 405  
`enumXii` . . . . . 405  
`enumXiii` . . . . . 405  
`enumXiv` . . . . . 405  
`enumXv` . . . . . 405  
`enumXvi` . . . . . 405  
`enumXvii` . . . . . 405  
`enumXviii` . . . . . 405  
**Environments provide by enumext:**  
`enumext*` 23, 24, 26–28, 33, 36, 37, 40–43, 49, 50, 53–57, 59, 61–71, 74, 76, 77, 82–84, 95, 96, 98, 99, 101, 103, 105, 107, 108, 111, 113  
`enumext` 23, 24, 26, 28, 33–39, 41–49, 51–57, 59, 61–71, 74, 76–78, 80, 81, 83, 84, 86, 87, 91, 92, 94, 96, 107, 108, 110, 112  
`keyans*` 23–29, 33, 36, 37, 40–43, 49, 50, 53, 54, 60, 61, 64, 65, 67, 73, 77, 82, 102, 111, 113  
`keyanspic` 23–26, 29, 33, 34, 37, 51, 60, 61, 64, 67, 73–75, 89–92, 112  
`keyans` . . 23–26, 28, 29, 33, 34, 37–39, 41–43, 45, 48, 49, 51–53, 60, 61, 64, 65, 67, 73–75, 79–81, 87, 89–91, 94, 103, 111, 112  
**Environments:**  
`list` . . . . . 27, 31, 80, 81, 83  
`lrbox` . . . . . 92, 98, 99, 105  
`minipage` . . . . . 27, 31, 43, 45, 46, 89–92, 98, 99, 105  
`multicols` . . . . . 44–46, 51, 85, 86, 88, 89

exp commands:

`\exp_after:wN` . . . . . 3985  
`\exp_args:Ne` . . . . . 2903, 3973  
`\exp_not:N` . 45, 436, 544, 589, 606, 659, 798, 812, 813, 824, 825, 836, 837, 2273, 2379, 2380, 2495, 2550, 2551, 2563, 2564, 3872, 3873, 3982  
`\exp_not:n` 248, 262, 274, 281, 288, 544, 545, 589, 590, 606, 607, 659, 660, 799, 1504, 1516, 2071, 2083, 2238, 2249, 2259, 2273, 2274, 2351, 2459, 2497, 2499

## F

`\fbox` . . . . . 1961

file commands:

`\file_input_stop:` . . . . . 4344  
`first` . . . . . 850  
`font` . . . . . 445  
`\footnote` . . . . . 77  
`\footnote` . . . . . 77, 2602  
`\footnotemark` . . . . . 2612  
`\footnotesize` . . . . . 2380, 2551, 2564, 3873  
`\footnotetext` . . . . . 2596

## G

`\getkeyans` . . . . . 14, 106, 3971

group commands:

`\group_begin:` . . 2203, 2378, 2549, 2562, 3545, 3564, 3871, 3898, 3908, 3993, 4027  
`\group_end:` 2209, 2385, 2556, 2569, 3574, 3586, 3878, 3918, 3937, 3995, 4034

## H

`\hbadness` . . . . . 3593, 3944

hbox commands:

`\hbox_set:Nn` . . . . . 425

`\hfill` 475, 479, 484, 485, 1336, 1354, 2273, 2495, 3355, 3713

hook commands:

`\hook_gput_code:nnn` . . . . . 9, 181, 185, 343

`\hook_gset_rule:nnnn` . . . . . 344

`\hspace` . . . . . 3604, 3955

`\hyperlink` . . . . . 70, 75

`\hyperlink` . . . . . 2273, 2495

`\hypertarget` . . . . . 32

`\hypertarget` . . . . . 373

## I

`\IfHyperBoolean` . . . . . 351

`\IfPackageLoadedTF` . . . . . 11, 347, 361

`\ignorespaces` . . . . . 801

`\inputlineno` . . . . . 248, 262, 274, 281, 288

int commands:

`\int_add:Nn` . . . . . 3315, 3673

`\int_case:nn` . . . . 975, 1126, 1814, 1836, 1874, 1898

`\int_compare:nNnTF` . 577, 594, 614, 621, 1051, 1170, 1315, 1319, 1323, 1922, 1941, 1947, 2195, 2199, 2393, 2432, 2437, 2442, 2467, 2545, 2887, 2897, 2918, 2931, 2970, 2986, 3000, 3014, 3060, 3064, 3093, 3118, 3131, 3151, 3155, 3211, 3285, 3295, 3311, 3406, 3443, 3453, 3599, 3608, 3643, 3653, 3669, 3761, 3768, 3950, 3959, 4096

`\int_compare_p:nNn` . . . 219, 228, 240, 241, 254, 255, 1820, 1842, 2294, 2304, 2316, 2317, 2332, 2334

`\int_decr:N` . . . . . 3314, 3672

`\int_eval:n` . 330, 2089, 2346, 2380, 2450, 2551, 2564, 2806, 2848, 3303, 3661, 3873

`\int_from_alph:n` . . . . . 679, 693

`\int_from_roman:n` . . . . . 681, 695

`\int_gadd:Nn` ..... 3316, 3674  
`\int_gdecr:N` ..... 1823, 1827, 1830, 1833, 1845  
`\int_gincr:N` 1657, 1662, 2206, 2505, 2633, 2663, 2701, 2960, 3084, 3173, 3463, 3541, 3796, 3862  
`\int_gset:Nn` ..... 1867, 2610  
`\int_gset_eq:NN` 1556, 1563, 1569, 1575, 1583, 1590, 1596, 1602, 2607  
`\int_gzero:N` . 301, 302, 303, 1344, 1361, 1953, 3019, 3136, 3617, 3968  
`\int_if_exist:NTF` 1531, 1567, 1573, 1594, 1600, 1777  
`\int_incr:N` 2886, 3055, 3210, 3405, 3462, 3760, 3795  
`\int_mod:nn` ..... 3610, 3961  
`\int_new:N` . 20, 21, 22, 23, 24, 48, 49, 73, 89, 101, 108, 128, 129, 137, 138, 139, 140, 151, 157, 158, 159, 160, 161, 1533, 1780  
`\int_set:Nn` 675, 679, 681, 1670, 1677, 1689, 1698, 2233, 3250, 3251, 3273, 3284, 3290, 3306, 3593, 3631, 3642, 3648, 3664, 3944, 4092  
`\int_set_eq:NN` ..... 1658, 1663, 3313, 3671  
`\int_sign:n` ..... 1869  
`\int_step_function:nnN` ..... 2310, 2324, 2339  
`\int_step_inline:nnn` ..... 3252  
`\int_to_roman:n` ..... 189, 2290, 2328  
`\int_use:N` 323, 328, 329, 1052, 1672, 1679, 1691, 1700, 2806, 2825, 2848, 2904, 2971, 2980, 2995, 3001, 3288, 3289, 3301, 3646, 3647, 3659  
`\int_zero:N` ..... 3602, 3953  
`\c_one_int` . 3273, 3292, 3298, 3304, 3308, 3311, 3631, 3650, 3656, 3662, 3666, 3669  
`\c_zero_int` 2294, 2304, 2316, 2317, 2332, 2334, 3443, 3453, 3613, 3964  
`\item` ..... 31, 42, 43, 68, 77, 89, 90, 92, 94, 101  
`\item` 77, 79, 96, 98, 103, 105, 336, 2107, 2113, 2138, 2146, 2230, 2469, 2472, 2670, 2705, 3391, 3393, 3747, 3749, 3860  
`\item*` ..... 5, 13, 64, 2703  
`item-pos*` ..... 2578  
`item-sym*` ..... 2578  
`\itemindent` ..... 24, 81  
`\itemindent` ..... 81  
`itemindent` ..... 765  
`\itemsep` ..... 90, 91  
`\itemsep` ..... 3226, 3232  
`\itemwidth` ..... 3280, 3324, 3328, 3638, 3682, 3686

K

`keyans` ..... 12, 3031  
`keyans*` ..... 12, 3736  
`keyanspic` ..... 13, 3186  
Keys for environments provide by [enumext](#):  
`above*` ..... 25, 52, 53  
`above` ..... 25, 52, 53, 85, 87, 95, 102  
`after` ..... 41-43, 86, 89, 96, 102  
`align` ..... 25, 34, 80, 98  
`before*` ..... 41, 42, 85, 95, 102  
`before` ..... 41, 42, 87  
`below*` ..... 25, 52-54  
`below` ..... 25, 52-54, 86, 89, 96, 102  
`check-ans` 25, 26, 28, 29, 59-64, 67, 69, 75, 77, 78, 85, 86, 99, 111  
`columns-sep` ..... 43, 85, 88  
`columns` ..... 25, 43, 46, 52, 85, 88  
`first` ..... 41-43, 98  
`font` ..... 34, 80, 98  
`item-pos*` ..... 69, 70, 76  
`item-sym*` ..... 24, 69, 70, 76, 78

`item*-sep` ..... 78  
`itemindent` ..... 25, 39, 40, 79, 98  
`itemsep` ..... 39, 82  
`keys` ..... 95  
`labelsep` ..... 34, 78, 81, 98  
`labelwidth` ..... 33-38, 81  
`label` ..... 24, 33, 35, 38, 92  
`lisparindent` ..... 82  
`list-indent` ..... 24, 39, 40, 91  
`list-offset` ..... 39, 40  
`listparindent` ..... 39, 98  
`mark-ans` ..... 26, 64, 67, 72  
`mark-pos` ..... 64, 65  
`mark-ref` ..... 26, 64, 67, 71  
`mini-env` 25, 43, 46, 51, 52, 67, 77, 85, 88, 94, 95, 101, 102  
`mini-right*` ..... 25, 43, 67, 94  
`mini-right` ..... 25, 43, 50, 67, 94  
`mini-sep` ..... 25, 43, 67, 85, 88  
`minirigth*` ..... 28  
`minirigth` ..... 28  
`no-store` ..... 26, 59-61, 66  
`noitemsep` ..... 39, 47  
`nosep` ..... 39, 47  
`parindent` ..... 82  
`parsep` ..... 39, 82, 98  
`partopsep` ..... 39  
`ref` ..... 24, 28, 35-37, 111  
`resume*` ..... 24, 54, 55, 59, 60, 66, 86  
`resume` ..... 24, 30, 54-60, 66, 67, 86, 96  
`rightmargin` ..... 39  
`save-ans` 25, 30, 55-59, 61-63, 65-67, 69, 73, 74, 79, 84, 87, 90, 96, 103, 104, 106, 107, 111  
`save-key` ..... 26, 55, 66, 84  
`save-pos` ..... 67  
`save-ref` ..... 27, 32, 64, 67, 69-71, 73, 75, 79, 104  
`save-sep` ..... 64, 67, 104  
`series` ..... 24, 54-59, 67, 84, 86  
`show-ans` ..... 26, 64, 65, 67-69, 72, 79, 104  
`show-length` ..... 28, 41, 110, 111  
`show-pos` ..... 26, 64, 65, 68, 69, 72, 75, 79, 104  
`start` ..... 25, 28, 38, 55, 67  
`store-brk` ..... 69, 70  
`store-key` ..... 65  
`topsep` ..... 39  
`widest` ..... 24, 28, 38  
`wrap-ans` ..... 64, 67, 68, 72  
`wrap-label*` ..... 34, 77, 80, 97, 98, 103  
`wrap-label` ..... 34, 80, 97, 98, 103  
`wrap-opt` ..... 64, 67

keys commands:

`\keys_define:nn` 447, 469, 498, 511, 558, 629, 703, 723, 767, 786, 843, 852, 931, 948, 1367, 1478, 1725, 1785, 1958, 1994, 2030, 2035, 2179, 2580, 3998, 4061  
`\l_keys_key_str` ..... 4177  
`\keys_precompile:nnN` . 107, 3997, 4000, 4004, 4008, 4012, 4016, 4020  
`\keys_set:nn` . 461, 955, 1372, 1377, 1619, 1624, 1711, 1719, 2222, 2899, 2903, 3071, 3418, 3777, 4063, 4064, 4065, 4066, 4067, 4068, 4069, 4070, 4071, 4072, 4073, 4074, 4075, 4113

keyval commands:

`\keyval_parse:NNn` ..... 1492, 2060

L

`label` ..... 509, 556, 629

Labels provide by **enumext**:

<code>\Alph*</code>	33
<code>\Roman*</code>	33
<code>\alph*</code>	33
<code>\arabic*</code>	28, 33
<code>\roman*</code>	33
<code>\labelsep</code>	91
<code>\labelsep</code>	3227, 3230
<code>labelsep</code>	445
<code>\labelwidth</code>	33, 91
<code>\labelwidth</code>	3227, 3228
<code>labelwidth</code>	445
<code>\leftmargin</code>	24, 81
<code>\leftmargin</code>	81, 3227
legacy commands:	
<code>\legacy_if:nTF</code>	3529, 3532, 3886, 3889
<code>\legacy_if_gset_false:n</code>	1045
<code>\legacy_if_set_false:n</code>	3531, 3888
<code>\legacy_if_set_true:n</code>	3491, 3516, 3523, 3536, 3824, 3852, 3893
<code>\linewidth</code>	85, 88
<code>\linewidth</code>	2955, 3081, 3249, 3276, 3337, 3634, 3695
<code>\list</code>	31
<code>\list</code>	334
<code>list-indent</code>	765
<code>list-offset</code>	765
<code>\listparindent</code>	3229
<code>listparindent</code>	765
<code>\lrbx</code>	3546, 3899

M

<code>\makebox</code>	92
<code>\makebox</code>	2169, 2171, 2725, 3560, 3568, 3572, 3912, 3916
<code>\makelabel</code>	77, 80, 92
<code>\makelabel</code>	80, 2731, 2747
<code>\makesavenoteenv</code>	367
<code>mark-ans</code>	1956
<code>mark-pos</code>	1956, 1992
<code>mark-ref</code>	1956
<code>mini-env</code>	929
<code>mini-sep</code>	929
<code>\minipage</code>	31
<code>\minipage</code>	340
<code>\miniright</code>	10, 51, 1313, 3017, 3134
<code>\miniright*</code>	10

mode commands:

<code>\mode_if_vertical:TF</code>	1000, 1028, 1151, 1230
<code>\mode_leave_vertical:</code>	798, 812, 824, 836, 2138, 2146, 2167, 2723, 3558

msg commands:

<code>\msg_error:nn</code>	3062, 3066, 3153, 3213, 3408, 3763, 3770, 4076
<code>\msg_error:nnn</code>	534, 581, 598, 651, 1317, 1321, 1346, 1363, 1631, 1635, 1750, 3987, 3992, 4058, 4129
<code>\msg_error:nnnn</code>	2193, 2197, 2201, 3053, 3149, 3157
<code>\msg_fatal:nn</code>	2888
<code>\msg_fatal:nnn</code>	399
<code>\msg_info:nnn</code>	13, 16, 349, 363
<code>\msg_line_context:</code>	4149, 4153, 4157, 4181, 4186, 4191, 4206, 4221, 4225, 4229, 4233, 4237, 4241, 4247, 4253, 4259, 4272, 4277, 4282, 4287, 4292, 4296, 4301, 4306, 4310, 4315, 4319, 4324, 4329, 4334, 4338, 4342
<code>\msg_log:nnn</code>	1769, 1774, 1779
<code>\msg_log:nnnnn</code>	327, 1907, 1912, 1917
<code>\msg_log:nnnnnn</code>	319

<code>\msg_new:nnn</code>	4130, 4134, 4138, 4142, 4147, 4151, 4155, 4159, 4165, 4171, 4175, 4179, 4184, 4189, 4204, 4219, 4223, 4227, 4231, 4235, 4239, 4243, 4250, 4256, 4262, 4266, 4270, 4275, 4280, 4285, 4290, 4294, 4299, 4304, 4308, 4313, 4317, 4322, 4327, 4332, 4336, 4340
<code>\msg_term:nnnn</code>	1734, 1739, 2815, 2825, 2854, 2859
<code>\msg_term:nnnnn</code>	1888
<code>\msg_warning:nn</code>	3016, 3133
<code>\msg_warning:nnnn</code>	1944, 1950, 2763, 2768, 3287, 3300, 3645, 3658
<code>\msg_warning:nnnnn</code>	1883, 1893
<code>\multicolsep</code>	85, 88
<code>\multicolsep</code>	2985, 3106

N

<code>\NeedsTeXFormat</code>	3
<code>\newcounter</code>	402
<code>\NewDocumentCommand</code>	1313, 2189, 3145, 3971, 4025, 4083
<code>\NewDocumentEnvironment</code>	2864, 3031, 3186, 3379, 3736
<code>\newlabel</code>	32
<code>\newlabel</code>	385
<code>no-store</code>	1783
<code>\noindent</code>	94, 101
<code>\noindent</code>	2962, 3086, 3346, 3392, 3601, 3704, 3748, 3952
<code>\nointerlineskip</code>	2962, 3086, 3346, 3704
<code>noitemsep</code>	721
<code>\nopagebreak</code>	1011, 1039, 1162, 1241, 1304, 1310
<code>\normalfont</code>	2379, 2550, 2563, 3872
<code>nosep</code>	721

P

Packages:

<code>enumext</code>	23, 35, 59, 81, 89, 110
<code>enumitem</code>	32, 33
<code>expl3</code>	92
<code>footnotehyper</code>	31
<code>hyperref</code>	26, 27, 31, 32, 70, 75, 98, 109
<code>lua-visual-debug</code>	46
<code>multicol</code>	23, 109
<code>shortlst</code>	92
<code>\par</code>	1011, 1039, 1162, 1241, 1304, 1310, 1339, 1356, 2358, 3006, 3021, 3123, 3138, 3261, 3364, 3371, 3601, 3615, 3722, 3729, 3952, 3966
<code>\parindent</code>	3578, 3922
<code>\parsep</code>	44, 47, 90, 91
<code>\parsep</code>	2139, 2147, 2845, 3226, 3233, 3238
<code>parsep</code>	721
<code>\parskip</code>	3579, 3923
<code>\partopsep</code>	91
<code>\partopsep</code>	2846, 3231
<code>partopsep</code>	721
peek commands:	
<code>\peek_meaning:NTF</code>	3468, 3482, 3499, 3510, 3801, 3815, 3832
<code>\peek_meaning_remove:NTF</code>	3475, 3808
<code>\peek_remove_spaces:n</code>	2709
<code>\phantomsection</code>	32
<code>\phantomsection</code>	374
prg commands:	
<code>\prg_do_nothing:</code>	378
<code>\prg_new_protected_conditional:Npnn</code>	191
<code>\prg_replicate:nn</code>	208
<code>\prg_return_false:</code>	195
<code>\prg_return_true:</code>	194
<code>\printkeyans</code>	14, 107, 4025

prop commands:

\prop_count:N	321, 2089, 2346, 2382, 2450, 2553, 2566, 3875
\prop_gput_if_not_in:Nnn	2087
\prop_if_exist:NTF	1767, 3991
\prop_item:Nn	3994
\prop_new:N	1770
\ProvidesExplPackage	4

R

\raggedcolumns	2994, 3112
\ref	71, 73
ref	509, 556, 629
\refstepcounter	3538, 3895
regex commands:	
\regex_match:nnTF	193, 678, 680, 692, 694
\regex_replace_once:nnN	201
\renewcommand	544, 589, 606, 659
\RenewDocumentCommand	2602, 2670, 2705, 2731, 2747
\RequirePackage	17
resume	1476
resume*	1476
rightmargin	765
\Roman	33, 38
\Roman	421
\roman	33, 38
\roman	422, 527, 4015

S

save-ans	1723
save-key	2028
save-ref	1956
save-sep	1956
scan commands:	
\scan_stop:	92, 3240, 3391, 3747, 3982, 3985
seq commands:	
\seq_clear:N	4090
\seq_const_from_clist:Nn	4078
\seq_count:N	322, 3199, 4094
\seq_gclear:N	2600, 2601
\seq_gput_right:Nn	2096, 2613, 2614
\seq_if_empty:NTF	2619, 4040, 4108
\seq_if_exist:NTF	1772, 4038
\seq_item:Nn	3258
\seq_map_function:NN	4099
\seq_map_inline:Nn	4047, 4052, 4087, 4109, 4110
\seq_map_pairwise_function:NNN	2621
\seq_new:N	109, 110, 126, 152, 153, 1775
\seq_pop_left:NN	4098
\seq_put_right:Nn	3159, 4106, 4123
\seq_set_from_clist:Nn	4091
\seq_set_map_e:NNn	4100
\seq_show:N	4042
series	1476
\setcounter	689, 693, 695, 2806, 2848, 3204
\setenumext	6, 108, 4083
\setlength	2140, 2148
show-ans	1956, 1992
show-length	841
show-pos	1992
skip commands:	
\skip_add:Nn	980, 986, 992, 1002, 1006, 1030, 1034, 1131, 1137, 1143, 1153, 1157, 1179, 1232, 1236, 3226
\skip_eval:n	2139, 2147
\skip_gset:Nn	1252, 1256, 1260

\skip_gzero_new:N	1247, 1248
\skip_horizontal:N	813, 825, 837, 3561, 3575, 3919
\skip_horizontal:n	799, 2168, 2176, 2724, 2726, 3559, 3927
\skip_if_eq:nnTF	978, 984, 990, 1054, 1088, 1129, 1135, 1141, 1172, 1177, 1198, 1249, 1271, 1384, 1398, 1412, 1423, 1434, 1445, 1456, 1467
\skip_new:N	69, 70, 74, 75, 76, 77, 78, 130, 171
\skip_set:Nn	963, 967, 1016, 1020, 1057, 1061, 1065, 1072, 1076, 1080, 1091, 1096, 1100, 1106, 1111, 1116, 1174, 1175, 1176, 1183, 1187, 1191, 1200, 1205, 1209, 1212, 1216, 1220, 1251, 1255, 1273, 1277, 1281, 1287, 1291, 1295, 3220, 3234
\skip_set_eq:NN	2804, 2844, 2845, 3578, 3579, 3922, 3923
\skip_use:N	965, 969, 1004, 1008, 1012, 1032, 1036, 1055, 1074, 1083, 1089, 1094, 1098, 1109, 1113, 1114, 1119, 1155, 1159, 1185, 1385, 1389, 1392, 1399, 1403, 1406, 3006
\skip_zero:N	2846, 2985, 3106, 3231, 3232
\skip_zero_new:N	1167, 1168, 1169, 1246, 1268, 1269, 1270
\c_zero_skip	978, 984, 990, 1055, 1089, 1129, 1135, 1141, 1172, 1177, 1198, 1249, 1271, 1385, 1399, 1412, 1423, 1434, 1445, 1456, 1467
\small	4003, 4007, 4011, 4015, 4019, 4023
\star	2584
start	701
\stepcounter	2606, 3166
str commands:	
\c_backslash_str	4149, 4153, 4157, 4161, 4162, 4163, 4167, 4168, 4264, 4268, 4277, 4278, 4282, 4283, 4287, 4288, 4319, 4320, 4324, 4329, 4330
\c_colon_str	2345, 2449, 3982
\c_left_brace_str	4229, 4233, 4246, 4252, 4258
\c_right_brace_str	4229, 4233, 4246, 4252, 4258
\str_case:nn	213, 268
\str_case:nnTF	1499, 1508, 2067, 2075
\str_clear:N	2896, 3417
\str_count:n	208
\str_if_empty:NTF	1520, 1561, 1588
\str_if_eq:nnTF	2807, 2850
\str_if_in:nnTF	3978
\str_new:N	125, 166
\str_set:Nn	501, 502, 503, 1973, 1974, 1997, 1998
\string	367
\strutbox	1059, 1063, 1067, 1078, 1082, 1093, 1102, 1108, 1118, 1131, 1137, 1143, 1174, 1175, 1176, 1179, 1189, 1193, 1202, 1209, 1214, 1222, 1251, 1252, 1255, 1262, 1275, 1283, 1289, 1297, 3236

T

TeX and LaTeX commands:	
\@auxout	383
\@currenvir	213, 268
\protected@write	383
text commands:	
\text_expand:n	3974
\textasteriskcentered	1970, 1984
\thepage	389
tl commands:	
\c_space_tl	2531, 4191, 4206, 4229, 4233
\tl_clear:N	474, 480, 1954, 2014, 2024, 2045, 2053, 2219, 2392, 2466, 3838
\tl_clear_new:N	431

<code>\tl_const:Nn</code> . . . . .	37, 415		
<code>\tl_gclear:N</code> . . . . .	313, 314, 315, 1541, 1546, 2742, 3375, 3562, 3733	<code>\tl_set_eq:NN</code> . . . . .	3869, 4118
<code>\tl_gclear_new:N</code> . . . . .	1528	<code>\tl_set_eq:NN</code> . . . . .	442, 537, 540, 584, 586, 601, 603, 654, 656, 2283, 2417, 2430, 2693, 2697, 3178, 3180
<code>\tl_gput_right:Nn</code> . . . . .	416	<code>\tl_to_str:n</code> . . . . .	1614, 1620, 1625, 3974
<code>\tl_greplace_all:Nnn</code> . . . . .	437	<code>\tl_trim_spaces:n</code> . . . . .	432, 4106, 4118, 4124
<code>\tl_gset:Nn</code> . . . . .	245, 246, 259, 260, 1529, 1542, 1547, 1766, 3505	<code>\tl_use:N</code> . . . . .	438, 441, 553, 618, 625, 668, 867, 871, 875, 879, 883, 887, 891, 895, 899, 903, 907, 911, 915, 919, 923, 927, 2173, 2290, 2298, 2309, 2323, 2328, 2340, 2636, 2642, 2666, 2684, 2688, 2696, 2733, 2734, 2741, 2749, 2750, 2756, 2871, 3037, 3183, 3369, 3565, 3576, 3580, 3727, 3909, 3920, 3926, 3930, 4028, 4029, 4030, 4031, 4032, 4045, 4102
<code>\tl_gset_eq:NN</code> . . . . .	433, 2651, 3555	token commands:	
<code>\tl_if_blank:nTF</code> . . . . .	3553	<code>\token_to_str:N</code> . . . . .	385
<code>\tl_if_empty:nTF</code> . . . . .	532, 551, 579, 596, 616, 623, 649, 666, 1554, 1559, 1581, 1586, 1644, 1708, 1716, 1745, 1804, 1924, 2103, 2134, 2245, 2403, 2476, 2525, 2721, 3841, 4121	<code>\topsep</code> . . . . .	2140, 2148
<code>\tl_if_empty:nTF</code> . . . . .	1609	<code>topsep</code> . . . . .	721
<code>\tl_if_exist:nTF</code> . . . . .	1614	<code>\typeout</code> . . . . .	353, 356, 366, 367
<code>\tl_if_novalue:nTF</code> . . . . .	2220, 2231, 2400, 2474, 2510, 2604, 2629, 2647, 2652, 2681, 2894, 3197, 3415, 3775, 3839, 4085		
<code>\tl_map_inline:Nn</code> . . . . .	199, 434	<b>U</b>	
<code>\tl_new:N</code> . . . . .	34, 39, 40, 43, 44, 50, 52, 53, 54, 56, 57, 90, 91, 92, 98, 99, 100, 102, 103, 104, 105, 106, 107, 111, 114, 116, 117, 123, 124, 134, 135, 136, 143, 144, 145, 148, 165, 168	<code>\u</code> . . . . .	202
<code>\tl_put_left:Nn</code> . . . . .	2111, 2144, 2228, 2537, 2572, 3857, 3860	use commands:	
<code>\tl_put_right:Nn</code> . . . . .	432, 542, 587, 604, 657, 2115, 2150, 2230, 2236, 2244, 2247, 2257, 2262, 2265, 2271, 2297, 2307, 2321, 2337, 2343, 2348, 2395, 2398, 2405, 2407, 2434, 2439, 2444, 2447, 2456, 2469, 2472, 2478, 2483, 2493, 3843, 3844	<code>\use:N</code> . . . . .	209, 2738, 2873
<code>\tl_remove_all:Nn</code> . . . . .	4120	<code>\use:n</code> . . . . .	1490, 2058, 3980
<code>\tl_remove_once:Nn</code> . . . . .	2285, 2419	<code>\use_none:nn</code> . . . . .	377
<code>\tl_replace_all:Nnn</code> . . . . .	436	<code>\usecounter</code> . . . . .	2805, 2847
<code>\tl_reverse:N</code> . . . . .	2284, 2286, 2418, 2420		
<code>\tl_set:Nn</code> . . . . .	45, 272, 279, 286, 401, 475, 479, 484, 485, 531, 576, 648, 796, 810, 822, 834, 1643, 1744, 2015, 2025, 2046, 2054, 2376, 2512, 2547, 2560, 2649, 3846,	<b>V</b>	
		<code>\value</code> . . . . .	1557, 1563, 1570, 1576, 1584, 1590, 1597, 1603
		<code>\vspace</code> . . . . .	1046, 1389, 1392, 1403, 1406, 1416, 1418, 1427, 1429, 1438, 1440, 1449, 1451, 1460, 1462, 1471, 1473, 2139, 2147, 3194, 3205, 3616, 3967
		<b>W</b>	
		<code>widest</code> . . . . .	701
		<code>wrap-ans</code> . . . . .	1956
		<code>wrap-label</code> . . . . .	445
		<code>wrap-label*</code> . . . . .	445
		<code>wrap-opt</code> . . . . .	1956