

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-04^{*}

©2024 by Pablo González[†]

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

1	Introduction	1	5	The storage system	10
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for label and ref	11
1.3	User interface	3	5.1.2	Keys for wrap and display	11
1.3.1	Internal counters	3	5.1.3	Keys for debug and checking	11
1.3.2	Support for multicol	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for minipage	4	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	12
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
2	The environments provided	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	13
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	14
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	15
3	The command <code>\setenumext</code>	6	6	Full examples	16
4	The keyval system	6	7	The way of non-enumerated lists	19
4.1	Keys for label and ref	6	8	References	21
4.2	Keys for spaces	7	9	Change history	21
4.2.1	Vertical spaces	7	10	Index of Documentation	22
4.2.2	Horizontal spaces	8	11	Implementation	24
4.3	Keys for add code	8	12	Index of Implementation	119
4.4	Keys for start, series and resume	9			
4.5	Keys for multicol	9			
4.6	Keys for minipage	9			
4.6.1	The command <code>\miniright</code>	10			
4.6.2	The key <code>mini-right</code>	10			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-06-04.

[†]E-mail: 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 (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.
The enumext package loads and requires multicol[3] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TTeX. It has been tested with the standard classes provided by L^AT_EX: book, report, article and letter on 10pt, 11pt and 12pt.

1 Introduction

In the L^AT_EX 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^AT_EXze 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^AT_EXze 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

A

A) B) C)

A

duck

D) E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B
- 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 / 131

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 \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 \TeX Live and \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.

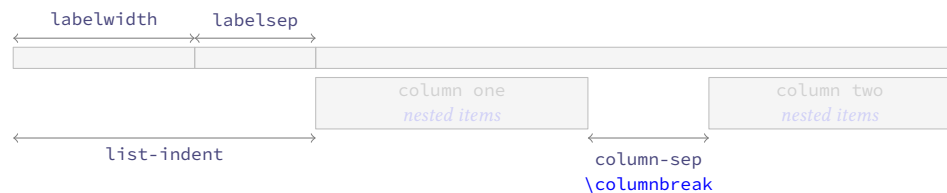


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 \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 fourth level.
- (a) This text is in the second level.
- X This text is in the first level.
- i. This text is in the third 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.
2. This text is in the first level.
- X 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 argument* “*” 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 columns \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 ‘0’ for `\arabic*`, ‘M’ for `\Alph*`, ‘m’ for `\alph*`, ‘VIII’ for `\Roman*` and ‘viii’ for `\roman*`.

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

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

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

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

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

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

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

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

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

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

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

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

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_EX 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_EX 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 \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 argument* “*” inhibits the use of `\centering` command i.e. the usual L^AT_EX 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` `\anskey`[⟨keys⟩]{⟨content⟩}

The command `\anskey` takes a mandatory argument {⟨content⟩} and “stores” it in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. By design the command cannot be nested or passed *verbatim* in the argument and it is assumed that each `\item` or `\item*` within the environment in which it is active it has a “single execution” of `\anskey` unless `\item` or `\item*` open a nested level or use the `no-store` key.

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 \TeX will be used.

The `\anskey` command is available for all levels of the `enumext` environment and the `enumext*` environment, but is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

5.2.1 Keys for `\anskey`

By default the {⟨content⟩} argument passed to `\anskey` when “storing” in the *sequence* {⟨store name⟩} has the form `\item` {⟨content⟩}, the following {⟨keys⟩} allow modifying the way in which it is “stored” in the *sequence*.

`break-col` {⟨value forbidden⟩} default: *not used*
 Stores {⟨content⟩} in the *sequence* {⟨store name⟩} of the form `\columnbreak` `\item` {⟨content⟩}.

`item-join` = {⟨columns⟩} default: *not set*
 Set the *number of columns* to be used for `\item`(⟨columns⟩) and stores {⟨content⟩} in the *sequence* {⟨store name⟩} of the form `\item`(⟨columns⟩) {⟨content⟩}.

`item-star` {⟨value forbidden⟩} default: *not used*
 Stores {⟨content⟩} in the *sequence* {⟨store name⟩} of the form `\item*` {⟨content⟩}.

`item-sym*` = {⟨symbol⟩} default: *\$\star\$*
 Sets the *symbol* for `\item*` when using the key `item-star` and stores {⟨content⟩} in the *sequence* {⟨store name⟩} of the form `\item*`[⟨symbol⟩] {⟨content⟩}. The *symbol* can be in text or math mode, for example `item-sym*={\ast}` stores `\item*`[\$\ast\$] {⟨content⟩}.

`item-pos*` = {⟨rigid length⟩} default: *not set*
 Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores {⟨content⟩} in the *sequence* {⟨store name⟩} of the form `\item*`[⟨symbol⟩][⟨offset⟩] {⟨content⟩}.

Example

```
\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}
```

- | | |
|--|---|
| <ul style="list-style-type: none"> ★ 1. Text containing our instructions or questions. * first answer 2. Text containing our instructions or questions. (a) Question. * second answer | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. * third answer 4. Text containing our instructions or questions. * fourth answer |
|--|---|

5.3 The environment anskey*

anskey* `\begin{anskey*}[\langle key = val \rangle] \langle body content \rangle \end{anskey*}`

The environment `anskey*` takes a mandatory `\langle body content \rangle` and “stores” it in the *sequence* and *prop list* `\langle store name \rangle` set by `save-ans` key. By design the environment cannot be nested or passed *verbatim* in the body and it is assumed that each `\item` or `\item*` within the environment in which it is active it has a “single execution” of `\anskey` unless `\item` or `\item*` open a nested level or use the `no-store` key.

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^AT_EX will be used.

The `anskey*` environment uses the same `\langle keys \rangle` as the `\anskey` command and is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

Example

```
\begin{enumext}[save-ans=test,show-pos=true,start=5]
  \item* Text containing our instructions or questions.
    \begin{anskey*}
      \langle first answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          \langle second answer \rangle
        \end{anskey*}
      \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle third answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle fourth answer \rangle
    \end{anskey*}
\end{enumext}
```

- | | |
|--|---|
| <ul style="list-style-type: none"> ★ 5. Text containing our instructions or questions. [5] first answer 6. Text containing our instructions or questions. (a) Question. [6] second answer | <ul style="list-style-type: none"> 7. Text containing our instructions or questions. [7] third answer 8. Text containing our instructions or questions. [8] fourth answer |
|--|---|

5.4 The environments keyans and keyans*

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

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

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

The `\langle keys \rangle` set in the optional argument of the environment are the same (almost) as those of the `enumext` and `enumext*` environments and have higher precedence than those set by `\setenumext[\langle keyans \rangle]{\langle key`

= val}} or \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*).

5.4.1 The \item* in keyans and keyans*

\item* \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 sequence and prop list {⟨store name⟩} set by save-ans key in the “first level” of the enumext or enumext* environments.

The starred 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 \item* will only appear “once” within the environment.


• The behavior of \item* in keyans and keyans* environments is NOT the same as in the enumext or enumext* environments.

Example

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

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


Some text

5.5 The environment keyanspic

keyanspic \begin{keyanspic}[⟨n° above, n° 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.

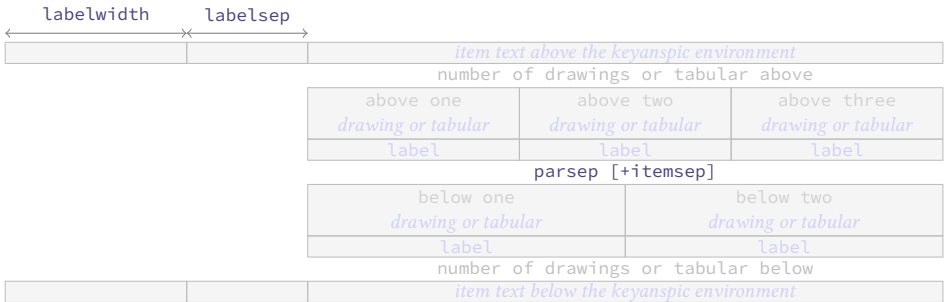


Figure 6: Representation of the keyanspic environment with optional argument [3,2] in enumext.

The optional argument determines the number drawings or tabular “above” and “below” within the environment. The vertical separation between “above” and “below” is controlled by the values set by

`parsep` and `itemsep` keys passed to `keyans` environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

5.5.1 The command `\anspic`

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

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

The *starred argument* ‘*’ 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 argument* ‘*’ will only appear “once” within the environment.

Example

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

1. Question with images.



A)



B)



C)



D)



* E)[note]

5.6 Printing stored content

5.6.1 The command `\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.

5.6.2 The command `\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 argument* ‘*’ 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 managing the *<keys>* in the “first level” of the environment in which the “stored content” of the *sequence* `{<store name>}` will be printed, if the *starred argument* ‘*’ is used it will be `enumext*` otherwise `enumext`.

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* `{<store name>}` 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* `{<store name>}` it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

Since the command encapsulates by default the `enumext` environment or the `enumext*` environment, we must take some considerations:

- If we execute `\printkeyans*{<store name>}` and the *sequence* `{<store name>}` already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeyans*{<store name>}` and the *sequence* `{<store name>}` contains any `enumext` environments, they will start with the `<keys>` set for the first level unless they are set in the optional argument or `save-key` is used to modify it.
- If we execute `\printkeyans{<store name>}` and the *sequence* `{<store name>}` contains any environment `enumext*`, they will start with the `<keys>` set by default unless they are set in the optional argument or `save-key` is used to modify it.

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[<print , 1>]{<keys>}` and `\setenumext[<print*>]{<keys>}`. If we need to set the `<keys>` for the environment `enumext` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print , level>]{<keys>}` and if we need to set the `<keys>` for the environment `enumext*` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print , *>]{<keys>}`.

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 \LaTeX2e\ 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)

~~TeX~~X2e 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:

```
$ pdftdetach -saveall enumext.pdf
```

and then you can use the excellent `arara`¹ tool to compile them.

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

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

- ☐ A 36 km/h.
☐ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

- ☐ A 36 km/h.
☐ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) 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 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

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 2

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

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

- ☐ A 36 km/h.
☒ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) 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 $1,00 \times 10^2$ m/s espressa in km/h è:

- ☐ A 36 km/h.
☒ B 360 km/h.
☐ C 27,8 km/h.
☐ D $3,60 \times 10^8$ km/h.

4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) 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

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

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

A

value

B

value

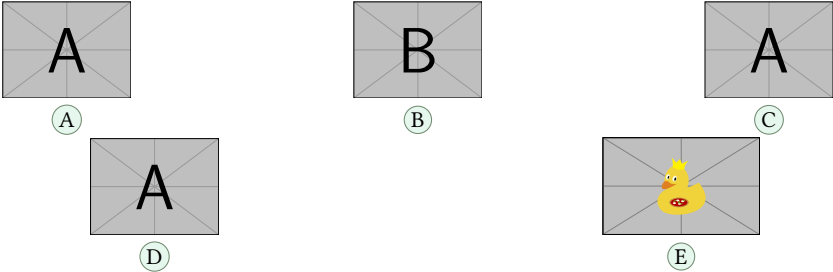
C

value

D

value

E

value

5. Question with image on left side:
- A

value
- B

value
- C

value
- D

correct
- E

value



Test keys

1. B, $x = 5$

2. D

3. C, some note
4. E, A duck

5. D, other note

Example 4

A “simple worksheet” using ducks :) 🦆.

Factor $x^2 - 2x + 1$

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

- (a) $\alpha > \delta$
- (b) \LaTeX is cool?

Related to Linux

- (a) You use linux?
- (b) Usually uses the package manager?
- (c) Rate the following package and class

i.

xsim-exam

ii.

xsim

iii.

exsheets

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

Example 5

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

- 1

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she

- has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

- 2

Which choice best describes what happens in the passage?

A) One character argues with another character

ter who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character for pursuing an unexpected course of action.

3

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character for pursuing an unexpected course of action.

4

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character for pursuing an unexpected course of action.

1. A) 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 \TeX for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

- First level item
 - Second level item
 - * Third level item
 - Fourth level item
 - First level item
- * First level item
 - ◊ Second level item
 - Third level item
 - ★ Fourth level item
 - * First level item

Fake description environment

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

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

Something A short *one-line* description text.

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

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

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

Something A short *one-line* description text.

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

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

Description indented by label

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

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

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

Something	A short one-line description. This is an entry <i>without</i> 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 <i>without</i> 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 <i>without</i> 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 <i>without</i> 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. 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 \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 \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. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [7] The \LaTeX Project. “Extensive support for hypertext in \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 \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [10] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \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 2024-06-04 – First public release.

10 Index of Documentation

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

C

Document class:

article2

book2

exam2

letter2

report2

\columnbreak4, 12

\columnsep10

Commands provide by enumext:

\anskey4, 10–13

\anspic*4, 10–12, 14, 15

\anspic14, 15

\getkeyans4, 12, 15

\item*4–7, 10–14

\itemwidth5

\item5–7, 9, 10, 12, 13

\miniright4, 10

\printkeyans4, 6, 11, 15

\setenumext4–7, 11, 13, 14, 16

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

anskey*13

enumext*4–16

enumext4–16, 19

keyans*4–14

keyanspic4, 6, 8, 10–14, 19

keyans4–15, 19

Environments:

enumerate1, 3, 5, 20

figure5

list3, 9, 20

minipage3–5, 10, 20

multicols3, 4, 10

table5

task5

F

\footnote5

I

\item3, 5

\itemsep8

K

Keys for command provide by enumext:

break-col12

item-join12

item-pos*12

item-star12

item-sym*12

Keys for environments provide by enumext:

above*8

above8

after9, 10

align7, 20

before*9

before9

below*8

below8

check-ans12

columns-sep4, 10

columns4, 8, 10

first9

font7

item-pos*5, 6

item-sym*5, 6

itemindent8

itemsep8, 15

labelsep3, 5–10, 12, 19, 20

labelwidth3, 6, 7, 9, 10, 12, 19, 20

labelwidth5

label7, 9, 14, 19, 20

list-indent3, 9

list-offset3, 9, 20

listparindent9

mark-ans11, 12

mark-pos12

mark-ref11

mini-env4, 8, 10

mini-right*6, 10

mini-right6, 10

mini-sep4, 10

no-store11–13

noitemsep8

nosep8, 19

parsep8, 15

partopsep8

ref4, 7

resume*6, 9, 10

resume6, 9, 10

rightmargin8

save-ans4, 6, 9–15

save-key9, 11, 16

save-ref4, 7, 11–13, 15

save-sep11

series6, 9, 10

show-ans11

show-length7

show-pos11, 12, 15

start9

topsep8

widest7

wrap-ans11

wrap-label*7, 20

wrap-label7

wrap-opt11

L

\label4

Labels provide by enumext:

\Alph*7, 14

\Roman*	7	l3keys	6
\alph*	7	l3prop	1, 20
\arabic*	7	l3seq	1, 20
\roman*	7	multicol	1, 2, 4, 20
\labelsep	3, 7	task	5, 6
\labelwidth	3, 7	xsim	2
\linewidth	10	\parsep	8
\listparindent	9	\partopsep	8
P		R	
Packages:		\raggedcolumns	4
enumerate	20	\ref	4
enumext	1–6, 14, 20	\rightmargin	8
enumitem	3–5, 9, 20		
footnotehyper	4, 5	T	
hyperref	4, 5, 11–13, 20	\topsep	8

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_EXpert 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^AT_EX3 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^AT_EX to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2024-06-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-06-04}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the `multicol` and `scontents` packages are 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   \IfPackageLoadedTF { scontents }
20   {
21     \msg_info:nnn { enumext } { package-load } { scontents }
22   }
23   {
24     \msg_info:nnn { enumext } { package-not-load } { scontents }
25     \RequirePackage{scontents}
26   }
27 }
```

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.

```
\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int
```

Integer variables will control the nesting levels of the environments and `\anskey` command.

```
28 \int_new:N \l__enumext_level_int
29 \int_new:N \l__enumext_level_h_int
30 \int_new:N \l__enumext_anskey_level_int
31 \int_new:N \l__enumext_keyans_level_int
32 \int_new:N \l__enumext_keyans_level_h_int
33 \int_new:N \l__enumext_keyans_pic_level_int
```

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

```
\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_first_bool
\l__enumext_keyans_env_bool
```

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.

```
34 \bool_new:N \l__enumext_starred_bool
35 \bool_new:N \g__enumext_starred_bool
36 \bool_new:N \l__enumext_starred_first_bool
37 \bool_new:N \l__enumext_standar_bool
38 \bool_new:N \g__enumext_standar_bool
39 \bool_new:N \l__enumext_standar_first_bool
40 \bool_new:N \l__enumext_keyans_env_bool
```

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

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

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

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

The initial values of these variables are set by the function `__enumext_define_counters:Nn` (§11.9) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.12).

```
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { l__enumext_counter_#1_tl }
44 }
45 \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.)

```
\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl
```

Internal variables used by `ref` key (§11.12).

```
46 \tl_const:Nn \c__enumext_counter_style_tl
47 { { arabic } { roman } { Roman } { alph } { Alph } }
48 \tl_new:N \l__enumext_ref_key_arg_tl
49 \tl_new:N \l__enumext_ref_the_count_tl
50 \cs_set_protected:Npn \__enumext_tmp:n #1
51 {
52   \tl_new:c { l__enumext_renew_the_count_#1_tl }
53   \tl_new:c { l__enumext_the_counter_#1_tl }
54   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
55 }
56 \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.)

```
\g__enumext_resume_int
\g__enumext_resume_vii_int
\l__enumext_resume_name_tl
\l__enumext_resume_active_bool
\g__enumext_item_symbol_tl
\g__enumext_standar_series_tl
\g__enumext_starred_series_tl
```

Internal variables used by `resume`, `resume*` and `series` keys. The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§11.28).

```
57 \int_new:N \g__enumext_resume_int
58 \int_new:N \g__enumext_resume_vii_int
59 \tl_new:N \l__enumext_resume_name_tl
60 \bool_new:N \l__enumext_resume_active_bool
61 \tl_new:N \g__enumext_item_symbol_tl
62 \tl_new:N \g__enumext_standar_series_tl
63 \tl_new:N \g__enumext_starred_series_tl
```

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

```
\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box
```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default `<label style>` and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§11.13) and `label` (§11.11) keys.

```
64 \dim_new:N \l__enumext_current_widest_dim
65 \tl_new:N \g__enumext_counter_styles_tl
66 \tl_new:N \g__enumext_widest_label_tl
67 \box_new:N \l__enumext_label_width_by_box
```

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

```
\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim
```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§11.15).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.32.1) which determines the internal values for `\leftmargin` and `\itemindent`.

```
68 \cs_set_protected:Npn \__enumext_tmp:n #1
69 {
70   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
71   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
72   \dim_new:c { \l__enumext_leftmargin_#1_dim }
73   \dim_new:c { \l__enumext_itemindent_#1_dim }
74 }
75 \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 §11.19).

```
76 \cs_set_protected:Npn \__enumext_tmp:n #1
77 {
78   \skip_new:c { \l__enumext_multicols_above_#1_skip }
79   \skip_new:c { \l__enumext_multicols_below_#1_skip }
80 }
81 \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 (§11.20.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.18, §11.20).

```
82 \int_new:N \g__enumext_minipage_stat_int
83 \skip_new:N \l__enumext_minipage_left_skip
84 \skip_new:N \l__enumext_minipage_right_skip
85 \skip_new:N \l__enumext_minipage_after_skip
86 \skip_new:N \g__enumext_minipage_right_skip
87 \skip_new:N \g__enumext_minipage_after_skip
88 \cs_set_protected:Npn \__enumext_tmp:n #1
89 {
90   \dim_new:c { \l__enumext_minipage_left_#1_dim }
91   \bool_new:c { \l__enumext_minipage_active_#1_bool }
92 }
93 \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 (§11.13), 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 (§11.11). 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

```
94 \cs_set_protected:Npn \__enumext_tmp:n #1
95 {
96   \bool_new:c { \l__enumext_wrap_label_#1_bool }
97   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
98   \int_new:c { \l__enumext_start_#1_int }
99   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
100  \tl_new:c { \l__enumext_label_fill_left_#1_tl }
101  \tl_new:c { \l__enumext_label_fill_right_#1_tl }
102  \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
103  \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
104 }
105 \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_seq
\l__enumext_store_anskey_arg_tl
\l__enumext_store_anskey_env_tl
\l__enumext_store_anskey_opt_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` (§11.25) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§11.30.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§11.35.1) for the `keyanspic` environment.

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

```
106 \bool_new:N \l__enumext_store_active_bool
107 \tl_new:N \l__enumext_store_name_tl
108 \tl_new:N \g__enumext_store_name_tl
109 \seq_new:N \l__enumext_store_anskey_seq
110 \tl_new:N \l__enumext_store_anskey_arg_tl
111 \tl_new:N \l__enumext_store_anskey_env_tl
112 \tl_new:N \l__enumext_store_anskey_opt_tl
113 \int_new:N \l__enumext_store_columns_join_int
114 \tl_new:N \l__enumext_store_keyans_label_tl
115 \tl_new:N \l__enumext_store_keyans_item_opt_tl
116 \tl_new:N \l__enumext_keyans_item_opt_tl
117 \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` (§11.40).

```
118 \tl_new:N \l__enumext_setkey_tmpa_tl
119 \tl_new:N \l__enumext_setkey_tmpb_tl
120 \int_new:N \l__enumext_setkey_tmpa_int
121 \seq_new:N \l__enumext_setkey_tmpa_seq
122 \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` (§11.39) and `save-key` key.

```
123 \tl_new:N \l__enumext_print_keyans_starred_tl
124 \cs_set_protected:Npn \__enumext_tmp:n #1
125 {
126   \tl_new:c { \l__enumext_store_save_key_#1_tl }
127   \bool_new:c { \l__enumext_store_save_key_#1_bool }
128   \tl_new:c { \l__enumext_store_active_keys_#1_tl }
129   \tl_new:c { \l__enumext_print_keyans_#1_tl }
130   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
131 }
132 \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` (§11.25), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§11.24).

```
133 \bool_new:N \l__enumext_show_answer_bool
134 \bool_new:N \l__enumext_show_position_bool
135 \tl_new:N \l__enumext_mark_ref_sym_tl
136 \tl_new:N \l__enumext_mark_answer_sym_tl
137 \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 (§11.35.2).

```
138 \seq_new:N \l__enumext_keyans_pic_body_seq
139 \dim_new:N \l__enumext_keyans_pic_width_dim
140 \int_new:N \l__enumext_keyans_pic_above_int
141 \int_new:N \l__enumext_keyans_pic_below_int
142 \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
143 \bool_new:N \l__enumext_check_answers_bool
144 \bool_new:N \l__enumext_check_ans_key_bool
145 \bool_new:N \g__enumext_check_ans_key_bool
146 \tl_new:N \l__enumext_check_start_line_env_tl
147 \tl_new:N \g__enumext_start_line_tl
148 \tl_new:N \g__enumext_envir_name_tl
149 \int_new:N \g__enumext_check_starred_cmd_int
150 \int_new:N \g__enumext_item_anskey_int
151 \int_new:N \g__enumext_item_number_int
152 \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 (§11.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

153 \bool_new:N \l__enumext_hyperref_bool
154 \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.

```

155 \tl_new:N \l__enumext_newlabel_arg_one_tl
156 \tl_new:N \l__enumext_newlabel_arg_two_tl
157 \tl_new:N \l__enumext_store_write_aux_file_tl
158 \cs_set_protected:Npn \__enumext_tmp:n #1
159 {
160   \tl_new:c { l__enumext_label_copy_#1_tl }
161 }
162 \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`.

```

163 \int_new:N \g__enumext_footnote_int
164 \seq_new:N \g__enumext_footnote_arg_seq
165 \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.

```

166 \cs_set_protected:Npn \__enumext_tmp:n #1
167 {
168   \bool_new:c { l__enumext_item_starred_#1_bool }
169   \int_new:c { l__enumext_item_column_pos_#1_int }
170   \int_new:c { g__enumext_item_count_all_#1_int }
171   \int_new:c { l__enumext_joined_item_#1_int }
172   \int_new:c { l__enumext_joined_item_aux_#1_int }
173   \int_new:c { l__enumext_tmpa_#1_int }
174   \box_new:c { l__enumext_item_text_#1_box }
175   \dim_new:c { l__enumext_joined_width_#1_dim }
176   \dim_new:c { l__enumext_item_width_#1_dim }
177   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
178   \str_new:c { l__enumext_align_label_#1_str }
179   \bool_new:c { g__enumext_minipage_active_#1_bool }
180   \tl_new:c { g__enumext_miniright_code_#1_tl }
181   \bool_new:c { g__enumext_minipage_center_#1_bool }
182   \dim_new:c { g__enumext_minipage_right_#1_dim }
183   \skip_new:c { g__enumext_minipage_right_#1_skip }
184 }
185 \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`.

```
186 \clist_const:Nn \c__enumext_all_envs_clist
187 {
188   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
189   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
190 }
```

(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.

```
191 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
192 {
193   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
194 }
```

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

`__enumext_after_env:nn` and `__enumext_before_env:nn` A internal “hook” functions for execute code `mini-rigth` and `mini-rigth*` keys outside the `enumext*` and `keyans*` environments and print check-ans outside the `enumext` and `enumext*` environments.

```
195 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
196 {
197   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
198 }
199 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
200 {
201   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
202 }
```

(End of definition for `__enumext_after_env:nn` and `__enumext_before_env:nn`.)

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

```
203 \cs_new:Nn \__enumext_level:
204 {
205   \int_to_roman:n { \l__enumext_level_int }
206 }
```

(End of definition for `__enumext_level:`.)

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

```
207 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
208 {
209   \regex_match:nnTF { ^[\+-]?[\d]+$ } {#1} % $
210   { \prg_return_true: }
211   { \prg_return_false: }
212 }
```

(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.

```
213 \cs_new_protected:Nn \__enumext_regex_counter_style:
214 {
215   \tl_map_inline:Nn \c__enumext_counter_style_tl
216   {
217     \regex_replace_once:nnN { \c{##1}\* }
218     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
219   }
220 }
```

(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.

```

221 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
222 {
223   * ~ #2
224   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
225   = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
226 }

```

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

11.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`
`__enumext_is_on_first_level:`

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.

```

227 \cs_new_protected:Nn \__enumext_is_not_nested:
228 {
229   \str_case:en { \@currentenv }
230   {
231     {enumext}
232     {
233       \bool_lazy_and:nnT
234       { \bool_not_p:n { \g__enumext_standar_bool } }
235       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
236       {
237         \bool_gset_true:N \g__enumext_standar_bool
238       }
239     }
240     {enumext*}
241     {
242       \bool_lazy_and:nnT
243       { \bool_not_p:n { \g__enumext_starred_bool } }
244       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
245       {
246         \bool_gset_true:N \g__enumext_starred_bool
247       }
248     }
249   }
250 }

```

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.

```

251 \cs_new_protected:Nn \__enumext_is_on_first_level:
252 {
253   \bool_lazy_all:nT
254   {
255     { \bool_if_p:N \g__enumext_standar_bool }
256     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
257     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
258   }
259   {
260     \bool_set_true:N \l__enumext_standar_first_bool
261     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
262     \tl_gset:Nn \g__enumext_start_line_tl
263     {
264       on ~ line ~ \exp_not:V \inputlineno
265     }
266   }
267   \bool_lazy_all:nT
268   {
269     { \bool_if_p:N \g__enumext_starred_bool }
270     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
271     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
272   }
273   {
274     \bool_set_true:N \l__enumext_starred_first_bool
275     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
276     \tl_gset:Nn \g__enumext_start_line_tl

```

```

277         {
278             on ~ line ~ \exp_not:V \inputlineno
279         }
280     }
281 }

```

(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.

```

282 \cs_new_protected:Nn \__enumext_keyans_save_start_line:
283 {
284     \str_case:en { \@currentenv }
285     {
286         {keyans}
287         {
288             \tl_set:Nc \l__enumext_check_start_line_env_tl
289             {
290                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
291             }
292         }
293         {keyans*}
294         {
295             \tl_set:Nc \l__enumext_check_start_line_env_tl
296             {
297                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
298             }
299         }
300         {keyanspic}
301         {
302             \tl_set:Nc \l__enumext_check_start_line_env_tl
303             {
304                 in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
305             }
306         }
307     }
308 }

```

(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.

```

309 \cs_new_protected:Nn \__enumext_reset_global_vars:
310 {
311     \__enumext_reset_global_int:
312     \__enumext_reset_global_bool:
313     \__enumext_reset_global_tl:
314 }
315 \cs_new_protected:Nn \__enumext_reset_global_int:
316 {
317     \int_gzero:N \g__enumext_item_number_int
318     \int_gzero:N \g__enumext_item_anskey_int
319     \int_gzero:N \g__enumext_item_answer_diff_int
320 }
321 \cs_new_protected:Nn \__enumext_reset_global_bool:
322 {
323     \bool_gset_false:N \g__enumext_check_ans_key_bool
324     \bool_gset_false:N \g__enumext_standar_bool
325     \bool_gset_false:N \g__enumext_starred_bool
326 }
327 \cs_new_protected:Nn \__enumext_reset_global_tl:
328 {
329     \tl_gclear:N \g__enumext_store_name_tl
330     \tl_gclear:N \g__enumext_start_line_tl
331     \tl_gclear:N \g__enumext_envir_name_tl
332 }

```

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

`__enumext_log_global_vars:` 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.

```

333 \cs_new_protected:Nn \__enumext_log_global_vars:
334 {
335     \msg_log:nneeee { enumext } { prop-seq-int-hook }
336     { \g__enumext_store_name_tl }
337     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
338     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
339     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
340 }

```

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.

```

341 \cs_new_protected:Nn \__enumext_log_answer_vars:
342 {
343     \msg_log:nneeee { enumext } { item-answer-hook }
344     { \int_use:N \g__enumext_item_number_int }
345     { \int_use:N \g__enumext_item_anskey_int }
346     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
347 }

```

(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 \TeX has the following plain form:

```

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

```

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

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

```

348 \__enumext_at_begin_document:n
349 {
350     \cs_new_eq:NN \__enumext_start_list:nn \list
351     \cs_new_eq:NN \__enumext_stop_list: \endlist
352     \cs_new_eq:NN \__enumext_item_std:w \item
353 }

```

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

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

```

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

```

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

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

```

354 \__enumext_at_begin_document:n
355 {
356     \cs_new_eq:NN \__enumext_minipage:w \minipage
357     \cs_new_eq:NN \__enumext_endminipage: \endminipage
358 }

```

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

11.7 The internal minipage environment

`__enumext_internal_mini_page:`
`__enumext_mini_env*`

The function `__enumext_internal_mini_page:` creates a internal `__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. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§11.33) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.36)

```

359 \cs_new_protected:Nn \__enumext_internal_mini_page:
360 {
361   \int_compare:nNtT { \__enumext_level_int } = { 0 }
362   {
363     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
364     {
365       \__enumext_minipage:w [ t ] { ##1 }
366       \legacy_if_gset_false:n { @minipage }
367       \vspace { 0pt }
368     }
369     { \__enumext_endminipage: }
370   }
371 }
```

(End of definition for `__enumext_internal_mini_page:` and `__enumext_mini_env*`.)

11.8 Compatibility with hyperref and footnotehyper

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

```

372 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
373 \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 `\l__enumext_footnotes_key_bool` to “true”.

```

374 \cs_new_protected:Nn \__enumext_after_hyperref:
375 {
376   \IfPackageLoadedTF { hyperref }
377   {
378     \msg_info:nnn { enumext } { package-load } { hyperref }
379     \bool_set_true:N \l__enumext_hyperref_bool
380     \IfHyperBoolean{hyperfootnotes}
381     {
382       \typeout{hyperfootnotes=true}
383       \bool_set_true:N \l__enumext_footnotes_key_bool
384     }
385     { \typeout{hyperfootnotes=false} }
386   }
387   { }
```

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`.

```

388   \bool_if:NT \l__enumext_footnotes_key_bool
389   {
390     \IfPackageLoadedTF { footnotehyper }
391     {
392       \msg_info:nnn { enumext } { package-load } { footnotehyper }
393     }
394     {
395       \typeout{No ~ footnotehyper ~ load}
396       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
397       \bool_set_false:N \l__enumext_footnotes_key_bool
398     }
399   }
```

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.

```

400 \bool_if:NTF \__enumext_hyperref_bool
401 {
402   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
403   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
404 }
405 {
406   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
407   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
408 }
409 }

```

(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: `__enumext_newlabel_arg_one_tl`

#2: `__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.

```

410 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
411 {
412   \protected@write \@auxout { }
413   {
414     \token_to_str:N \newlabel {#1}
415     {
416       {#2}
417       \bool_if:NT \__enumext_hyperref_bool
418       { { \thepage } {#2} {#1} }
419       { }
420     }
421   }
422   \__enumext_hypertarget:nn {#1} { }
423   \__enumext_phantomsection:
424 }

```

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

11.9 Definition of counters

`__enumext_define_counters:Nn` 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 `__enumext_counter_X_tl` for “store” the counter’s name.

#2: The counter’s name.

```

425 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
426 {
427   \cs_if_exist:cTF { c@ #2 }
428   { \msg_fatal:nnn { enumext } { counters } { #2 } }
429   {
430     \tl_set:Nn #1 { #2 }
431     \newcounter { #2 }
432   }
433 }

```

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

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

```

enumXi 434 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv 435 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi 436 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii 437 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 438 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
439 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
440 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
441 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

11.10 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.

```
442 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
443 {
444   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
445   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
446 }
447 \__enumext_register_counter_style:Nn \arabic { 0 }
448 \__enumext_register_counter_style:Nn \Alph { M }
449 \__enumext_register_counter_style:Nn \alph { m }
450 \__enumext_register_counter_style:Nn \Roman { VIII }
451 \__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.

```
452 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
453 {
454   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
455   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
456 }
457 \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.

```
458 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
459 {
460   \tl_clear_new:N #1
461   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
462   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
463   \tl_map_inline:Nn \g__enumext_counter_styles_tl
464   {
465     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
466     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
467     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
468   }
469   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
470   { \tl_use:N \g__enumext_widest_label_tl }
471   \tl_set_eq:cN { the #2 } #1
472 }
473 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

11.11 Setting keys associated with label

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
474 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
475 {
476   \keys_define:nn { enumext / #1 }
477   {
478     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
479     font      .value_required:n = true,
480     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
481     labelsep  .initial:n   = {0.3333em},
482     labelsep  .value_required:n = true,
483     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
484     labelwidth .value_required:n = true,
```

```

485     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
486     wrap-label .initial:n = {##1},
487     wrap-label .value_required:n = true,
488     wrap-label* .code:n = {
489         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
490         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
491     },
492     wrap-label* .value_required:n = true,
493 }
494 }
495 \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.

```

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 =
502             {
503                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
504                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
505             },
506         align / right .code:n =
507             {
508                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
509                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
510             },
511         align / center .code:n =
512             {
513                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
514                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
515             },
516         align .initial:n = left,
517         align .value_required:n = true,
518     }
519 }
520 \clist_map_inline:nn
521 {
522     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
523 }
524 { \__enumext_tmp:nn #1 }

525 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
526 {
527     \keys_define:nn { enumext / #1 }
528     {
529         align .choice:,
530         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
531         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
532         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
533         align .initial:n = left,
534         align .value_required:n = true,
535     }
536 }
537 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

11.12 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.

11.12.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl 538 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 539 {
\l__enumext_label_iii_tl 540   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 541   {
542     label .code:n = {
543       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
544       { l__enumext_counter_#2_tl } {##1}
545       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
546       \l__enumext_current_widest_dim
547     },
548     label .initial:n = #3,
549     label .value_required:n = true,
550     ref .code:n = \__enumext_standar_ref:n {##1},
551     ref .value_required:n = true,
552   }
553 }
554 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
555 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
556 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
557 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

`__enumext_standar_ref:n` The `__enumext_standar_ref:n` first we will pass the key argument to `__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 `__enumext_ref_the_count_tl` and we will execute the function `__enumext_regex_counter_style:` which will return the modified `__enumext_ref_key_arg_tl` and we make the value of `__enumext_ref_the_count_tl` the same as that `__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `__enumext_renew_the_count_X_tl` with the renewed command.

```

558 \cs_new_protected:Npn \__enumext_standar_ref:n #1
559 {
560   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
561   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
562   {
563     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
564   }
565   {
566     \tl_set_eq:Nc
567     \__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
568     \__enumext_regex_counter_style:
569     \tl_set_eq:Nc
570     \__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
571     \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
572     {
573       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
574       { \exp_not:V \__enumext_ref_key_arg_tl }
575     }
576   }
577 }

```

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

```

578 \cs_new_protected:Nn \__enumext_standar_ref:
579 {
580   \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
581   {
582     \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
583   }
584 }

```

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

11.12.2 Define and set label and ref keys for enumext* and keyans* environments

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

```

\l__enumext_label_vii_tl 585 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 586 {

```

```

587 \keys_define:nn { enumext / #1 }
588 {
589   label .code:n = {
590     \__enumext_label_style:cvn { l__enumext_label_#2_tl }
591     { l__enumext_counter_#2_tl } {##1}
592     \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
593     \l__enumext_current_widest_dim
594   },
595   label .initial:n = #3,
596   label .value_required:n = true,
597   ref .code:n = \__enumext_starred_ref:n {##1},
598   ref .value_required:n = true,
599 }
600 }
601 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
602 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

603 \cs_new_protected:Npn \__enumext_starred_ref:n #1
604 {
605   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
606   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
607   {
608     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
609     {
610       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
611     }
612     {
613       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
614       \__enumext_regex_counter_style:
615       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
616       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
617       {
618         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
619         { \exp_not:V \l__enumext_ref_key_arg_tl }
620       }
621     }
622   }
623   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
624   {
625     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
626     {
627       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
628     }
629     {
630       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
631       \__enumext_regex_counter_style:
632       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
633       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
634       {
635         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
636         { \exp_not:V \l__enumext_ref_key_arg_tl }
637       }
638     }
639   }
640 }

```

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.

```

641 \cs_new_protected:Npn \__enumext_starred_ref:
642 {
643   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
644   {
645     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
646     {
647       \tl_use:N \l__enumext_renew_the_count_vii_tl
648     }
649   }

```



```

650 \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
651 {
652   \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
653   {
654     \tl_use:N \l__enumext_renew_the_count_viii_tl
655   }
656 }
657 }

```

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

11.12.3 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.

```

label \l__enumext_label_v_tl
ref   \l__enumext_label_vi_tl
658 \keys_define:nn { enumext / keyans }
659 {
660   label .code:n = {
661     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
662     { \l__enumext_counter_v_tl } {#1}
663     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
664     \l__enumext_current_widest_dim
665     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
666     { \l__enumext_counter_vi_tl } {#1}
667     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
668     \l__enumext_current_widest_dim
669   },
670   label .initial:n = \Alph*,
671   label .value_required:n = true,
672   ref .code:n = \__enumext_keyans_ref:n {#1},
673   ref .value_required:n = true,
674 }

```

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

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

```

675 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
676 {
677   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
678   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
679   {
680     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
681   }
682   {
683     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
684     \__enumext_regex_counter_style:
685     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
686     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
687     {
688       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
689       { \exp_not:V \l__enumext_ref_key_arg_tl }
690     }
691   }
692 }

```

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

```

693 \cs_new_protected:Nn \__enumext_keyans_ref:
694 {
695   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
696   {
697     \tl_use:N \l__enumext_renew_the_count_v_tl
698   }
699 }

```

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

11.13 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.

```
700 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
701 {
702   \__enumext_if_is_int:nTF { #3 }
703   {
704     \int_set:Nn #2 {#3}
705   }
706   {
707     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
708     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
709     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
710     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
711   }
712 }
713 \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.

```
714 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
715 {
716   \__enumext_if_is_int:nTF {#4}
717   {
718     \setcounter{enumX#1} { #4 }
719   }
720   {
721     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
722     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
723     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
724     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
725   }
726   \__enumext_label_width_by_box:cv
727   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
728 }
729 \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`, `enumext*`, `keyans` and `keyans*` environments.

```
730 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
731 {
732   \keys_define:nn { enumext / #1 }
733   {
734     start .code:n = {
735       \__enumext_start_from:ccn
736       { \l__enumext_label_#2_tl }
737       { \l__enumext_start_#2_int } {##1}
738     },
739     start .initial:n = 1,
740     widest .code:n = {
741       \__enumext_widest_from:nccn {#2}
742       { \l__enumext_label_#2_tl }
743       { \l__enumext_labelwidth_#2_dim } {##1}
744     },
```

```

745         widest .value_required:n = true,
746         start .value_required:n = true,
747     }
748 }
749 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.14 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
750 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
751 {
752     \keys_define:nn { enumext / #1 }
753     {
754         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
755         topsep .initial:n = {#3},
756         topsep .value_required:n = true,
757         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
758         partopsep .initial:n = {#4},
759         partopsep .value_required:n = true,
760         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
761         parsep .initial:n = {#5},
762         parsep .value_required:n = true,
763         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
764         itemsep .initial:n = {#6},
765         itemsep .value_required:n = true,
766         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
767         noitemsep .value_forbidden:n = true,
768         nosep .meta:n = {
769             itemsep = 0pt, parsep = 0pt,
770             topsep = 0pt, partopsep = 0pt,
771         },
772         nosep .value_forbidden:n = true,
773     }
774 }

```

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

```

775 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
776 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
777 { 4.0pt plus 2.0pt minus 1.0pt }
778 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
779 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
780 { 2.0pt plus 1.0pt minus 1.0pt }
781 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
782 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
783 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
784 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
785 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
786 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
787 { 2.0pt plus 1.0pt minus 1.0pt }
788 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
789 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
790 { 4.0pt plus 2.0pt minus 1.0pt }
791 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
792 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
793 { 2.0pt plus 1.0pt minus 1.0pt }

```

(End of definition for topsep and others.)

11.15 Setting keys for horizontal spaces

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

```

794 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
795 {
796     \keys_define:nn { enumext / #1 }
797     {
798         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
799         itemindent .value_required:n = true,
800         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },

```

```

801     rightmargin .value_required:n = true,
802     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
803     listparindent .value_required:n = true,
804     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
805     list-offset .value_required:n = true,
806     list-indent .code:n =
807         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
808         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
809     list-indent .value_required:n = true,
810 }
811 }
812 \clist_map_inline:Nn \__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.

```

813 \cs_set_protected:Npn \__enumext_tmp:n #1
814 {
815     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
816 }
817 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

11.15.1 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.

```

818 \cs_set_protected:Nn \__enumext_fake_item:
819 {
820     \dim_compare:nNnT
821     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
822     >
823     { \c_zero_dim }
824     {
825         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
826         {
827             \exp_not:N \mode_leave_vertical:
828             \exp_not:n { \skip_horizontal:n }
829             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
830             \ignorespaces
831         }
832     }
833 }
834 \cs_set_protected:Nn \__enumext_keyans_fake_item:
835 {
836     \dim_compare:nNnT
837     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
838     {
839         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
840         {
841             \exp_not:N \mode_leave_vertical:
842             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
843         }
844     }
845 }
846 \cs_set_protected:Nn \__enumext_fake_item_vii:
847 {
848     \dim_compare:nNnT
849     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
850     {
851         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
852         {
853             \exp_not:N \mode_leave_vertical:
854             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
855         }
856     }
857 }
858 \cs_set_protected:Nn \__enumext_fake_item_viii:
859 {
860     \dim_compare:nNnT

```

```

861     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
862     {
863         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
864         {
865             \exp_not:N \mode_leave_vertical:
866             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
867         }
868     }
869 }

```

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

11.16 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.

```

870 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
871 {
872     \keys_define:nn { enumext / #1 }
873     {
874         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
875         show-length .initial:n = false,
876     }
877 }
878 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.17 Setting before, after and first keys

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

```

879 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
880 {
881     \keys_define:nn { enumext / #1 }
882     {
883         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
884         before .value_required:n = true,
885         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
886         before* .value_required:n = true,
887         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
888         after .value_required:n = true,
889         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
890         first .value_required:n = true,
891     }
892 }
893 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.17.1 Functions for before, after and first keys in enumext

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

```

894 \cs_new_protected:Nn \__enumext_before_args_exec:
895 {
896     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
897 }

```

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

```

898 \cs_new_protected:Nn \__enumext_before_keys_exec:
899 {
900     \tl_use:c { \l__enumext_before_no_starred_key_ \__enumext_level: _tl }
901 }

```

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

```

902 \cs_new_protected:Nn \__enumext_after_stop_list:
903 {
904     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
905 }

```

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

```

906 \cs_new_protected:Nn \__enumext_after_args_exec:
907 {
908     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
909 }

```

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

11.17.2 Functions for before, after and first keys in keyans

```

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

```

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

```

910 \cs_new_protected:Nn \__enumext_before_args_exec_v:
911 {
912     \tl_use:N \l__enumext_before_starred_key_v_tl
913 }

```

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

```

914 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
915 {
916     \tl_use:N \l__enumext_before_no_starred_key_v_tl
917 }

```

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

```

918 \cs_new_protected:Nn \__enumext_after_stop_list_v:
919 {
920     \tl_use:N \l__enumext_after_stop_list_v_tl
921 }

```

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

```

922 \cs_new_protected:Nn \__enumext_after_args_exec_v:
923 {
924     \tl_use:N \l__enumext_after_list_args_v_tl
925 }

```

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

11.17.3 Functions for before, after and first keys in enumext* and keyans*

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:

```

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

```

926 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
927 {
928     \tl_use:N \l__enumext_before_starred_key_vii_tl
929 }
930 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
931 {
932     \tl_use:N \l__enumext_before_starred_key_viii_tl
933 }

```

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

```

934 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
935 {
936     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
937 }
938 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
939 {

```



```

940     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
941   }

```

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

```

942 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
943 {
944   \tl_use:N \l__enumext_after_stop_list_vii_tl
945 }
946 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
947 {
948   \tl_use:N \l__enumext_after_stop_list_viii_tl
949 }

```

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`.

```

950 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
951 {
952   \tl_use:N \l__enumext_after_list_args_vii_tl
953 }
954 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
955 {
956   \tl_use:N \l__enumext_after_list_args_viii_tl
957 }

```

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

11.18 Setting keys for multicol 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`, `enumext*`, `keyans` and `keyans*` environments.

```

958 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
959 {
960   \keys_define:nn { enumext / #1 }
961   {
962     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
963     mini-env .value_required:n = true,
964     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
965     mini-sep .initial:n = 0.3333em,
966     mini-sep .value_required:n = true,
967     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
968     columns-sep .value_required:n = true,
969     columns .int_set:c = { l__enumext_columns_#2_int },
970     columns .initial:n = 1,
971     columns .value_required:n = true,
972   }
973 }
974 \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 command `\miniright` is not available, so we will add the keys `mini-right` and `mini-right*` to implement support for `minipage` environment.

```

975 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
976 {
977   \keys_define:nn { enumext / #1 }
978   {
979     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
980     mini-right .value_required:n = true,
981     mini-right* .code:n = {
982       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
983       \keys_set:nn { enumext / #1 } { miniright = {##1} }
984     },
985     mini-right* .value_required:n = true,
986   }
987 }
988 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.19 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.

11.19.1 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 `__enumext_multicol_above_X_skip` and `__enumext_multicol_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

989 \cs_new_protected:Nn \__enumext_multi_set_vskip:
990 {
991   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip {
992     {
993       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
994       }
995     }
996   \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip {
997     {
998       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
999     }
1000   }

```

(End of definition for `__enumext_multi_set_vskip:`)

`__enumext_add_pre_parsep:`

The function `__enumext_add_pre_parsep:` “*adjusted*” the value of `__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*.

```

1001 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1002 {
1003   \int_case:nn { \__enumext_level_int }
1004   {
1005     { 2 } {
1006       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip } {
1007         {
1008           \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
1009         }
1010       }
1011     { 3 } {
1012       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip } {
1013         {
1014           \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip
1015         }
1016       }
1017     { 4 } {
1018       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip } {
1019         {
1020           \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip
1021         }
1022       }

```

```

1023     }
1024 }

```

(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 TeX is in *(horizontal mode)* or *(vertical mode)*.

```

1025 \cs_new_protected:Nn \__enumext_multi_addvspace:
1026 {
1027   \__enumext_multi_set_vskip:
1028   \mode_if_vertical:T
1029   {
1030     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
1031     {
1032       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1033     }
1034     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
1035     {
1036       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1037     }
1038   }
1039   \par\nopagebreak
1040   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
1041 }

```

(End of definition for `__enumext_multi_addvspace:`)

11.19.2 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:`

```

1042 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1043 {
1044   \skip_set:Nn \__enumext_multicols_above_v_skip
1045   {
1046     \__enumext_topsep_v_skip
1047   }
1048   \skip_set:Nn \__enumext_multicols_below_v_skip
1049   {
1050     \__enumext_topsep_v_skip
1051   }
1052 }
1053 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1054 {
1055   \__enumext_keyans_multi_set_vskip:
1056   \mode_if_vertical:T
1057   {
1058     \skip_add:Nn \__enumext_multicols_above_v_skip
1059     {
1060       \skip_use:N \__enumext_partopsep_v_skip
1061     }
1062     \skip_add:Nn \__enumext_multicols_below_v_skip
1063     {
1064       \skip_use:N \__enumext_partopsep_v_skip
1065     }
1066   }
1067   \par\nopagebreak
1068   \addvspace{ \__enumext_multicols_above_v_skip }
1069 }

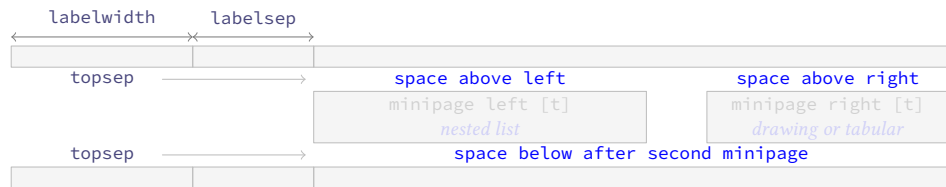
```

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

11.20 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.

11.20.1 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`.

```

1070 \cs_new_protected:Nn __enumext_mini_set_vskip:
1071 {
1072   \int_compare:nNnTF
1073     { \int_use:c { l__enumext_columns_ __enumext_level: _int } } > { 1 }
1074     {

```

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

```

1075   \skip_if_eq:nNnTF
1076     { \skip_use:c { l__enumext_topsep_ __enumext_level: _skip } } { \c_zero_skip }
1077   {
1078     \skip_set:Nn \l__enumext_minipage_left_skip
1079     {
1080       -0.150\box_dp:N \strutbox
1081     }
1082     \skip_set:Nn \l__enumext_minipage_right_skip
1083     {
1084       0.695\box_dp:N \strutbox
1085     }
1086     \skip_set:Nn \l__enumext_minipage_after_skip
1087     {
1088       \box_dp:N \strutbox
1089     }
1090     __enumext_zero_parsep:
1091   }
1092   {
1093     \skip_set:Nn \l__enumext_minipage_left_skip
1094     {
1095       \skip_use:c { l__enumext_topsep_ __enumext_level: _skip }
1096     }
1097     \skip_set:Nn \l__enumext_minipage_right_skip
1098     {
1099       0.695\box_dp:N \strutbox
1100     }
1101     \skip_set:Nn \l__enumext_minipage_after_skip
1102     {
1103       1.85\box_dp:N \strutbox
1104       + \skip_use:c { l__enumext_topsep_ __enumext_level: _skip }
1105     }

```

```

1106     }
1107   }
1108   {

```

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.

```

1109     \skip_if_eq:nnTF
1110     { \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip } } { \c_zero_skip }
1111     {
1112       \skip_set:Nn \l__enumext_minipage_left_skip
1113       {
1114         0.5\box_dp:N \strutbox
1115         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1116       }
1117       \skip_set:Nn \l__enumext_minipage_right_skip
1118       {
1119         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1120       }
1121       \skip_set:Nn \l__enumext_minipage_after_skip
1122       {
1123         1.6\box_dp:N \strutbox
1124       }
1125     }
1126   {
1127     \skip_set:Nn \l__enumext_minipage_left_skip
1128     {
1129       0.5875\box_dp:N \strutbox
1130       - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1131     }
1132     \skip_set:Nn \l__enumext_minipage_right_skip
1133     {
1134       + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1135       + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1136     }
1137     \skip_set:Nn \l__enumext_minipage_after_skip
1138     {
1139       0.325\box_dp:N \strutbox
1140       + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1141     }
1142   }
1143 }
1144 }

```

(End of definition for `\l__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.

```

1145 \cs_new_protected:Nn \__enumext_zero_parsep:
1146 {
1147   \int_case:nn { \l__enumext_level_int }
1148   {
1149     { 2 } {
1150       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1151       {
1152         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1153       }
1154     }
1155     { 3 } {
1156       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1157       {
1158         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1159       }
1160     }
1161     { 4 } {
1162       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1163       {
1164         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1165       }
1166     }

```

```

1167     }
1168 }

```

(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 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*.

```

1169 \cs_new_protected:Nn \__enumext_mini_addvspace:
1170 {
1171   \__enumext_mini_set_vskip:
1172   \mode_if_vertical:T
1173   {
1174     \skip_add:Nn \l__enumext_minipage_left_skip
1175     {
1176       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1177     }
1178     \skip_add:Nn \l__enumext_minipage_after_skip
1179     {
1180       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1181     }
1182   }
1183   \par\nopagebreak
1184   \addvspace { \l__enumext_minipage_left_skip }
1185 }

```

(End of definition for `__enumext_mini_addvspace:`.)

11.20.2 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`.

```

1186 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1187 {
1188   \skip_zero_new:N \l__enumext_minipage_after_skip
1189   \skip_zero_new:N \l__enumext_minipage_left_skip
1190   \skip_zero_new:N \l__enumext_minipage_right_skip
1191   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1192   {
1193     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1194     {
1195       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1196       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1197       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1198       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1199       {
1200         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1201       }
1202     }
1203     {
1204       \skip_set:Nn \l__enumext_minipage_left_skip
1205       {
1206         \skip_use:N \l__enumext_topsep_v_skip
1207       }
1208       \skip_set:Nn \l__enumext_minipage_right_skip
1209       {
1210         0.705\box_dp:N \strutbox
1211       }
1212       \skip_set:Nn \l__enumext_minipage_after_skip
1213       {
1214         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1215       }
1216     }
1217   }
1218   {
1219     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1220     {
1221       \skip_set:Nn \l__enumext_minipage_left_skip
1222       {

```



```

1223         0.5\box_dp:N \strutbox
1224         + \l__enumext_partopsep_v_skip
1225     }
1226     \skip_set:Nn \l__enumext_minipage_right_skip
1227     {
1228         \l__enumext_partopsep_v_skip
1229     }
1230     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1231 }
1232 {
1233     \skip_set:Nn \l__enumext_minipage_left_skip
1234     {
1235         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1236     }
1237     \skip_set:Nn \l__enumext_minipage_right_skip
1238     {
1239         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1240     }
1241     \skip_set:Nn \l__enumext_minipage_after_skip
1242     {
1243         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1244     }
1245 }
1246 }
1247 }

```

(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 \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`.

```

1248 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1249 {
1250     \l__enumext_keyans_mini_set_vskip:
1251     \mode_if_vertical:T
1252     {
1253         \skip_add:Nn \l__enumext_minipage_left_skip
1254         {
1255             \l__enumext_partopsep_v_skip
1256         }
1257         \skip_add:Nn \l__enumext_minipage_after_skip
1258         {
1259             \l__enumext_partopsep_v_skip
1260         }
1261     }
1262     \par\nopagebreak
1263     \addvspace { \l__enumext_minipage_left_skip }
1264 }

```

(End of definition for `\l__enumext_keyans_mini_addvspace:`)

11.20.3 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*`.

```

1265 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1266 {
1267     \skip_zero_new:N \l__enumext_minipage_left_skip
1268     \skip_gzero_new:N \g__enumext_minipage_right_skip
1269     \skip_gzero_new:N \g__enumext_minipage_after_skip
1270     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1271     {
1272         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1273         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1274     }
1275     {
1276         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1277         \skip_gset:Nn \g__enumext_minipage_right_skip

```

```

1278         {
1279             \l__enumext_topsep_vii_skip
1280         }
1281         \skip_gset:Nn \g__enumext_minipage_after_skip
1282         {
1283             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1284         }
1285     }
1286 }
1287 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1288 {
1289     \skip_zero_new:N \l__enumext_minipage_after_skip
1290     \skip_zero_new:N \l__enumext_minipage_left_skip
1291     \skip_zero_new:N \l__enumext_minipage_right_skip
1292     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1293     {
1294         \skip_set:Nn \l__enumext_minipage_left_skip
1295         {
1296             0.5\box_dp:N \strutbox
1297         }
1298         \skip_set:Nn \l__enumext_minipage_right_skip
1299         {
1300             \l__enumext_partopsep_viii_skip
1301         }
1302         \skip_set:Nn \l__enumext_minipage_after_skip
1303         {
1304             1.6\box_dp:N \strutbox
1305         }
1306     }
1307     {
1308         \skip_set:Nn \l__enumext_minipage_left_skip
1309         {
1310             0.5875\box_dp:N \strutbox
1311         }
1312         \skip_set:Nn \l__enumext_minipage_right_skip
1313         {
1314             \l__enumext_topsep_viii_skip
1315         }
1316         \skip_set:Nn \l__enumext_minipage_after_skip
1317         {
1318             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1319         }
1320     }
1321 }

```

(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 \TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1322 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1323 {
1324     \__enumext_mini_set_vskip_vii:
1325     \par\nopagebreak
1326     \addvspace { \l__enumext_minipage_left_skip }
1327 }
1328 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1329 {
1330     \__enumext_mini_set_vskip_viii:
1331     \par\nopagebreak
1332     \addvspace { \l__enumext_minipage_left_skip }
1333 }

```

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

11.20.4 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 argument* ‘`*`’ inhibits the use

of `\centering` command i.e. the usual \LaTeX 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.

```

1334 \NewDocumentCommand \miniright { s }
1335 {
1336   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1337   {
1338     \msg_error:nnn { enumext } { wrong-miniright-place }
1339   }
1340   \int_compare:nNt { \l__enumext_level_int } = { 0 }
1341   {
1342     \msg_error:nnn { enumext } { wrong-miniright-place }
1343   }
1344   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1345   {
1346     \__enumext_keyans_mini_right_cmd:n {#1}
1347   }
1348   { \__enumext_mini_right_cmd:n {#1} }
1349 }

```

(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* ‘*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\l__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.

```

1350 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1351 {
1352   \dim_compare:nNtF
1353   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1354   {
1355     \__enumext_multicols_stop:
1356     \end{__enumext_mini_env*}
1357     \hfill
1358     \begin{__enumext_mini_env*}
1359     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1360     \par\addvspace { \l__enumext_minipage_right_skip }
1361     \bool_if:nF {#1}
1362     {
1363       \centering
1364     }
1365     \int_gzero:N \g__enumext_minipage_stat_int
1366   }
1367   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1368 }

```

(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* ‘*’ of the `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1369 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1370 {
1371   \dim_compare:nNtF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1372   {
1373     \__enumext_keyans_multicols_stop:
1374     \end{__enumext_mini_env*}
1375     \hfill
1376     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1377     \par\addvspace { \l__enumext_minipage_right_skip }
1378     \bool_if:nF {#1}
1379     {
1380       \centering
1381     }

```

```

1382         \int_gzero:N \g__enumext_minipage_stat_int
1383     }
1384     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1385 }

```

(End of definition for `__enumext_keyans_mini_right_cmd:n`.)

11.21 Setting above and below keys

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

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

```

above* 1386 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below* 1387 {
below* 1388   \keys_define:nn { enumext / #1 }
1389   {
1390     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1391     above .value_required:n = true,
1392     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1393                     \keys_set:nn { enumext / #1 } { above = {##1} },
1394     above* .value_required:n = true,
1395     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1396     below .value_required:n = true,
1397     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1398                     \keys_set:nn { enumext / #1 } { below = {##1} },
1399     below* .value_required:n = true,
1400   }
1401 }
1402 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.21.1 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.

```

1403 \cs_new_protected:Nn \__enumext_vspace_above:
1404 {
1405   \skip_if_eq:nnF
1406   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1407   {
1408     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1409     {
1410       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1411     }
1412     {
1413       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1414     }
1415   }
1416 }

```

(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.

```

1417 \cs_new_protected:Nn \__enumext_vspace_below:
1418 {
1419   \skip_if_eq:nnF
1420   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1421   {
1422     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1423     {
1424       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1425     }
1426     {
1427       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1428     }
1429   }
1430 }

```

(End of definition for `__enumext_vspace_below:`)

11.21.2 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.

```

1431 \cs_new_protected:Nn \__enumext_vspace_above_v:
1432 {
1433   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1434   {
1435     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1436     {
1437       \vspace*{ \l__enumext_vspace_above_v_skip }
1438     }
1439     { \vspace { \l__enumext_vspace_above_v_skip } }
1440   }
1441 }

```

(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.

```

1442 \cs_new_protected:Nn \__enumext_vspace_below_v:
1443 {
1444   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1445   {
1446     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1447     {
1448       \vspace*{ \l__enumext_vspace_below_v_skip }
1449     }
1450     { \vspace { \l__enumext_vspace_below_v_skip } }
1451   }
1452 }

```

(End of definition for `__enumext_vspace_below_v:`)

11.21.3 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:`

```

1453 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1454 {
1455   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1456   {
1457     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1458     {
1459       \vspace*{ \l__enumext_vspace_above_vii_skip }
1460     }
1461     { \vspace { \l__enumext_vspace_above_vii_skip } }
1462   }
1463 }
1464 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1465 {
1466   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1467   {
1468     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1469     {
1470       \vspace*{ \l__enumext_vspace_above_viii_skip }
1471     }
1472     { \vspace { \l__enumext_vspace_above_viii_skip } }
1473   }
1474 }

```

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

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

`__enumext_vspace_below_viii:`

```

1475 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1476 {
1477   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1478   {
1479     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1480     {

```

```

1481         \vspace*{ \l__enumext_vspace_below_vii_skip }
1482     }
1483     { \vspace { \l__enumext_vspace_below_vii_skip } }
1484 }
1485 }
1486 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1487 {
1488     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1489     {
1490         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1491         {
1492             \vspace*{ \l__enumext_vspace_below_viii_skip }
1493         }
1494         { \vspace { \l__enumext_vspace_below_viii_skip } }
1495     }
1496 }

```

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

11.22 Setting series, resume and resume* keys

The `series` key is responsible for the whole process of the `resume` and `resume*` keys. The idea behind this is to be able to absorb the $\langle keys \rangle$ passed to the optional argument of the “first level” of the environments `enumext` and `enumext*`, but, discarding some specific $\langle keys \rangle$. 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*
1497 \cs_set_protected:Npn \__enumext_tmp:n #1
1498 {
1499     \keys_define:nn { enumext / #1 }
1500     {
1501         series .str_set:N = \l__enumext_series_str,
1502         series .value_required:n = true,
1503         resume .code:n = \__enumext_resume_series:n {##1},
1504         resume* .code:n = \__enumext_resume_starred:,
1505         resume* .value_forbidden:n = true,
1506     }
1507 }
1508 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {##1} }

```

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

11.22.1 Internal functions for series key

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

```

1509 \cs_new:Npn \__enumext_filter_series:n #1
1510 {
1511     \use:e
1512     {
1513         \keyval_parse:NNn
1514         \__enumext_filter_series_key:n
1515         \__enumext_filter_series_pair:nn {##1}
1516     }
1517 }

```

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

```

1518 \cs_new:Npn \__enumext_filter_series_key:n #1
1519 {
1520     \str_case:nnF {##1}
1521     {
1522         { resume } {}
1523         { resume* } {}
1524     }
1525     { , { \exp_not:n {##1} } }
1526 }

```

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

```

1527 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1528 {
1529     \str_case:nnF {##1}

```



```

1530     {
1531         { series } {}
1532         { resume } {}
1533         { start } {}
1534         { save-ans } {}
1535         { save-key } {}
1536     }
1537     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1538 }

```

(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 (§11.33) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.36).

```

1539 \cs_new_protected:Npn \__enumext_parse_series:n #1
1540 {
1541     \str_if_empty:NTF \l__enumext_series_str
1542     {
1543         \bool_if:NF \l__enumext_resume_active_bool
1544         {
1545             \__enumext_resume_last:n {#1}
1546         }
1547     }
1548     {
1549         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1550         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1551         { \__enumext_filter_series:n {#1} }
1552         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1553         {
1554             \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1555         }
1556     }
1557 }

```

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.

```

1558 \cs_new_protected:Npn \__enumext_resume_last:n #1
1559 {
1560     \bool_if:NT \l__enumext_standar_first_bool
1561     {
1562         \tl_gclear:N \g__enumext_standar_series_tl
1563         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1564     }
1565     \bool_if:NT \l__enumext_starred_first_bool
1566     {
1567         \tl_gclear:N \g__enumext_starred_series_tl
1568         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1569     }
1570 }

```

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

11.22.2 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 `__enumext_series_str` and `__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments, the integer variable with `__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§11.33) and the `enumext*` environment definition (§11.36).

```

1571 \cs_new_protected:Nn \__enumext_resume_save_counter:
1572 {
1573   \bool_if:NT \g__enumext_standar_bool
1574   {
1575     \tl_if_empty:NF \__enumext_series_str
1576     {
1577       \int_gset_eq:cN
1578       { g__enumext_series_ \__enumext_series_str_int } \value{enumXi}
1579     }
1580     \tl_if_empty:NTF \__enumext_resume_name_tl
1581     {
1582       \str_if_empty:NT \__enumext_series_str
1583       {
1584         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1585       }
1586     }
1587     {
1588       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1589       {
1590         \int_gset_eq:cN
1591         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXi}
1592       }
1593     }
1594     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1595     {
1596       \int_gset_eq:cN
1597       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXi}
1598     }
1599   }
1600   \bool_if:NT \g__enumext_starred_bool
1601   {
1602     \tl_if_empty:NF \__enumext_series_str
1603     {
1604       \int_gset_eq:cN
1605       { g__enumext_series_ \__enumext_series_str_int } \value{enumXvii}
1606     }
1607     \tl_if_empty:NTF \__enumext_resume_name_tl
1608     {
1609       \str_if_empty:NT \__enumext_series_str
1610       {
1611         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1612       }
1613     }
1614     {
1615       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1616       {
1617         \int_gset_eq:cN
1618         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXvii}
1619       }
1620     }
1621     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1622     {
1623       \int_gset_eq:cN
1624       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXvii}
1625     }
1626   }
1627 }

```

(End of definition for `__enumext_resume_save_counter:`)

11.22.3 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.

```

1628 \cs_new_protected:Npn \__enumext_resume_series:n #1
1629 {
1630   \tl_if_empty:NTF {#1}
1631   {
1632     \__enumext_resume_counter:n { }
1633   }
1634   {
1635     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1636     {
1637       \__enumext_resume_counter:n {#1}
1638       \bool_if:NT \g__enumext_standar_bool
1639       {
1640         \keys_set:nv { enumext / level-1 }
1641         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1642       }
1643       \bool_if:NT \g__enumext_starred_bool
1644       {
1645         \keys_set:nv { enumext / enumext* }
1646         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1647       }
1648     }
1649     {
1650       \bool_if:NT \g__enumext_standar_bool
1651       {
1652         \msg_error:nnn { enumext } { unknown-series } {#1}
1653       }
1654       \bool_if:NT \g__enumext_starred_bool
1655       {
1656         \msg_error:nnn { enumext } { unknown-series } {#1}
1657       }
1658     }
1659   }
1660 }

```

(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`.

```

1661 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1662 {
1663   \bool_set_true:N \l__enumext_resume_active_bool
1664   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1665   \tl_if_empty:NTF \l__enumext_resume_name_tl
1666   {
1667     \__enumext_resume_counter:
1668   }
1669   {
1670     \__enumext_resume_counter_series:
1671   }
1672   \__enumext_resume_counter_save_ans:
1673 }

```

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.

```

1674 \cs_new_protected:Nn \__enumext_resume_counter:
1675 {
1676   \bool_if:NT \g__enumext_standar_bool
1677   {
1678     \int_gincr:N \g__enumext_resume_int
1679     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1680   }
1681   \bool_if:NT \g__enumext_starred_bool
1682   {
1683     \int_gincr:N \g__enumext_resume_vii_int

```

```

1684         \int_set_eq:Nn \l__enumext_start_vii_int \g__enumext_resume_vii_int
1685     }
1686 }

```

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.

```

1687 \cs_new_protected:Nn \__enumext_resume_counter_series:
1688 {
1689     \bool_if:NT \g__enumext_standar_bool
1690     {
1691         \int_set:Nn \l__enumext_start_i_int
1692         {
1693             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1694         }
1695     }
1696     \bool_if:NT \g__enumext_starred_bool
1697     {
1698         \int_set:Nn \l__enumext_start_vii_int
1699         {
1700             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1701         }
1702     }
1703 }

```

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.

```

1704 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1705 {
1706     \bool_lazy_and:nnT
1707     { \bool_if_p:N \l__enumext_standar_first_bool }
1708     { \bool_if_p:N \l__enumext_store_active_bool }
1709     {
1710         \int_set:Nn \l__enumext_start_i_int
1711         {
1712             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1713         }
1714     }
1715     \bool_lazy_and:nnT
1716     { \bool_if_p:N \l__enumext_starred_first_bool }
1717     { \bool_if_p:N \l__enumext_store_active_bool }
1718     {
1719         \int_set:Nn \l__enumext_start_vii_int
1720         {
1721             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1722         }
1723     }
1724 }

```

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

11.22.4 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.

```

1725 \cs_new_protected:Nn \__enumext_resume_starred:
1726 {
1727     \bool_if:NT \g__enumext_standar_bool
1728     {
1729         \tl_if_empty:NF \g__enumext_standar_series_tl
1730         {
1731             \__enumext_resume_counter:n { }
1732             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1733         }
1734     }
1735     \bool_if:NT \g__enumext_starred_bool
1736     {
1737         \tl_if_empty:NF \g__enumext_starred_series_tl
1738         {

```

```

1739         \__enumext_resume_counter:n { }
1740         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1741     }
1742 }
1743 }

```

(End of definition for __enumext_resume_starred:.)

11.23 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.

11.23.1 Setting save-ans key

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

```

1744 \cs_set_protected:Npn \__enumext_tmp:n #1
1745 {
1746     \keys_define:nn { enumext / #1 }
1747     {
1748         save-ans .code:n = \__enumext_storing_set:n {##1},
1749         save-ans .value_required:n = true,
1750     }
1751 }
1752 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.23.2 Internal functions for save-ans key

`__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:`.

```

1753 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1754 {
1755     \msg_term:nnVV { enumext } { save-ans-log }
1756     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1757 }
1758 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1759 {
1760     \msg_term:nnVV { enumext } { save-ans-log-hook }
1761     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1762 }

```

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

`__enumext_storing_set:n` and `__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.

```

1763 \cs_new_protected:Npn \__enumext_storing_set:n #1
1764 {
1765     \tl_set:Nx \l__enumext_store_name_tl {#1}
1766     \tl_if_empty:NTF \l__enumext_store_name_tl
1767     {
1768         \bool_lazy_or:nnT
1769         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1770         {
1771             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1772         }
1773     }
1774     {
1775         \bool_lazy_or:nnT
1776         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1777         {
1778             \__enumext_start_save_ans_msg:
1779             \__enumext_storing_exec:
1780         }
1781     }
1782 }

```

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*`.

```

1783 \cs_new_protected:Nn \__enumext_storing_exec:
1784 {
1785   \bool_set_true:N \l__enumext_store_active_bool
1786   \bool_set_true:N \l__enumext_check_answers_bool
1787   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1788   \__enumext_scontents_anskey:V \l__enumext_store_name_tl
1789   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1790   {
1791     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1792     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1793   }
1794   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1795   {
1796     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1797     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1798   }
1799   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1800   {
1801     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1802     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1803   }
1804 }

```

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

11.23.3 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:

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

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

11.23.4 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
1805 \cs_set_protected:Npn \__enumext_tmp:n #1
1806 {
1807   \keys_define:nn { enumext / #1 }

```

```

1808     {
1809         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1810         check-ans .initial:n = false,
1811         check-ans .value_required:n = true,
1812         no-store .code:n = {
1813             \bool_set_false:N \l__enumext_check_answers_bool
1814             \bool_set_false:N \l__enumext_check_ans_key_bool
1815         },
1816         no-store .value_forbidden:n = true,
1817     }
1818 }
1819 \clist_map_inline:nn
1820 {
1821     level-1, level-2, level-3, level-4, enumext*
1822 }
1823 { \l__enumext_tmp:n {#1} }

```

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

11.23.5 Set-up check answer mechanism

`\l__enumext_check_ans_active:` The function `\l__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 `\l__enumext_check_ans_level:` only if “*true*”, i.e. the key `no-store` is not active.

```

1824 \cs_new_protected:Nn \l__enumext_check_ans_active:
1825 {
1826     \tl_if_empty:NF \l__enumext_store_name_tl
1827     {
1828         \bool_if:NT \l__enumext_check_answers_bool
1829         {
1830             \l__enumext_check_ans_level:
1831         }
1832     }
1833 }

```

The function `\l__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.

```

1834 \cs_new_protected:Nn \l__enumext_check_ans_level:
1835 {
1836     \int_case:nn { \l__enumext_level_int }
1837     {
1838         { 1 }{
1839             \bool_lazy_all:nT
1840             {
1841                 { \bool_if_p:N \g__enumext_starred_bool }
1842                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1843             }
1844             {
1845                 \int_gdecr:N \g__enumext_item_number_int
1846             }
1847         }
1848         { 2 }{
1849             \int_gdecr:N \g__enumext_item_number_int
1850         }
1851         { 3 }{
1852             \int_gdecr:N \g__enumext_item_number_int
1853         }
1854         { 4 }{
1855             \int_gdecr:N \g__enumext_item_number_int
1856         }
1857     }

```

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.

```

1858     \int_case:nn { \l__enumext_level_h_int }
1859     {
1860         { 1 }{
1861             \bool_lazy_all:nT
1862             {

```



```

1863         { \bool_if_p:N \g__enumext_standar_bool }
1864         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1865     }
1866     {
1867         \int_gdecr:N \g__enumext_item_number_int
1868     }
1869 }
1870 }
1871 }

```

(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.

```

1872 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1873 {
1874     \bool_lazy_and:nnT
1875     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1876     { \bool_if_p:N \g__enumext_standar_bool }
1877     {
1878         \bool_gset_true:N \g__enumext_check_ans_key_bool
1879     }
1880     \bool_lazy_and:nnT
1881     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1882     { \bool_if_p:N \g__enumext_starred_bool }
1883     {
1884         \bool_gset_true:N \g__enumext_check_ans_key_bool
1885     }
1886 }

```

(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:`.

```

1887 \cs_new_protected:Nn \__enumext_item_answer_diff:
1888 {
1889     \int_gset:Nn \g__enumext_item_answer_diff_int
1890     {
1891         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1892     }
1893 }

```

(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:`.

```

1894 \cs_new_protected:Nn \__enumext_check_ans_show:
1895 {
1896     \int_case:nn { \g__enumext_item_answer_diff_int }
1897     {
1898         { -1 } { \__enumext_check_ans_msg_less: }
1899         { 0 } { \__enumext_check_ans_msg_same_ok: }
1900         { 1 } { \__enumext_check_ans_msg_greater: }
1901     }
1902 }
1903 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1904 {
1905     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1906     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1907 }
1908 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1909 {
1910     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1911     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1912 }

```

```

1913 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1914 {
1915     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1916     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1917 }

```

(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:.

```

1918 \cs_new_protected:Nn \__enumext_check_ans_log:
1919 {
1920     \int_case:nn { \g__enumext_item_answer_diff_int }
1921     {
1922         { -1 } { \__enumext_check_ans_log_msg_less: }
1923         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1924         { 1 } { \__enumext_check_ans_log_msg_greater: }
1925     }
1926 }
1927 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1928 {
1929     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1930     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1931 }
1932 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1933 {
1934     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1935     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1936 }
1937 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1938 {
1939     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1940     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1941 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.23.6 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.

```

1942 \cs_new_protected:Nn \__enumext_execute_after_env:
1943 {
1944     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1945     {
1946         \tl_if_empty:NF \g__enumext_store_name_tl
1947         {
1948             \__enumext_stop_save_ans_msg:
1949             \__enumext_item_answer_diff:
1950             \__enumext_log_global_vars:
1951             \__enumext_log_answer_vars:
1952             \bool_if:NTF \g__enumext_check_ans_key_bool
1953             {
1954                 \__enumext_check_ans_show:
1955             }
1956             { \__enumext_check_ans_log: }
1957             \cs_undefine:N \__scontents_anskeyenv_env_begin:
1958             \cs_undefine:N \__scontents_anskeyenv_env_end:
1959         }
1960         \__enumext_reset_global_vars:
1961     }
1962 }

```

(End of definition for __enumext_execute_after_env:.)

11.23.7 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.

```

1963 \cs_new_protected:Npn \_enumext\_check\_starred\_cmd:n #1
1964 {
1965   \int_compare:nNtT
1966     { \g\_enumext\_check\_starred\_cmd\_int } = { 0 }
1967     {
1968       \msg_warning:nnnV
1969         { enumext } { missing-starred } { #1 } \l\_enumext\_check\_start\_line\_env\_tl
1970     }
1971   \int_compare:nNtT
1972     { \g\_enumext\_check\_starred\_cmd\_int } > { 1 }
1973     {
1974       \msg_warning:nnnV
1975         { enumext } { many-starred } { #1 } \l\_enumext\_check\_start\_line\_env\_tl
1976     }
1977   \int_gzero:N \g\_enumext\_check\_starred\_cmd\_int
1978   \tl_clear:N \l\_enumext\_check\_start\_line\_env\_tl
1979 }

```

(End of definition for _enumext_check_starred_cmd:n.)

11.24 Keys and functions associated with storage

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

```

1980 \cs_set_protected:Npn \_enumext\_tmp:n #1
1981 {
1982   \keys_define:nn { enumext / #1 }
1983   {
1984     wrap-ans .cs_set_protected:Np = \_enumext\_anskey\_wrapper:n ##1,
1985     wrap-ans .initial:n = \fbox{##1},
1986     wrap-ans .value_required:n = true,
1987     wrap-opt .cs_set_protected:Np = \_enumext\_keyans\_wrapper\_opt:n ##1,
1988     wrap-opt .initial:n = [{##1}],
1989     wrap-opt .value_required:n = true,
1990     save-sep .tl_set:N = \l\_enumext\_store\_keyans\_item\_opt\_sep\_tl,
1991     save-sep .initial:n = {, ~ },
1992     save-sep .value_required:n = true,
1993     mark-ans .tl_set:N = \l\_enumext\_mark\_answer\_sym\_tl,
1994     mark-ans .initial:n = \textasteriskcentered,
1995     mark-ans .value_required:n = true,
1996     mark-pos .choice:,
1997     mark-pos / left .code:n = \str_set:Nn \l\_enumext\_mark\_position\_str { l },
1998     mark-pos / right .code:n = \str_set:Nn \l\_enumext\_mark\_position\_str { r },
1999     mark-pos .initial:n = right,
2000     mark-pos .value_required:n = true,
2001     show-ans .bool_set:N = \l\_enumext\_show\_answer\_bool,
2002     show-ans .initial:n = false,
2003     show-ans .value_required:n = true,
2004     show-pos .bool_set:N = \l\_enumext\_show\_position\_bool,
2005     show-pos .initial:n = false,
2006     show-pos .value_required:n = true,
2007     mark-ref .tl_set:N = \l\_enumext\_mark\_ref\_sym\_tl,
2008     mark-ref .initial:n = \textasteriskcentered,
2009     mark-ref .value_required:n = true,
2010     save-ref .bool_set:N = \l\_enumext\_store\_ref\_key\_bool,
2011     save-ref .initial:n = false,
2012     save-ref .value_required:n = true,
2013   }
2014 }
2015 \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`.

```

2016 \cs_set_protected:Npn \_enumext\_tmp:n #1

```

```

2017 {
2018   \keys_define:nn { enumext / #1 }
2019   {
2020     mark-pos .choice:,
2021     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2022     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2023     mark-pos .initial:n = right,
2024     mark-pos .value_required:n = true,
2025     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2026     show-ans .initial:n = false,
2027     show-ans .value_required:n = true,
2028     show-pos .bool_set:N = \l__enumext_show_position_bool,
2029     show-pos .initial:n = false,
2030     show-pos .value_required:n = true,
2031   }
2032 }
2033 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

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

11.24.1 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 `\l__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered *<keys>*.

```

2034 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2035 {
2036   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2037   {
2038     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2039     \tl_set:ce
2040       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2041       { \__enumext_filter_save_key:n {#1} }
2042   }
2043 }
2044 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2045 {
2046   \bool_if:NF \l__enumext_store_save_key_vii_bool
2047   {
2048     \tl_clear:N \l__enumext_store_save_key_vii_tl
2049     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2050   }
2051 }

```

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

11.24.2 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.

```

2052 \cs_set_protected:Npn \__enumext_tmp:n #1
2053 {
2054   \keys_define:nn { enumext / enumext* }
2055   {
2056     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2057     save-key .value_required:n = true,
2058   }
2059   \keys_define:nn { enumext / #1 }
2060   {

```

```

2061         save-key .code:n = \__enumext_parse_save_key:n {##1},
2062         save-key .value_required:n = true,
2063     }
2064 }
2065 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

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*`.

```

2066 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2067 {
2068     \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2069     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2070     \tl_set:ce
2071     { \l__enumext_store_save_key_ \__enumext_level: _tl }
2072     { \__enumext_filter_save_key:n {#1} }
2073 }
2074 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2075 {
2076     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2077     \tl_clear:N \l__enumext_store_save_key_vii_tl
2078     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2079 }

```

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

11.24.3 Internal functions to store optional arguments

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.

```

2080 \cs_new:Npn \__enumext_filter_save_key:n #1
2081 {
2082     \use:e
2083     {
2084         \keyval_parse:NNn
2085         \__enumext_filter_save_key_key:n
2086         \__enumext_filter_save_key_pair:nn {#1}
2087     }
2088 }

```

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.

```

2089 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2090 {
2091     \str_case:nnF {#1}
2092     {
2093         { resume } {} { resume* } {} { no-store } {}
2094     }
2095     { , { \exp_not:n {#1} } }
2096 }

```

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`, `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.

```

2097 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2098 {
2099     \str_case:nnF {#1}
2100     {
2101         { series } {} { resume } {} { save-ans } {}
2102         { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2103         { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2104         { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2105         { mini-right } {} { mini-right* } {}
2106     }
2107     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2108 }

```

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

11.24.4 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 $\{\langle position \rangle\}\{\langle content \rangle\}$. This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
2109 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2110 {
2111   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2112   {
2113     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2114   }
2115   { #1 }
2116 }
2117 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

11.24.5 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.

```
2118 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2119 {
2120   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2121 }
2122 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

11.24.6 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.

```
2123 \cs_new_protected:Nn \__enumext_store_level_open:
2124 {
2125   \bool_if:NT \l__enumext_check_answers_bool
2126   {
2127     \tl_if_empty:cTF { l__enumext_store_save_key_ \l__enumext_level: _tl }
2128     {
2129       \__enumext_store_addto_seq:n
2130       {
2131         \item \begin{enumext}
2132       }
2133     }
2134     {
2135       \tl_put_left:cn { l__enumext_store_save_key_ \l__enumext_level: _tl }
2136       {
2137         \item \begin{enumext} [
2138       }
2139       \tl_put_right:cn { l__enumext_store_save_key_ \l__enumext_level: _tl }
2140       {
2141         ]
2142       }
2143       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \l__enumext_level: _tl }
2144     }
2145   }
2146 }
2147 \cs_new_protected:Nn \__enumext_store_level_close:
2148 {
2149   \bool_if:NT \l__enumext_check_answers_bool
2150   {
2151     \__enumext_store_addto_seq:n { \end{enumext} }
2152   }
2153 }
```

(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.

```

2154 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2155 {
2156   \bool_if:NT \l__enumext_check_answers_bool
2157   {
2158     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2159     {
2160       \__enumext_store_addto_seq:n
2161       {
2162         \item \mode_leave_vertical:
2163         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2164         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2165       }
2166     }
2167     {
2168       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2169       {
2170         \item \mode_leave_vertical:
2171         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2172         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2173       }
2174       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2175       {
2176     ]
2177     }
2178     \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2179   }
2180 }
2181 }
2182 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2183 {
2184   \bool_if:NT \l__enumext_check_answers_bool
2185   {
2186     \__enumext_store_addto_seq:n { \end{enumext*} }
2187   }
2188 }

```

(End of definition for `__enumext_store_level_open_vii:` and `__enumext_store_level_close_vii:.`)

11.24.7 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`

```

2189 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2190 {
2191   \mode_leave_vertical:
2192   \skip_horizontal:n { -\dim_use:N #2 }
2193   \makebox[\opt][ r ]
2194   {
2195     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2196     {
2197       \tl_use:N \l__enumext_mark_answer_sym_tl
2198     }
2199   }
2200   \skip_horizontal:n { \dim_use:N #2 }
2201 }
2202 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

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

11.25 The command \anskey and internal label and ref

Since we will be “*storing content*” in a list environment within *(sequences)* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over \item when storing. The \anskey command will cover this point and give it similar behaviour to that of \item in the enumext and enumext* environments executed as follows: \anskey[⟨key = val⟩]{⟨content⟩} so first we’ll add the keys break-col, item-join, item-star, item-sym* and item-pos*.

```
2203 \keys_define:nn { enumext / anskey }
2204 {
2205     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2206     break-col .default:n = true,
2207     break-col .value_forbidden:n = true,
2208     item-join .int_set:N = \l__enumext_store_item_join_int,
2209     item-join .value_required:n = true,
2210     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2211     item-star .default:n = true,
2212     item-star .value_forbidden:n = true,
2213     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2214     item-sym* .value_required:n = true,
2215     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2216     item-pos* .value_required:n = true,
2217 }
```

The \anskey command will only be present when using the save-ans key in enumext and enumext* environments, otherwise it will return an error.

\anskey We will first call the function __enumext_anskey_safe_outer: to be sure where we execute the command, then we will check the state of the variable \l__enumext_check_answers_bool set by the key no-store, if is true we will increment \g__enumext_item_anskey_int for the internal “*check answer*” system and execute the function __enumext_anskey_safe_inner:n to ensure that the command is not nested and that the argument is not empty, finally we call the function __enumext_store_anskey_code:nn.

```
2218 \NewDocumentCommand \anskey { o +m }
2219 {
2220     \__enumext_anskey_safe_outer:
2221     \group_begin:
2222         \bool_if:NT \l__enumext_check_answers_bool
2223         {
2224             \int_gincr:N \g__enumext_item_anskey_int
2225             \__enumext_anskey_safe_inner:n {#2}
2226             \__enumext_store_anskey_code:nn {#1} {#2}
2227         }
2228     \group_end:
2229 }
```

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

11.25.1 Internal functions for the command

__enumext_anskey_safe_outer:
__enumext_anskey_safe_inner:n

The __enumext_store_anskey_safe_outer: function will return the appropriate messages when the command is executed outside the environment in which the save-ans key was activated.

```
2230 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2231 {
2232     \bool_if:NF \l__enumext_store_active_bool
2233     {
2234         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2235     }
2236     \int_compare:nNtT { \l__enumext_keyans_level_int } = { 1 }
2237     {
2238         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2239     }
2240     \int_compare:nNtT { \l__enumext_keyans_pic_level_int } = { 1 }
2241     {
2242         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2243     }
2244 }
```

The __enumext_anskey_safe_inner:n function will first check to see if the passed argument is empty and then check to see if the command is nested by returning the appropriate messages.

```
2245 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2246 {
2247     \tl_if_empty:nT {#1}
```

```

2248     {
2249         \msg_error:nn { enumext } { anskey-empty-arg }
2250     }
2251     \int_incr:N \l__enumext_anskey_level_int
2252     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2253     {
2254         \msg_error:nn { enumext } { anskey-nested }
2255     }
2256 }

```

(End of definition for `__enumext_anskey_safe_outer:` and `__enumext_anskey_safe_inner:n`.)

`__enumext_store_anskey_code:nn`

The internal function `__enumext_store_anskey_code:nn` first we pass the *⟨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.

```

2257 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2258 {
2259     \__enumext_store_addto_prop:n {#2}
2260     \bool_if:NT \l__enumext_store_ref_key_bool
2261     {
2262         \__enumext_store_internal_ref:
2263     }
2264     \__enumext_store_anskey_show_left:n { #2 }

```

Now we start processing the *[⟨key = val⟩]* 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 the *⟨keys⟩*, if the `break-col` key is present and the command is running under `enumext` (not in `enumext*`) we will add `\columnbreak` and then `\item`.

```

2265     \tl_if_novalue:nF {#1}
2266     {
2267         \keys_set:nn { enumext / anskey } {#1}
2268     }
2269     \tl_clear:N \l__enumext_store_anskey_arg_tl
2270     \bool_lazy_and:nnT
2271     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2272     { \bool_not_p:n { \l__enumext_starred_bool } }
2273     {
2274         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2275     }
2276     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

If the `item-join` key is present and the command is running under `enumext*` we will add *⟨⟨number⟩⟩* to `\l__enumext_store_anskey_arg_tl`.

```

2277     \bool_lazy_and:nnT
2278     { \bool_not_p:n { \l__enumext_starred_bool } }
2279     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2280     {
2281         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2282         {
2283             ( \exp_not:V \l__enumext_store_item_join_int )
2284         }
2285     }

```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_arg_tl` along with the *⟨argument⟩*.

```

2286     \bool_if:NTF \l__enumext_store_item_star_bool
2287     {
2288         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2289         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2290         {
2291             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2292             {
2293                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2294             }
2295         }
2296         \dim_compare:nT
2297         {
2298             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2299         }
2300         {

```

```

2301         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2302         {
2303             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2304         }
2305     }
2306     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2307 }
2308 {
2309     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2310 }

```

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

```

2311 \bool_lazy_and:nnT
2312 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2313 { \bool_if_p:N \l__enumext_hyperref_bool }
2314 {
2315     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2316     {
2317         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2318         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2319     }
2320 }
2321 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2322 }

```

(End of definition for `__enumext_store_anskey_code:nn`.)

`__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`.

```

2323 \cs_new_protected:Nn \__enumext_store_internal_ref:
2324 {
2325     \cs_set_protected:Npn \__enumext_tmp:n ##1
2326     {
2327         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2328         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2329         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2330         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2331     }
2332     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2333     \cs_set:Npn \__enumext_tmp:n ##1
2334     { . \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.

```

2335 \bool_lazy_all:nT
2336 {
2337     { \bool_if_p:N \g__enumext_starred_bool }
2338     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2339 }
2340 {
2341     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2342     { \tl_use:N \l__enumext_label_copy_vii_tl }
2343 }
2344 \bool_lazy_all:nT
2345 {
2346     { \bool_if_p:N \l__enumext_standar_bool }
2347     { \bool_if_p:N \g__enumext_starred_bool }
2348     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2349 }
2350 {
2351     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2352     {
2353         \tl_use:N \l__enumext_label_copy_vii_tl
2354         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2355     }
2356 }

```

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.

```

2357 \bool_lazy_all:nT
2358 {
2359   { \bool_if_p:N \l__enumext_standar_bool }
2360   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2361   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2362   { \bool_not_p:n { \l__enumext_starred_bool } }
2363 }
2364 {
2365   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2366   {
2367     \tl_use:N \l__enumext_label_copy_i_tl
2368     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2369   }
2370 }
2371 \cs_set:Npn \__enumext_tmp:n ##1
2372 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2373 \bool_lazy_all:nT
2374 {
2375   { \bool_if_p:N \l__enumext_standar_bool }
2376   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2377   { \bool_not_p:n { \g__enumext_starred_bool } }
2378   { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2379 }
2380 {
2381   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2382   {
2383     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2384     . \tl_use:N \l__enumext_label_copy_vii_tl
2385   }
2386 }

```

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

```

2387 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2388 {
2389   \l__enumext_store_name_tl \c_colon_str
2390   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2391 }

```

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.

```

2392 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2393 {
2394   \__enumext_newlabel:nn
2395   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2396   { \l__enumext_newlabel_arg_two_tl }
2397 }
2398 \l__enumext_store_write_aux_file_tl
2399 }

```

(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.

```

2400 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2401 {
2402   \par
2403   \bool_if:NT \l__enumext_starred_bool
2404   {
2405     \cs_set:Nn \__enumext_level: { vii }
2406   }
2407   \__enumext_print_keyans_box:cc
2408   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2409   { \l__enumext_labelsep_ \__enumext_level: _dim }
2410   \__enumext_anskey_wrapper:n { #1 }
2411 }

```

(End of definition for `__enumext_store_anskey_show_wrap:n`)

`__enumext_store_anskey_show_left:n`

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

```

2412 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2413 {
2414   \bool_if:NT \l__enumext_show_answer_bool
2415   {
2416     \__enumext_store_anskey_show_wrap:n { #1 }
2417   }
2418   \bool_if:NT \l__enumext_show_position_bool
2419   {
2420     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2421     {
2422       \group_begin:
2423       \exp_not:N \normalfont
2424       \exp_not:N \footnotesize [ \int_eval:n
2425       {
2426         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2427       }
2428       ]
2429       \group_end:
2430     }
2431     \__enumext_store_anskey_show_wrap:n { #1 }
2432   }
2433 }

```

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

11.26 The environment `anskey*`

`__enumext_scontents_anskey:n`

`__enumext_scontents_anskey:V`

`anskeyenv`

The function `__enumext_scontents_anskey:n` creates a internal `anskeyenv` environment (*custom version* of `scontents`) 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. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§11.33) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.36)

```

2434 \cs_new_protected:Npn \__enumext_scontents_anskey:n #1
2435 {
2436
2437   \newenvsc{anskeyenv}[store-env=#1,print-env=false]
2438   \__enumext_anskey_env_exec:
2439 }
2440 \cs_generate_variant:Nn \__enumext_scontents_anskey:n { V }

```

(End of definition for `__enumext_scontents_anskey:n` and `anskeyenv`.)

`__enumext_anskey_env_exec:`

`__enumext_anskey_env_keys:`

`__enumext_anskey_env_store:`

`__enumext_anskey_env_clean:`

The function `__enumext_internal_mini_page:` creates a internal `__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.

```

2441 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2442 {
2443   \keys_define:nn { scontents / scontents }
2444   {
2445     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2446     break-col .default:n = true,
2447     break-col .value_forbidden:n = true,
2448     item-join .int_gset:N = \g__enumext_store_item_join_int,
2449     item-join .value_required:n = true,
2450     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2451     item-star .default:n = true,
2452     item-star .value_forbidden:n = true,
2453     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2454     item-sym* .value_required:n = true,
2455     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2456     item-pos* .value_required:n = true,
2457     print-env .undefine:,
2458     store-env .undefine:,

```

```

2459         write-out .undefine:,
2460     }
2461 }
2462 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2463 {
2464     \keys_define:nn { scontents / scontents }
2465     {
2466         break-col .undefine:,
2467         item-join .undefine:,
2468         item-star .undefine:,
2469         item-sym* .undefine:,
2470         item-pos* .undefine:,
2471     }
2472 }
2473 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2474 {
2475     \__enumext_before_env:nn { anskeyenv }
2476     {
2477         \__enumext_anskey_env_define_keys:
2478     }
2479     \__enumext_after_env:nn { anskeyenv }
2480     {
2481         \tl_clear:N \l__enumext_store_anskey_env_tl
2482         \tl_clear:N \l__enumext_store_anskey_opt_tl
2483         \seq_gpop_right:cNT
2484         { g__scontents_name_ \g__enumext_store_name_tl _seq } \l__enumext_store_anskey_env_tl
2485         { \seq_item:ce { g__scontents_name_ \g__enumext_store_name_tl _seq } { -1 } }
2486         % add to prop
2487         \regex_replace_all:nnN { \s{2,}}\u{c__scontents_hidden_space_str} { { \% } } \l__enumext_store_
2488         % keys
2489         \__enumext_anskey_env_keys:
2490         % store
2491         \__enumext_anskey_env_store:
2492         % clean
2493         \__enumext_anskey_env_clean:
2494         % Undef
2495         \__enumext_anskey_env_undefine_keys:
2496     }
2497 }

```

The function `__enumext_scontents_anskey_keys:` processing the `[⟨key = val⟩]` passed to the environment and save this in the variable `\l__enumext_store_anskey_opt_tl`. If the `break-col` key is present and the environment is running under `enumext` (not in `enumext*`) we will add the key `break-col`.

```

2498 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2499 {
2500     \bool_lazy_and:nnT
2501     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2502     { \bool_not_p:n { \l__enumext_starred_bool } }
2503     {
2504         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2505     }

```

If the `item-join` key is present and the command is running under `enumext*` we will add to `\l__enumext_store_anskey_opt_tl`.

```

2506     \bool_lazy_and:nnT
2507     { \bool_not_p:n { \l__enumext_starred_bool } }
2508     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2509     {
2510         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2511         {
2512             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2513         }
2514     }

```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_opt_tl`.

```

2515     \bool_if:NT \g__enumext_store_item_star_bool
2516     {
2517         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2518         {
2519             ,item-star,
2520         }

```

```

2521 \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2522 {
2523   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2524   {
2525     ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2526   }
2527 }
2528 \dim_compare:nT
2529 {
2530   \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2531 }
2532 {
2533   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2534   {
2535     ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2536   }
2537 }
2538 }
2539 }

```

We check if the `save-ref` key are active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` with `symbol` set by `mark-ref` key and then store in *(sequence)*.

```

2540 \cs_new_protected:Nn \__enumext_anskey_env_store:
2541 {
2542   \group_begin:
2543     \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2544     {
2545       \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2546       {
2547         \exp_args:Ne
2548         \anskey{ \__scontents_rescan_tokens:x { \l__enumext_store_anskey_env_tl } }
2549       }
2550       {
2551         \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2552         \exp_args:Ne
2553         \anskey{ \__scontents_rescan_tokens:x { \l__enumext_store_anskey_env_tl } }
2554       }
2555     }
2556   \group_end:
2557 }

```

We check if the `save-ref` key are active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` with `symbol` set by `mark-ref` key and then store in *(sequence)*.

```

2558 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2559 {
2560   \bool_gset_false:N \g__enumext_store_columns_break_bool
2561   \int_gzero:N \g__enumext_store_item_join_int
2562   \bool_gset_false:N \g__enumext_store_item_star_bool
2563   \tl_gclear:N \g__enumext_store_item_symbol_tl
2564   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2565 }

```

(End of definition for `__enumext_anskey_env_exec:` and others.)

11.27 Common functions for keyans, keyans* and keyanspic

11.27.1 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`.

```

2566 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2567 {
2568   \tl_clear:N \l__enumext_store_keyans_label_tl
2569   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2570   {
2571     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2572   }
2573   {
2574     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }

```



```

2575     }
2576     \tl_if_novalue:nF { #1 }
2577     {
2578         % Set save-sep
2579         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2580         {
2581             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2582         }
2583         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2584     }
2585     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2586 }

```

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

11.27.2 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).

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.

```

2587 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2588 {
2589     \bool_if:NT \l__enumext_store_ref_key_bool
2590     {
2591         \cs_set_protected:Npn \__enumext_tmp:n ##1
2592         {
2593             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2594             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2595             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2596             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2597         }
2598         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2599         \__enumext_keyans_store_ref_aux_i:
2600     }
2601 }

```

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

```

2602 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2603 {
2604     \bool_if:NT \g__enumext_starred_bool
2605     {
2606         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2607     }
2608     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2609     {
2610         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2611         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vii_tl }
2612     }
2613     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2614     {
2615         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2616         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2617     }
2618     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2619     {
2620         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2621         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2622     }
2623     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2624     {
2625         \l__enumext_store_name_tl \c_colon_str
2626         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2627     }
2628     \__enumext_keyans_store_ref_aux_ii:
2629 }

```

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.

```

2630 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2631 {
2632   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2633   {
2634     \__enumext_newlabel:nn
2635     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2636     { \l__enumext_newlabel_arg_two_tl }
2637   }
2638   \l__enumext_store_write_aux_file_tl
2639 }

```

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

11.27.3 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 *label* `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the optional argument of both commands to the `\l__enumext_store_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

2640 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2641 {
2642   \tl_clear:N \l__enumext_store_keyans_label_tl
2643   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2644   {
2645     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2646   }
2647   {
2648     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2649   }
2650   \tl_if_novalue:nF { #1 }
2651   {
2652     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2653     {
2654       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2655       {
2656         \l__enumext_store_keyans_item_opt_sep_tl
2657       }
2658     }
2659     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2660   }
2661   \__enumext_keyans_addto_seq_link:
2662 }

```

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.

```

2663 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2664 {
2665   \bool_lazy_and:nnT
2666   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2667   { \bool_if_p:N \l__enumext_hyperref_bool }
2668   {
2669     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2670     {
2671       \hfill \exp_not:N \hyperlink
2672       {
2673         \exp_not:V \l__enumext_newlabel_arg_one_tl
2674       }
2675       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2676     }
2677   }
2678   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2679   \bool_if:NT \l__enumext_check_answers_bool
2680   {
2681     \int_gincr:N \g__enumext_item_anskey_int

```

```

2682     }
2683 }

```

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

11.27.4 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.

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.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:
2684 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2685 {
2686   \tl_if_novalue:nF { #1 }
2687   {
2688     \tl_set:Nc \l__enumext_keyans_item_opt_tl { #1 }
2689   }
2690   \bool_if:NT \l__enumext_show_answer_bool
2691   {
2692     \__enumext_keyans_show_ans:
2693   }
2694   \bool_if:NT \l__enumext_show_position_bool
2695   {
2696     \__enumext_keyans_show_pos:
2697   }
2698 }
2699 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2700 {
2701   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2702   {
2703     \bool_lazy_or:nnT
2704     { \bool_if_p:N \l__enumext_show_answer_bool }
2705     { \bool_if_p:N \l__enumext_show_position_bool }
2706     {
2707       \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2708     }
2709   }
2710 }
2711 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2712 {
2713   \tl_put_left:Nn \l__enumext_label_v_tl
2714   {
2715     \__enumext_print_keyans_box:NN
2716     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2717   }
2718 }
2719 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2720 {
2721   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2722   {
2723     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2724     {
2725       \group_begin:
2726       \exp_not:N \normalfont
2727       \exp_not:N \footnotesize [ \int_eval:n
2728       {
2729         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2730       }
2731       ]
2732       \group_end:
2733     }
2734   }
2735   {
2736     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2737     {
2738       \group_begin:
2739       \exp_not:N \normalfont
2740       \exp_not:N \footnotesize [ \int_eval:n
2741       {
2742         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2743       }

```

```

2744         ]
2745     \group_end:
2746 }
2747 }
2748 \tl_put_left:Nn \l__enumext_label_v_tl
2749 {
2750     \__enumext_print_keyans_box:NN
2751     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2752 }
2753 }

```

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

11.28 Setting `item-sym*` and `item-pos*` keys

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

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

```

item-pos* 2754 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2755 {
2756     \keys_define:nn { enumext / #1 }
2757     {
2758         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2759         item-sym* .value_required:n = true,
2760         item-sym* .initial:n = {$\star$},
2761         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2762         item-pos* .value_required:n = true,
2763     }
2764 }
2765 \clist_map_inline:nn
2766 {
2767     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2768 }
2769 { \__enumext_tmp:nn #1 }

```

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

11.29 Redefining `\footnote` command

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

```

2770 \cs_new_protected:Nn \__enumext_footnotetext:nn
2771 {
2772     \footnotetext[#1]{#2}
2773 }
2774 \cs_new_protected:Nn \__enumext_renew_footnote:
2775 {
2776     \seq_gclear:N \g__enumext_footnote_arg_seq
2777     \seq_gclear:N \g__enumext_footnote_int_seq
2778     \RenewDocumentCommand \footnote { o +m }
2779     {
2780         \tl_if_novalue:nTF {##1}
2781         {
2782             \stepcounter{footnote}
2783             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2784         }
2785         {
2786             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2787         }
2788         \footnotemark [ \g__enumext_footnote_int ]
2789         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2790         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2791     }
2792 }
2793 \cs_new_protected:Nn \__enumext_print_footnote:
2794 {
2795     \seq_if_empty:NF \g__enumext_footnote_int_seq
2796     {
2797         \seq_map_pairwise_function:NNN
2798         \g__enumext_footnote_int_seq

```

```

2799         \g__enumext_footnote_arg_seq
2800         \__enumext_footnotetext:nn
2801     }
2802 }

```

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

11.30 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.

11.30.1 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: (§11.31)`.

```

2803 \cs_new_protected:Npn \__enumext_default_item:n #1
2804 {
2805     \tl_if_novalue:nTF {#1}
2806     {
2807         \bool_if:NT \l__enumext_check_answers_bool
2808         {
2809             \int_gincr:N \g__enumext_item_number_int
2810         }
2811         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2812         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2813     }
2814     {
2815         \bool_set_eq:cc
2816         { \l__enumext_wrap_label_ \__enumext_level: _bool }
2817         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2818         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2819     }
2820 }

```

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

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

```

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

```

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

Then we will see the state of the variable `\l__enumext_check_ans_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`.

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label: (§11.31)`.

```

2821 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2822 {
2823     \tl_if_novalue:nF {#1}
2824     {
2825         \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2826     }
2827     \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2828     \tl_if_novalue:nTF {#2}

```

```

2829     {
2830         \dim_set_eq:cc
2831         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2832         { \__enumext_labelsep_ \__enumext_level: _dim }
2833     }
2834     {
2835         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2836     }
2837     \bool_if:NT \__enumext_check_answers_bool
2838     {
2839         \int_gincr:N \g__enumext_item_number_int
2840     }
2841     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2842     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2843 }

```

(End of definition for __enumext_starred_item:nn.)

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.32.2).

```

2844 \cs_new_protected:Nn \__enumext_redefine_item:
2845 {
2846     \RenewDocumentCommand \item { s o o }
2847     {
2848         \bool_if:nTF {##1}
2849         {
2850             \__enumext_starred_item:nn {##2} {##3}
2851         }
2852         { \__enumext_default_item:n {##2} }
2853     }
2854 }

```

(End of definition for __enumext_redefine_item:.)

11.30.2 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`.

```

2855 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2856 {
2857     \tl_if_novalue:nTF { #1 }
2858     {
2859         \bool_set_true:N \__enumext_wrap_label_v_bool
2860         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2861     }
2862     {
2863         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2864         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2865     }
2866 }

```

(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 $\langle label \rangle$.

```

2867 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2868 {
2869     \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2870     \__enumext_keyans_show_left:n { #1 }
2871     \bool_set_true:N \__enumext_wrap_label_v_bool
2872     \__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 $\langle sequence \rangle$.

```

2873 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2874 \__enumext_keyans_addto_prop:n { #1 }
2875 \__enumext_keyans_store_ref:
2876 \__enumext_keyans_addto_seq:n { #1 }
2877 \int_gincr:N \g__enumext_check_starred_cmd_int
2878 }

```

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

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.32.2).

```

2879 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2880 {
2881   \RenewDocumentCommand \item { s o }
2882   {
2883     \bool_if:nTF {##1}
2884     {
2885       \peek_remove_spaces:n
2886       {
2887         \__enumext_keyans_starred_item:n {##2}
2888       }
2889     }
2890     {
2891       \__enumext_keyans_default_item:n {##2}
2892     }
2893   }
2894 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 13.)

11.31 Redefining `\makelabel` command

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

11.31.1 Redefining `\makelabel` for `enumext`

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

```

2895 \cs_new_protected:Nn \__enumext_item_starred:
2896 {
2897   \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2898   {
2899     \mode_leave_vertical:
2900     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2901     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2902     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2903   }
2904 }

```

(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 (§11.32.2).

```

2905 \cs_new_protected:Nn \__enumext_make_label:
2906 {
2907   \RenewDocumentCommand \makelabel { m }
2908   {
2909     \tl_use:c { \l__enumext_label_fill_left_ \l__enumext_level: _tl }
2910     \tl_use:c { \l__enumext_label_font_style_ \l__enumext_level: _tl }
2911     \bool_if:cTF { \l__enumext_wrap_label_ \l__enumext_level: _bool }
2912     {
2913       \__enumext_item_starred:

```



```

2914         \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2915     }
2916     { ##1 }
2917     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2918     \tl_gclear:N \g__enumext_item_symbol_tl
2919 }
2920 }

```

(End of definition for `__enumext_make_label:`)

11.31.2 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 (§11.32.2).

```

2921 \cs_new_protected:Nn \__enumext_keyans_make_label:
2922 {
2923     \RenewDocumentCommand \makeLabel { m }
2924     {
2925         \tl_use:N \l__enumext_label_fill_left_v_tl
2926         \tl_use:N \l__enumext_label_font_style_v_tl
2927         \bool_if:NTF \l__enumext_wrap_label_v_bool
2928         {
2929             \__enumext_wrapper_label_v:n { ##1 }
2930         }
2931         { ##1 }
2932         \tl_use:N \l__enumext_label_fill_right_v_tl
2933     }
2934 }

```

(End of definition for `__enumext_keyans_make_label:`)

11.32 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`.

11.32.1 Calculation of `\leftmargin` and `\itemindent`

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

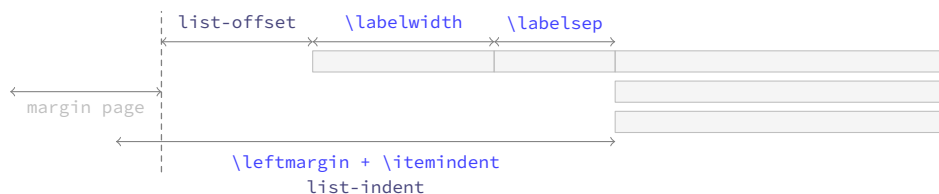


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

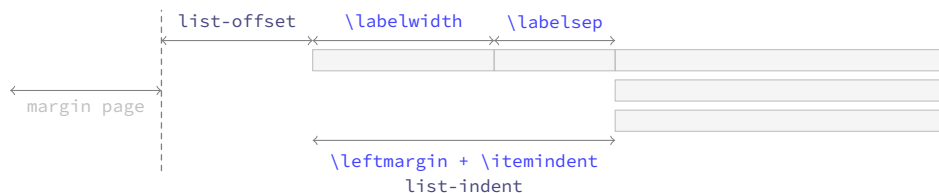


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

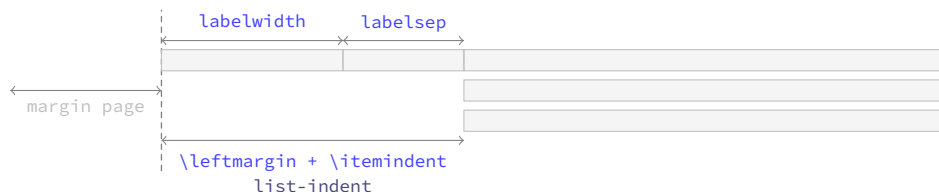


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

```

#1: \__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 (§11.32.2).

```

2935 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2936 {
2937   \dim_compare:nNt { #1 } < { \c_zero_dim }
2938   {
2939     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2940     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2941   }
2942   \dim_compare:nNt { #2 } < { \c_zero_dim }
2943   {
2944     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2945     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2946   }

```

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

```

2947   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2948   \dim_compare:nNtTF { #4 } < { \c_zero_dim }
2949   {
2950     \dim_set:Nn #6 { #1 + #2 - #4 }
2951     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2952   }
2953   {
2954     \dim_compare:nNt { #4 } = { #1 + #2 }
2955     { \dim_set:Nn #6 { \c_zero_dim } }
2956     \dim_compare:nNt { #4 } < { #1 + #2 }
2957     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2958     \dim_compare:nNt { #4 } > { #1 + #2 }
2959     {
2960       \dim_set:Nn #6 { -#1 - #2 + #4 }
2961       \dim_set:Nn #6 { #6*-1 }
2962     }
2963     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2964   }
2965 }
2966 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

11.32.2 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:
2967 \cs_set_protected:Npn \__enumext_tmp:n #1
2968 {
2969   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2970   {
2971     \__enumext_calc_hspace:ccccccc
2972     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2973     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2974     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2975     { \__enumext_leftmargin_tmp_#1_bool }
2976     \clist_map_inline:nn
2977     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2978     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2979     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2980     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2981     \usecounter { enumX#1 }
2982     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```

```

2983 \str_if_eq:nnTF {#1} { v }
2984 {
2985   \__enumext_keyans_redefine_item:
2986   \__enumext_keyans_make_label:
2987   \__enumext_keyans_ref:
2988   \__enumext_keyans_fake_item:
2989   \bool_if:cT { l__enumext_show_length_#1_bool }
2990   {
2991     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2992   }
2993 }
2994 {
2995   \__enumext_redefine_item:
2996   \__enumext_make_label:
2997   \__enumext_standar_ref:
2998   \__enumext_fake_item:
2999   \bool_if:cT { l__enumext_show_length_#1_bool }
3000   {
3001     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3002   }
3003 }
3004 }
3005 }
3006 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for `__enumext_list_arg_two_i:` and others.)

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `0pt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `parindent` to make it set the value of `listparindent` and `parsep` to set the value of `\parskip` locally.

```

3007 \cs_set_protected:Npn \__enumext_tmp:n #1
3008 {
3009   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3010   {
3011     \__enumext_calc_hspace:ccccc
3012     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3013     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3014     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3015     { l__enumext_leftmargin_tmp_#1_bool }
3016     \clist_map_inline:nn
3017     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3018     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3019     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3020     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3021     \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3022     \skip_zero:N \partopsep
3023     \usecounter { enumX#1 }
3024     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3025     \__enumext_starred_ref:
3026     \str_if_eq:nnTF {#1} { vii }
3027     {
3028       \__enumext_fake_item_vii:
3029       \bool_if:cT { l__enumext_show_length_vii_bool }
3030       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3031     }
3032     {
3033       \__enumext_fake_item_viii:
3034       \bool_if:cT { l__enumext_show_length_#1_bool }
3035       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3036     }
3037   }
3038 }
3039 \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:`.)

11.33 The environment `enumext`

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

```

3040 \NewDocumentEnvironment{enumext}{0}{}
3041 {

```

```

3042     \__enumext_safe_exec:
3043     \__enumext_parse_keys:n {#1}
3044     \__enumext_before_list:
3045     \__enumext_start_store_level:
3046     \__enumext_start_list:nn
3047     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3048     {
3049         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3050         \__enumext_before_keys_exec:
3051     }
3052     \__enumext_after_args_exec:
3053 }
3054 {
3055     \__enumext_stop_list:
3056     \__enumext_stop_store_level:
3057     \__enumext_after_list:
3058 }

```

(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:`.

```

3059 \cs_new_protected:Nn \__enumext_safe_exec:
3060 {
3061     \__enumext_internal_mini_page:
3062     \__enumext_is_not_nested:
3063     \int_incr:N \l__enumext_level_int
3064     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3065     { \msg_fatal:nn { enumext } { list-too-deep } }
3066     \bool_set_true:N \l__enumext_standar_bool
3067     \__enumext_is_on_first_level:
3068 }

```

(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`.

```

3069 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3070 {
3071     \tl_if_novalue:nF {#1}
3072     {
3073         \str_clear:N \l__enumext_series_str
3074         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3075         {
3076             \keys_set:nn { enumext / level-1 } {#1}
3077             \__enumext_parse_series:n {#1}
3078         }
3079         {
3080             \exp_args:Ne \keys_set:nn
3081             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3082         }
3083         \__enumext_store_active_keys:n {#1}
3084     }
3085 }

```

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

```

3086 \cs_new_protected:Nn \__enumext_start_store_level:

```

```

3087 {
3088   \bool_lazy_all:nT
3089   {
3090     { \bool_if_p:N \l__enumext_store_active_bool }
3091     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3092     { \bool_not_p:n { \g__enumext_starred_bool } }
3093   }
3094   {
3095     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3096     {
3097       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3098       \__enumext_store_level_open:
3099     }
3100   }
3101   \bool_lazy_all:nT
3102   {
3103     { \bool_if_p:N \l__enumext_store_active_bool }
3104     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3105     { \bool_if_p:N \g__enumext_starred_bool }
3106   }
3107   {
3108     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3109     {
3110       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3111       \__enumext_store_level_open:
3112     }
3113   }
3114 }
3115 \cs_new_protected:Nn \__enumext_stop_store_level:
3116 {
3117   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3118   {
3119     \__enumext_store_level_close:
3120   }
3121 }

```

(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.

```

3122 \cs_new_protected:Nn \__enumext_before_list:
3123 {
3124   \__enumext_vspace_above:
3125   \__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.

```

3126   \__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.

```

3127   \dim_compare:nNnT
3128   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3129   {
3130     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3131     {
3132       \linewidth
3133       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3134       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3135     }

```

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

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

```

3136         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3137         \int_gincr:N \g__enumext_minipage_stat_int
3138         \__enumext_mini_addvspace:
3139         \nointerlineskip\noindent
3140         \begin{__enumext_mini_env*}
3141         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3142     }
3143     \__enumext_multicols_start:
3144 }

```

(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.

```

3145 \cs_new_protected:Nn \__enumext_multicols_start:
3146 {
3147     \int_compare:nNt
3148     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3149     {
3150         \dim_compare:nNt
3151         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3152         {
3153             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3154             {
3155                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3156                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3157                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3158                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3159             }
3160         }
3161         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3162         \skip_zero:N \multicolsep
3163         \int_compare:nNt { \l__enumext_level_int } > { 1 }
3164         {
3165             \dim_zero:N \columnseprule
3166         }
3167     }

```

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.

```

3167         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3168         {
3169             \__enumext_multi_addvspace:
3170         }
3171         \raggedcolumns
3172         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3173     }
3174 }

```

(End of definition for __enumext_multicols_start:.)

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

```

3175 \cs_new_protected:Nn \__enumext_multicols_stop:
3176 {
3177     \int_compare:nNt
3178     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3179     {
3180         \end{multicols}
3181         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3182         {
3183             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3184         }
3185     }
3186 }

```

(End of definition for __enumext_multicols_stop:.)

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

```

3187 \cs_new_protected:Nn \__enumext_after_list:
3188 {
3189   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3190   {
3191     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3192     {
3193       \msg_warning:nn { enumext } { missing-miniright }
3194       \miniright
3195     }
3196     \int_gzero:N \g__enumext_minipage_stat_int
3197     \end{\__enumext_mini_env*}
3198     \par\addvspace { \l__enumext_minipage_after_skip }
3199   }
3200   { \__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`.

```

3201   \__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 `\l__enumext_standar_bool` to false and save the *current value* of the counter for `series`, `resume` and `resume*` keys.

```

3202   \__enumext_after_stop_list:
3203   \__enumext_vspace_below:
3204   \bool_set_false:N \l__enumext_standar_bool
3205   \__enumext_resume_save_counter:
3206 }

```

(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`.

```

3207 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

11.34 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.

```

3208 \NewDocumentEnvironment{keyans}{0}{}
3209 {
3210   \__enumext_keyans_safe_exec:
3211   \__enumext_keyans_parse_keys:n {#1}
3212   \__enumext_before_list_v:
3213   \__enumext_start_list:nn
3214   { \tl_use:N \l__enumext_label_v_tl }
3215   {
3216     \__enumext_list_arg_two_v:
3217     \__enumext_before_keys_exec_v:
3218   }
3219   \__enumext_after_args_exec_v:
3220 }
3221 {
3222   \__enumext_check_starred_cmd:n { item }
3223   \__enumext_stop_list:
3224   \__enumext_after_list_v:
3225 }

```

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

`__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.

```

3226 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3227 {

```



```

3228 \bool_if:NF \l__enumext_store_active_bool
3229 {
3230   \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3231 }
3232 \int_incr:N \l__enumext_keyans_level_int
3233 \bool_set_true:N \l__enumext_keyans_env_bool
3234 \__enumext_keyans_save_start_line:
3235 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3236 \bool_set_false:N \l__enumext_store_active_bool
3237 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3238 {
3239   \msg_error:nn { enumext } { keyans-nested }
3240 }
3241 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3242 {
3243   \msg_error:nn { enumext } { keyans-wrong-level }
3244 }
3245 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

3246 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3247 {
3248   \keys_set:nn { enumext / keyans } { #1 }
3249 }

```

(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.

```

3250 \cs_new_protected:Nn \__enumext_before_list_v:
3251 {
3252   \__enumext_vspace_above_v:
3253   \__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.

```

3254 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3255 {
3256   \dim_set:Nn \l__enumext_minipage_left_v_dim
3257   {
3258     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3259   }

```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\mini-right` 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*.

```

3260 \bool_set_true:N \l__enumext_minipage_active_v_bool
3261 \int_gincr:N \g__enumext_minipage_stat_int
3262 \__enumext_keyans_mini_addvspace:
3263 \nointerlineskip\noindent
3264 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3265 }

```

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

```

3266 \__enumext_keyans_multicols_start:
3267 }

```

(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.

```

3268 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3269 {
3270   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3271   {

```

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

```

3272     \dim_compare:nNt { \__enumext_columns_sep_v_dim } = { \c_zero_dim }
3273     {
3274       \dim_set:Nn \__enumext_columns_sep_v_dim
3275       {
3276         (
3277           \__enumext_labelwidth_v_dim + \__enumext_labelsep_v_dim
3278         ) / \__enumext_columns_v_int
3279         - \__enumext_listoffset_v_dim
3280       }
3281     }
3282     \dim_set_eq:NN \columnsep \__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).

```

3283     \skip_zero:N \multicolsep
3284     \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.

```

3285     \bool_if:NF \__enumext_minipage_active_v_bool
3286     {
3287       \__enumext_keyans_multi_addvspace:
3288     }
3289     \raggedcolumns
3290     \begin{multicols}{\__enumext_columns_v_int }
3291   }
3292 }

```

(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 `__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

3293 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3294 {
3295   \int_compare:nNt { \__enumext_columns_v_int } > { 1 }
3296   {
3297     \end{multicols}
3298     \bool_if:NF \__enumext_minipage_active_v_bool
3299     {
3300       \par\addvspace{ \__enumext_multicols_below_v_skip }
3301     }
3302   }
3303 }

```

(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 `__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 `__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3304 \cs_new_protected:Nn \__enumext_after_list_v:
3305 {
3306   \bool_if:NtF \__enumext_minipage_active_v_bool
3307   {
3308     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3309     {
3310       \msg_warning:nn { enumext } { missing-miniright }
3311       \miniright
3312     }
3313     \int_gzero:N \g__enumext_minipage_stat_int

```

```

3314         \end{__enumext_mini_env*}
3315         \par\addvspace{ \l__enumext_minipage_after_skip }
3316     }
3317     { __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.

```

3318     \bool_set_false:N \l__enumext_keyans_env_bool
3319     \__enumext_after_stop_list_v:
3320     \__enumext_vspace_below_v:
3321 }

```

(End of definition for `__enumext_after_list_v:`)

11.35 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.

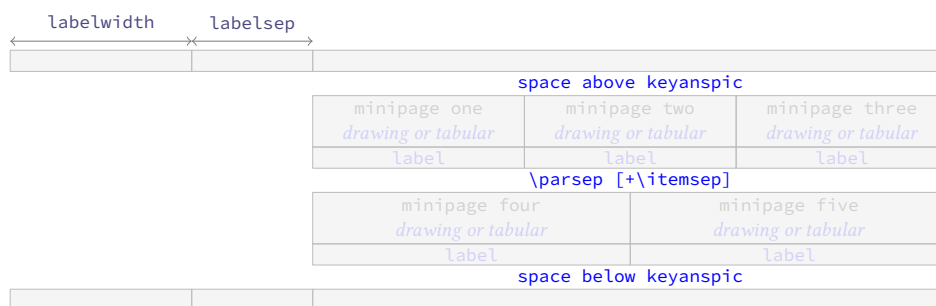


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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?](#).

11.35.1 The command `\anspic`

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

```

3322 \NewDocumentCommand \anspic { s o +m }
3323 {

```

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

```

3324     \bool_if:NF \l__enumext_store_active_bool
3325     {
3326         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3327     }
3328     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3329     {
3330         \msg_error:nn { enumext } { keyanspic-wrong-level }
3331     }
3332     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3333     {
3334         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3335     }

```

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

```

3336     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3337     {
3338         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3339     }
3340 }

```

(End of definition for `\anspic`. This function is documented on page 14.)

__enumext_keyans_anspic_code:nnn

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

```

3341 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3342 {
3343   \stepcounter { enumXvi }
3344   #3 \\\
3345   \bool_if:nT { #1 }
3346   {
3347     \__enumext_keyans_addto_prop:n { #2 }
3348     \__enumext_keyans_store_ref:
3349     \__enumext_keyans_addto_seq:n { #2 }
3350     \int_gincr:N \g__enumext_check_starred_cmd_int
3351     \bool_lazy_or:nnT
3352     { \bool_if_p:N \l__enumext_show_answer_bool }
3353     { \bool_if_p:N \l__enumext_show_position_bool }
3354     {
3355       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3356       \__enumext_keyans_show_left:n { #2 }
3357       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3358     }
3359   }
3360   \tl_use:N \l__enumext_label_font_style_v_tl
3361   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3362 }

```

(End of definition for __enumext_keyans_anspic_code:nnn.)

11.35.2 The environment *keyanspic*

keyanspic

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

```

3363 \NewDocumentEnvironment{keyanspic}{o}
3364 {
3365   \__enumext_keyans_pic_safe_exec:
3366   \__enumext_start_list:nn
3367   { }
3368   {
3369     \__enumext_keyans_pic_arg_two:
3370   }

```

We apply the “adjusted” vertical spacing above the environment

```

3371   \vspace { \l__enumext_keyans_pic_above_skip }
3372 }

```

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

```

3373 {
3374   \tl_if_novalue:nTF { #1 }
3375   {
3376     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3377   }
3378   { \__enumext_keyans_pic_do:n { #1 } }
3379   \__enumext_stop_list:
3380   \__enumext_check_starred_cmd:n { anspic }
3381   \setcounter { enumXvi } { 0 }
3382   \vspace { \l__enumext_topsep_v_skip }
3383   %\bool_set_false:N \l__enumext_store_active_bool
3384 }

```

(End of definition for *keyanspic*. This function is documented on page 14.)

__enumext_keyans_pic_safe_exec:

The function __enumext_keyans_pic_safe_exec: check nested and level position inside the *enumext* environment.

```

3385 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3386 {
3387   \int_incr:N \l__enumext_keyans_pic_level_int
3388   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3389   {
3390     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3391     }
3392     \__enumext_keyans_save_start_line:
3393 }

```

(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.

```

3394 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3395 {
3396     \dim_compare:nNnT { #1 } < { 0pt }
3397     { \skip_set:Nn #1 { -#1 } }
3398 }

```

(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.

```

3399 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3400 {

```

The first thing to do is to set the boolean variable \l__enumext_leftmargin_tmp_v_bool handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```

3401     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3402     \__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.

```

3403     \skip_add:Nn \parsep { \itemsep }
3404     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3405     \dim_zero:N \labelwidth
3406     \dim_zero:N \listparindent
3407     \dim_zero:N \labelsep
3408     \skip_zero:N \partopsep
3409     \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.

```

3410     \__enumext_keyans_pic_skip_abs:N \parsep
3411     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3412     {
3413         \box_dp:N \strutbox
3414         + \l__enumext_topsep_v_skip
3415         - \parsep
3416     }
3417     \__enumext_item_std:w \scan_stop:
3418 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

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

```

3419 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3420 {
3421     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3422 }
3423 \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.

```

3424 \cs_new_protected:Npn \__enumext_keyans_pic_row:n
3425 {
3426     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3427     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3428     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }

```

```

3429 \int_step_inline:nnn
3430 { \l__enumext_keyans_pic_above_int + 1 }
3431 { \l__enumext_keyans_pic_below_int }
3432 {
3433   \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3434   \centering
3435   \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3436   \__enumext_endminipage:
3437 }
3438 \par
3439 }

```

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

11.36 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`).

11.36.1 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.

```

3440 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3441 {
3442   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3443   {
3444     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3445     {
3446       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3447       / \l__enumext_columns_vii_int
3448     }
3449   }
3450   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3451   \dim_set:Nn \l__enumext_item_width_vii_dim
3452   {
3453     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3454     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3455     - \l__enumext_labelsep_vii_dim
3456   }
3457   \dim_zero_new:N \itemwidth
3458 }

```

(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(\langle number \rangle)` will be stored together with the value of `\itemwidth`.

```

3459 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3460 {
3461   \int_set:Nn \l__enumext_joined_item_vii_int { #1 }
3462   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3463   {
3464     \msg_warning:nnee { enumext } { item-joined }
3465     { \int_use:N \l__enumext_joined_item_vii_int }
3466     { \int_use:N \l__enumext_columns_vii_int }
3467     \int_set:Nn \l__enumext_joined_item_vii_int
3468     {
3469       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3470     }
3471   }
3472   \int_compare:nNnT
3473   { \l__enumext_joined_item_vii_int }
3474   >
3475   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```

```

3476     {
3477       \msg_warning:nnee { enumext } { item-joined-columns }
3478       { \int_use:N \l__enumext_joined_item_vii_int }
3479       {
3480         \int_eval:n
3481         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3482       }
3483       \int_set:Nn \l__enumext_joined_item_vii_int
3484       {
3485         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3486       }
3487     }

```

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

```

3488     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3489     {
3490       \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3491       \int_decr:N \l__enumext_joined_item_aux_vii_int
3492       \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3493       \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3494       \dim_set:Nn \l__enumext_joined_width_vii_dim
3495       {
3496         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3497         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3498           + \l__enumext_columns_sep_vii_dim
3499           ) * \l__enumext_joined_item_aux_vii_int
3500       }
3501       \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3502     }
3503     {
3504       \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3505       \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3506     }
3507   }

```

(End of definition for \l__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`.

```

3508   \cs_new_protected:Nn \__enumext_start_mini_vii:
3509   {
3510     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3511     {
3512       \dim_set:Nn \l__enumext_minipage_left_vii_dim
3513       {
3514         \linewidth
3515         - \l__enumext_minipage_right_vii_dim
3516         - \l__enumext_minipage_hsep_vii_dim
3517       }
3518       \bool_set_true:N \l__enumext_minipage_active_vii_bool
3519       \dim_gset_eq:NN
3520       \g__enumext_minipage_right_vii_dim
3521       \l__enumext_minipage_right_vii_dim
3522       \__enumext_mini_addvspace_vii:
3523       \nointerlineskip\noindent
3524       \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3525     }
3526   }

```

(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:n` to execute the `__enumext-mini_env*` on the “right side”.

```

3527   \cs_new_protected:Nn \__enumext_stop_mini_vii:
3528   {
3529     \bool_if:NT \l__enumext_minipage_active_vii_bool

```



```

3530     {
3531         \end{__enumext_mini_env*}
3532         \hfill
3533         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3534     }
3535 }

```

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”.

```

3536 \__enumext_after_env:nn {enumext*}
3537 {
3538     \bool_if:NT \g__enumext_minipage_active_vii_bool
3539     {
3540         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3541         \par\addvspace { \g__enumext_minipage_right_skip }
3542         \bool_if:NF \g__enumext_minipage_center_vii_bool
3543         {
3544             \centering
3545         }
3546         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3547         \end{__enumext_mini_env*}
3548         \par\addvspace{ \g__enumext_minipage_after_skip }
3549     }
3550     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3551     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3552     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3553     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3554     \bool_gset_false:N \g__enumext_starred_bool
3555 }

```

(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.

```

3556 \NewDocumentEnvironment{enumext*}{ o }
3557 {
3558     \__enumext_safe_exec_vii:
3559     \__enumext_parse_keys_vii:n {#1}
3560     \__enumext_before_list_vii:
3561     \__enumext_start_store_level_vii:
3562     \__enumext_start_list:nn { }
3563     {
3564         \__enumext_list_arg_two_vii:
3565         \__enumext_before_keys_exec_vii:
3566     }
3567     \__enumext_starred_columns_set_vii:
3568     \item[] \scan_stop:
3569     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3570     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3571 }
3572 {
3573     \__enumext_stop_item_tmp_vii:
3574     \__enumext_remove_extra_parsep_vii:
3575     \__enumext_stop_list:
3576     \__enumext_stop_store_level_vii:
3577     \__enumext_after_list_vii:
3578 }

```

(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`.

```

3579 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3580 {
3581     \__enumext_internal_mini_page:
3582     \__enumext_is_not_nested:
3583     \int_incr:N \l__enumext_level_h_int
3584     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3585     {
3586         \msg_error:nn { enumext } { nested }

```

```

3587     }
3588     \bool_set_true:N \l__enumext_starred_bool
3589     \__enumext_is_on_first_level:
3590 }

```

(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*.

```

3591 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3592 {
3593     \tl_if_novalue:nF {#1}
3594     {
3595         \str_clear:N \l__enumext_series_str
3596         \keys_set:nn { enumext / enumext* } {#1}
3597         \__enumext_parse_series:n {#1}
3598         \__enumext_store_active_keys_vii:n {#1}
3599     }
3600 }

```

(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*.

```

3601 \cs_new_protected:Nn \__enumext_before_list_vii:
3602 {
3603     \__enumext_vspace_above_vii:
3604     \__enumext_check_ans_active:
3605     \__enumext_before_args_exec_vii:
3606     \__enumext_start_mini_vii:
3607 }

```

(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.

```

3608 \cs_new_protected:Nn \__enumext_after_list_vii:
3609 {
3610     \__enumext_stop_mini_vii:
3611     \__enumext_after_stop_list_vii:
3612     \__enumext_check_ans_key_hook:
3613     \__enumext_vspace_below_vii:
3614     \bool_set_false:N \l__enumext_starred_bool
3615     \__enumext_resume_save_counter:
3616 }

```

(End of definition for __enumext_after_list_vii:.)

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

```

3617 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3618 {
3619     \bool_if:NT \l__enumext_store_active_bool
3620     {
3621         \int_compare:nNtT { \l__enumext_level_int } > { \c_zero_int }
3622         {
3623             \__enumext_store_level_open_vii:
3624         }
3625     }
3626 }
3627 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3628 {
3629     \bool_if:NT \l__enumext_store_active_bool

```

```

3630     {
3631         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3632     {
3633         \__enumext_store_level_close_vii:
3634     }
3635 }
3636 }

```

(End of definition for __enumext_start_store_level_vii: and __enumext_stop_store_level_vii:.)

11.36.2 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.

```

3637 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3638 {
3639     \__enumext_stop_item_tmp_vii:
3640     \int_incr:N \l__enumext_item_column_pos_vii_int
3641     \int_gincr:N \g__enumext_item_count_all_vii_int
3642     \__enumext_item_peek_args_vii:
3643 }

```

(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).

```

3644 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3645 {
3646     \peek_meaning:NTF (
3647     { \__enumext_joined_item_vii:w }
3648     { \__enumext_joined_item_vii:w (1) }
3649 }

```

(End of definition for __enumext_item_peek_args_vii:.)

__enumext_joined_item_vii:w

The function __enumext_joined_item_vii:w will first call the function __enumext_starred_joined_item_vii:n in charge of setting the *width* of the box that will store the content passed to \item. Then we will look for the argument “*”, if it is present we will call the function __enumext_starred_item_vii:w otherwise we will call the function __enumext_standar_item_vii:w.

```

3650 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3651 {
3652     \__enumext_starred_joined_item_vii:n {#1}
3653     \peek_meaning_remove:NTF *
3654     { \__enumext_starred_item_vii:w }
3655     { \__enumext_standar_item_vii:w }
3656 }

```

(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].

```

3657 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3658 {
3659     \bool_set_false:N \l__enumext_item_starred_vii_bool
3660     \peek_meaning:NTF [
3661     {
3662         \bool_set_eq:NN
3663         \l__enumext_wrap_label_vii_bool
3664         \l__enumext_wrap_label_opt_vii_bool
3665         \__enumext_start_item_vii:w

```

```

3666     }
3667     {
3668         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3669         \legacy_if_set_true:n { @noitemarg }
3670         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3671     }
3672 }

```

(End of definition for __enumext_standar_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*[\langle symbol \rangle] and \item*[\langle symbol \rangle][\langle offset \rangle].

```

\__enumext_starred_item_vii:w
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii:w
3673 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3674 {
3675     \bool_set_true:N \l__enumext_item_starred_vii_bool
3676     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3677     \peek_meaning:NTF [
3678         { \__enumext_starred_item_vii_aux_i:w }
3679         { \__enumext_starred_item_vii_aux_ii:w }
3680     }
3681     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3682     {
3683         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3684         \__enumext_starred_item_vii_aux_ii:w
3685     }
3686     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3687     {
3688         \peek_meaning:NTF [
3689             { \__enumext_starred_item_vii_aux_iii:w }
3690             {
3691                 \dim_set_eq:NN
3692                 \l__enumext_item_symbol_sep_vii_dim
3693                 \l__enumext_labelsep_vii_dim
3694                 \legacy_if_set_true:n { @noitemarg }
3695                 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3696             }
3697         }
3698     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3699     {
3700         \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3701         \legacy_if_set_true:n { @noitemarg }
3702         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3703     }

```

(End of definition for __enumext_starred_item_vii:w and others.)

11.36.3 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 __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.

```

3704 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3705 {
3706     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3707     \legacy_if:nT { @noitemarg }
3708     {
3709         \legacy_if_set_false:n { @noitemarg }
3710         \legacy_if:nT { @nmbrrlist }
3711         {
3712             \bool_if:NT \l__enumext_hyperref_bool
3713             {
3714                 \legacy_if_set_true:n { @hyper@item }
3715             }
3716             \refstepcounter{enumXvii}
3717             \bool_if:NT \l__enumext_check_answers_bool
3718             {
3719                 \int_gincr:N \g__enumext_item_number_int

```

```

3720     }
3721   }
3722 }

```

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

```

3723 \group_begin:
3724   \lrbox{ \l__enumext_item_text_vii_box }
3725   \bool_if:NF \l__enumext_footnotes_key_bool
3726   {
3727     \__enumext_renew_footnote:
3728   }
3729   \bool_if:NT \l__enumext_item_starred_vii_bool
3730   {
3731     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3732     {
3733       \tl_gset_eq:NN
3734         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3735     }
3736     \mode_leave_vertical:
3737     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3738     \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3739     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3740     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3741   }
3742   \group_begin:
3743     \tl_use:N \l__enumext_label_font_style_vii_tl
3744     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3745     {
3746       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3747         { \__enumext_wrapper_label_vii:n {#1} }
3748     }
3749     {
3750       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3751     }
3752   \group_end:
3753   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3754   \tl_use:N \l__enumext_after_list_args_vii_tl
3755   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3756   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3757   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3758   \tl_use:N \l__enumext_fake_item_indent_vii_tl
3759 }

```

(End of definition for `__enumext_start_item_vii:w`.)

`__enumext_stop_item_vii:` The function `__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its $\langle contents \rangle$. 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.

```

3760 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3761 {
3762   \__enumext_endminipage:
3763   \endlrbox
3764   \group_end:
3765   \box_set_wd:Nn \l__enumext_item_text_vii_box
3766   {
3767     \l__enumext_joined_width_vii_dim
3768     + \l__enumext_labelwidth_vii_dim
3769     + \l__enumext_labelsep_vii_dim
3770   }
3771   \int_set:Nn \hbadness { 10000 }
3772   \box_use:N \l__enumext_item_text_vii_box
3773   \bool_if:NF \l__enumext_footnotes_key_bool
3774   {
3775     \__enumext_print_footnote:
3776   }

```

```

3777 \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3778 {
3779   \par\noindent
3780   \int_zero:N \l__enumext_item_column_pos_vii_int
3781 }
3782 { \hspace{ \l__enumext_columns_sep_vii_dim } }
3783 }

```

(End of definition for `__enumext_stop_item_vii:`)

`__enumext_remove_extra_parsep_vii:` Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

3784 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3785 {
3786   \int_compare:nNnT
3787   {
3788     \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3789   }
3790   =
3791   { \c_zero_int }
3792   {
3793     \par
3794     \vspace{ -\l__enumext_itemsep_vii_skip }
3795     \int_gzero:N \g__enumext_item_count_all_vii_int
3796   }
3797 }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the “hook” function `__enumext_after_env:nn`.

```

3798 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

11.37 The environment `keyans*`

11.37.1 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.

```

3799 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3800 {
3801   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3802   {
3803     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3804     {
3805       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3806       / \l__enumext_columns_viii_int
3807     }
3808   }
3809   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3810   \dim_set:Nn \l__enumext_item_width_viii_dim
3811   {
3812     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3813     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3814     - \l__enumext_labelsep_viii_dim
3815   }
3816   \dim_zero_new:N \itemwidth
3817 }

```

(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`.

```

3818 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3819 {
3820   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3821   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3822   {
3823     \msg_warning:nnee { enumext } { item-joined }
3824     { \int_use:N \l__enumext_joined_item_viii_int }
3825     { \int_use:N \l__enumext_columns_viii_int }
3826     \int_set:Nn \l__enumext_joined_item_viii_int

```

```

3827         {
3828             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3829         }
3830     }
3831     \int_compare:nNnT
3832     { \l__enumext_joined_item_viii_int }
3833     >
3834     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3835     {
3836         \msg_warning:nnee { enumext } { item-joined-columns }
3837         { \int_use:N \l__enumext_joined_item_viii_int }
3838         {
3839             \int_eval:n
3840             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3841         }
3842         \int_set:Nn \l__enumext_joined_item_viii_int
3843         {
3844             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3845         }
3846     }
3847     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3848     {
3849         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3850         \int_decr:N \l__enumext_joined_item_aux_viii_int
3851         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3852         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3853         \dim_set:Nn \l__enumext_joined_width_viii_dim
3854         {
3855             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3856             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3857               + \l__enumext_columns_sep_viii_dim
3858               )*\l__enumext_joined_item_aux_viii_int
3859         }
3860         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3861     }
3862     {
3863         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3864         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3865     }
3866 }

```

(End of definition for `__enumext_starred_joined_item_viii:n`)

`__enumext_start_mini_viii:`
`__enumext_stop_mini_viii:`

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

```

3867 \cs_new_protected:Nn \__enumext_start_mini_viii:
3868 {
3869     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3870     {
3871         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3872         {
3873             \linewidth
3874             - \l__enumext_minipage_right_viii_dim
3875             - \l__enumext_minipage_hsep_viii_dim
3876         }
3877         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3878         \dim_gset_eq:NN
3879         \g__enumext_minipage_right_viii_dim
3880         \l__enumext_minipage_right_viii_dim
3881         \__enumext_mini_addvspace_viii:
3882         \nointerlineskip\noindent
3883         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3884     }
3885 }
3886 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3887 {
3888     \bool_if:NT \l__enumext_minipage_active_viii_bool
3889     {
3890         \end{__enumext_mini_env*}
3891         \hfill
3892         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3893     }

```



```

3894 }
3895 \__enumext_after_env:nn {keyans*}
3896 {
3897   \bool_if:NT \g__enumext_minipage_active_viii_bool
3898   {
3899     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3900     \par\addvspace { \g__enumext_minipage_right_skip }
3901     \bool_if:NF \g__enumext_minipage_center_viii_bool
3902     {
3903       \centering
3904     }
3905     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3906     \end{__enumext_mini_env*}
3907     \par\addvspace{ \g__enumext_minipage_after_skip }
3908   }
3909   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3910   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3911   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3912   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3913 }

```

(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.

```

3914 \NewDocumentEnvironment{keyans*}{ o }
3915 {
3916   \__enumext_safe_exec_viii:
3917   \__enumext_parse_keys_viii:n {#1}
3918   \__enumext_before_list_viii:
3919   \__enumext_start_list:nn { }
3920   {
3921     \__enumext_list_arg_two_viii:
3922     \__enumext_before_keys_exec_viii:
3923   }
3924   \__enumext_starred_columns_set_viii:
3925   \item[] \scan_stop:
3926   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3927   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3928 }
3929 {
3930   \__enumext_stop_item_tmp_viii:
3931   \__enumext_remove_extra_parsep_viii:
3932   \__enumext_check_starred_cmd:n { item }
3933   \__enumext_stop_list:
3934   \__enumext_after_list_viii:
3935 }

```

(End of definition for keyans*. This function is documented on page 13.)

__enumext_safe_exec_viii: First check the maximum nesting level for the **keyans*** environment.

```

3936 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3937 {
3938   \int_incr:N \l__enumext_keyans_level_h_int
3939   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3940   {
3941     \msg_error:nn { enumext } { nested }
3942   }
3943   \__enumext_keyans_save_start_line:
3944   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3945   \bool_set_false:N \l__enumext_store_active_bool
3946   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3947   {
3948     \msg_error:nn { enumext } { keyans-wrong-level }
3949   }
3950 }

```

(End of definition for __enumext_safe_exec_viii:.)

`__enumext_parse_keys_viii:n` Parse [*key* = *val*] for *keyans**

```

3951 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3952 {
3953   \tl_if_novalue:nF {#1}
3954   {
3955     \keys_set:nn { enumext / keyans* } {#1}
3956   }
3957 }

```

(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, then call the function `__enumext_start_mini_viii:` handle by *mini-env*.

```

3958 \cs_new_protected:Nn \__enumext_before_list_viii:
3959 {
3960   \__enumext_vspace_above_viii:
3961   \__enumext_before_args_exec_viii:
3962   \__enumext_start_mini_viii:
3963 }

```

(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.

```

3964 \cs_new_protected:Nn \__enumext_after_list_viii:
3965 {
3966   \__enumext_stop_mini_viii:
3967   \__enumext_after_stop_list_viii:
3968   \__enumext_vspace_below_viii:
3969 }

```

(End of definition for `__enumext_after_list_viii:`.)

11.37.2 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`.

```

3970 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3971 {
3972   \__enumext_stop_item_tmp_viii:
3973   \int_incr:N \l__enumext_item_column_pos_viii_int
3974   \int_gincr:N \g__enumext_item_count_all_viii_int
3975   \__enumext_item_peek_args_viii:
3976 }

```

(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).

```

3977 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3978 {
3979   \peek_meaning:NTF (
3980     { \__enumext_joined_item_viii:w }
3981     { \__enumext_joined_item_viii:w (1) }
3982   )

```

(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`.

```

3983 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3984 {
3985     \__enumext_starred_joined_item_viii:n {#1}
3986     \peek_meaning_remove:NTF *
3987     { \__enumext_starred_item_viii:w }
3988     { \__enumext_standar_item_viii:w }
3989 }

```

(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 [\l__enumext_label_viii_tl]`.

```

3990 \cs_new_protected:Npn \__enumext_standar_item_viii:w
3991 {
3992     \bool_set_false:N \l__enumext_item_starred_viii_bool
3993     \peek_meaning:NTF [
3994     {
3995         \bool_set_eq:NN
3996         \l__enumext_wrap_label_viii_bool
3997         \l__enumext_wrap_label_opt_viii_bool
3998         \__enumext_start_item_viii:w
3999     }
4000     {
4001         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4002         \legacy_if_set_true:n { @noitemarg }
4003         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4004     }
4005 }

```

(End of definition for `__enumext_standar_item_viii:w`.)

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

```

4006 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4007 {
4008     \bool_set_true:N \l__enumext_item_starred_viii_bool
4009     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4010     \peek_meaning:NTF [
4011     { \__enumext_starred_item_viii_aux_i:w }
4012     { \__enumext_starred_item_viii_aux_ii:w }
4013 }

```

The function `__enumext_starred_item_viii_aux_i:w` will save the optional argument to `\item*` in `\l__enumext_keyans_item_opt_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `\l__enumext_store_keyans_label_tl` if present, then call the function `__enumext_starred_item_viii_aux_ii:w`.

```

4014 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4015 {
4016     \tl_clear:N \l__enumext_store_keyans_label_tl
4017     \tl_if_no_value:nF { #1 }
4018     {
4019         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4020         {
4021             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4022             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
4023         }
4024         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
4025     }
4026     \__enumext_starred_item_viii_aux_ii:w
4027 }

```

```

4028 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4029 {
4030   \legacy_if_set_true:n { @noitemarg }
4031   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4032 }

```

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

`__enumext_starred_item_exec:`

The function `__enumext_starred_item_exec:` will be in charge of storing the current *label* for `\item*` followed by the `[content]` for `\item*[content]` if present in the *sequence* and *prop list* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

4033 \cs_new_protected:Nn \__enumext_starred_item_exec:
4034 {
4035   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
4036   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
4037   \__enumext_keyans_store_ref:
4038   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
4039   \__enumext_keyans_addto_seq_link:
4040   \int_gincr:N \g__enumext_check_starred_cmd_int
4041   \bool_if:NT \__enumext_show_answer_bool
4042   {
4043     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
4044   }
4045   \bool_if:NT \__enumext_show_position_bool
4046   {
4047     \tl_set:Ne \__enumext_mark_answer_sym_tl
4048     {
4049       \group_begin:
4050       \exp_not:N \normalfont
4051       \exp_not:N \footnotesize [ \int_eval:n
4052         {
4053           \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
4054         }
4055       ]
4056       \group_end:
4057     }
4058     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
4059   }
4060 }

```

(End of definition for `__enumext_starred_item_exec:`.)

Real definition of `\item in keyans*`

`__enumext_start_item_viii:w`

The implementation at this point is very similar to that of the `enumext*` environment.

```

4061 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4062 {
4063   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4064   \legacy_if:nT { @noitemarg }
4065   {
4066     \legacy_if_set_false:n { @noitemarg }
4067     \legacy_if:nT { @nmbrlist }
4068     {
4069       \bool_if:NT \__enumext_hyperref_bool
4070       {
4071         \legacy_if_set_true:n { @hyper@item }
4072       }
4073       \refstepcounter{enumXviii}
4074     }
4075   }

```

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

```

4076   \group_begin:
4077   \lrbox{ \__enumext_item_text_viii_box }
4078   \bool_if:NF \__enumext_footnotes_key_bool
4079   {
4080     \__enumext_renew_footnote:
4081   }
4082   \bool_if:NT \__enumext_item_starred_viii_bool
4083   {

```

```

4084         \__enumext_starred_item_exec:
4085     }
4086 \group_begin:
4087     \tl_use:N \__enumext_label_font_style_viii_tl
4088     \bool_if:NTF \__enumext_wrap_label_viii_bool
4089     {
4090         \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]
4091         { \__enumext_wrapper_label_viii:n {#1} }
4092     }
4093     {
4094         \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]{ #1
4095     }
4096 \group_end:
4097 \skip_horizontal:N \__enumext_labelsep_viii_dim
4098 \tl_use:N \__enumext_after_list_args_viii_tl
4099 \__enumext_minipage:w [ t ]{ \__enumext_joined_width_viii_dim }
4100 \skip_set_eq:NN \parindent \__enumext_listparindent_viii_dim
4101 \skip_set_eq:NN \parskip \__enumext_parsep_viii_skip
4102 \bool_if:NT \__enumext_item_starred_viii_bool
4103 {
4104     \tl_use:N \__enumext_fake_item_indent_viii_tl
4105     \__enumext_keyans_show_item_opt:
4106     \skip_horizontal:n { -\__enumext_fake_item_indent_viii_dim - \__enumext_labelsep_viii_dim }
4107 }
4108 {
4109     \tl_use:N \__enumext_fake_item_indent_viii_tl
4110 }
4111 }

```

(End of definition for __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The function __enumext_stop_item_viii: shall terminate with the capture of \item and its *(contents)*. Close the environments minipage, lrbox and the group. Then we only have to set the width of the box and print it next to \footnote, and add the horizontal and vertical separation between the boxes.

```

4112 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4113 {
4114     \__enumext_endminipage:
4115     \endlrbox
4116     \group_end:
4117     \box_set_wd:Nn \__enumext_item_text_viii_box
4118     {
4119         \__enumext_joined_width_viii_dim
4120         + \__enumext_labelwidth_viii_dim
4121         + \__enumext_labelsep_viii_dim
4122     }
4123     \int_set:Nn \hbadness { 10000 }
4124     \box_use:N \__enumext_item_text_viii_box
4125     \bool_if:NF \__enumext_footnotes_key_bool
4126     {
4127         \__enumext_print_footnote:
4128     }
4129     \int_compare:nNnTF
4130     { \__enumext_item_column_pos_viii_int } = { \__enumext_columns_viii_int }
4131     {
4132         \par\noindent
4133         \int_zero:N \__enumext_item_column_pos_viii_int
4134     }
4135     { \hspace{ \__enumext_columns_sep_viii_dim } }
4136 }

```

(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.

```

4137 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4138 {
4139     \int_compare:nNnT
4140     {
4141         \int_mod:nn
4142         { \__enumext_item_count_all_viii_int }
4143         { \__enumext_columns_viii_int }

```

```

4144     }
4145     =
4146     { \c_zero_int }
4147     {
4148         \par
4149         \vspace{ -\l__enumext_itemsep_viii_skip }
4150         \int_gzero:N \g__enumext_item_count_all_viii_int
4151     }
4152 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`.)

11.38 The command `\getkeyans`

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

```

4153 \NewDocumentCommand \getkeyans { m }
4154 {
4155     \exp_args:Ne \__enumext_getkeyans_aux:n
4156     { \tl_to_str:e { \text_expand:n {#1} } }
4157 }

```

(End of definition for `\getkeyans`. This function is documented on page 15.)

`__enumext_getkeyans_aux:n`

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

```

4158 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4159 {
4160     \str_if_in:nnTF {#1} { : }
4161     {
4162         \use:e
4163         {
4164             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4165             { {##1} {##2} }
4166         }
4167         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4168     }
4169     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4170 }

```

(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⟩`.

```

4171 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4172 {
4173     \prop_if_exist:cF { g__enumext_#1_prop }
4174     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4175     \group_begin:
4176     \prop_item:cn { g__enumext_#1_prop }{#2}
4177     \group_end:
4178 }

```

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

11.39 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 `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default `⟨keys⟩` for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `\l__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⟩]`

```

4179 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4180 \keys_define:nn { enumext / print }
4181 {
4182     print* .code:n = \keys_precompile:neN { enumext / enumext* }

```

```

4183         { \__enumext_filter_save_key:n {#1} }
4184         \__enumext_print_keyans_starred_tl, % starred cmd
4185     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4186     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4187         { \__enumext_filter_save_key:n {#1} }
4188         \__enumext_print_keyans_i_tl,
4189     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4190     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4191         { \__enumext_filter_save_key:n {#1} }
4192         \__enumext_print_keyans_ii_tl,
4193     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4194     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4195         { \__enumext_filter_save_key:n {#1} }
4196         \__enumext_print_keyans_iii_tl,
4197     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4198     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4199         { \__enumext_filter_save_key:n {#1} }
4200         \__enumext_print_keyans_iv_tl,
4201     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4202     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4203         { \__enumext_filter_save_key:n {#1} }
4204         \__enumext_print_keyans_vii_tl, % starred nested
4205     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4206 }

```

• 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.

`\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`.

```

4207 \NewDocumentCommand \printkeyans { s O{} m }
4208 {
4209     \group_begin:
4210         \tl_use:N \__enumext_print_keyans_i_tl
4211         \tl_use:N \__enumext_print_keyans_ii_tl
4212         \tl_use:N \__enumext_print_keyans_iii_tl
4213         \tl_use:N \__enumext_print_keyans_iv_tl
4214         \tl_use:N \__enumext_print_keyans_vii_tl
4215         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4216     \group_end:
4217 }

```

(End of definition for `\printkeyans`. This function is documented on page 15.)

`__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 if not empty.

```

4218 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4219 {
4220     \seq_if_exist:cTF { g__enumext_#3_seq }
4221     {
4222         \seq_if_empty:cF { g__enumext_#3_seq }
4223         {
4224             %%\seq_show:c { g__enumext_#3_seq }

```

If the starred if it is present we will check that the environment `enumext*` is not saved in the `<sequence>`, then execute the variable `__enumext_print_keyans_starred_tl` that contains the default `<keys>` for the environment `enumext*`, it will open the environment `enumext*` passing the optional argument to the first level and then will map the `<sequence>`

```

4225         \bool_if:nTF {#1}
4226         {
4227             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4228             {
4229                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4230             }
4231         }
4232         \tl_use:N \__enumext_print_keyans_starred_tl
4233         \begin{enumext*}[#2]
4234             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4235         \end{enumext*}

```



```

4236         }
4237     }

```

Otherwise it will open the environment `enumext` passing the optional argument to the first level and then map the `<sequence>`.

```

4238     {
4239         \begin{enumext}[#2]
4240         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4241         \end{enumext}
4242     }
4243 }
4244 }
4245 {
4246     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4247 }
4248 }

```

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

11.40 The command `\setenumext`

First we define a “meta families” of `<keys>` to access from `\setenumext`.

```

4249 \keys_define:nn { enumext / meta-families }
4250 {
4251     enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4252     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4253     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4254     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4255     keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4256     enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4257     keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4258     print*    .code:n = { \keys_set:nn { enumext / print   } { print* = {#1} } } ,
4259     print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
4260     print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
4261     print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
4262     print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
4263     print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
4264     unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4265 }

```

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

```

4266 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4267 {
4268     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4269     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4270 }

```

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

```

4271 \NewDocumentCommand \setenumext { O{enumext,1} +m }
4272 {
4273     \tl_if_novalue:nTF {#1}
4274     {
4275         \seq_map_inline:Nn \c__enumext_all_families_seq
4276     }
4277     {
4278         \seq_clear:N \l__enumext_setkey_tmpa_seq
4279         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4280         \int_set:Nn \l__enumext_setkey_tmpa_int
4281         {
4282             \seq_count:N \l__enumext_setkey_tmpb_seq
4283         }
4284         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4285         {
4286             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4287             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4288             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4289             {
4290                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4291             }
4292         }
4293         {
4294             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }

```

```

4295     }
4296     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4297     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4298     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4299   }
4300   {
4301     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4302   }
4303 }

```

(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.

```

4304 \cs_new_protected:Npn \__enumext_set_parse:n #1
4305 {
4306   \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4307   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4308   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4309   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4310   {
4311     \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
4312     { \tl_trim_spaces:n {#1} }
4313   }
4314   { \__enumext_set_error:nn {#1} { } }
4315 }
4316 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4317 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.41 Messages

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

```

4318 \msg_new:nnn { enumext } { package-load }
4319 {
4320   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4321 }
4322 \msg_new:nnn { enumext } { package-not-load }
4323 {
4324   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4325 }
4326 \msg_new:nnn { enumext } { package-load-foot }
4327 {
4328   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4329 }

```

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

```

4330 \msg_new:nnn { enumext } { counters }
4331 {
4332   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4333   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4334 }

```

Message used in the creation of `(prop list)` by `enumext` package.

```

4335 \msg_new:nnn { enumext } { store-prop }
4336 {
4337   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4338 }
4339 \msg_new:nnn { enumext } { store-seq }
4340 {
4341   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4342 }
4343 \msg_new:nnn { enumext } { store-int }
4344 {
4345   * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4346 }
4347 \msg_new:nnn { enumext } { prop-seq-int-hook }
4348 {
4349   * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4350   * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4351   * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4352 }
4353 \msg_new:nnn { enumext } { item-answer-hook }

```

```

4354 {
4355     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
4356     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
4357     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4358 }

```

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

```

4359 \msg_new:nnn { enumext } { invalid-key }
4360 {
4361     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4362 }
4363 \msg_new:nnn { enumext } { unknown-key-family }
4364 {
4365     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4366 }

```

Messages used in length calculation.

```

4367 \msg_new:nnn { enumext } { width-negative }
4368 {
4369     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4370     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4371 }
4372 \msg_new:nnn { enumext } { width-zero }
4373 {
4374     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4375     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4376 }

```

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

```

4377 \msg_new:nnn { enumext } { list-lengths }
4378 {
4379     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4380     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4381     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4382     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4383     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4384     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4385     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4386     \__enumext_show_length:nnn { skip } { topsep } {#1}
4387     \__enumext_show_length:nnn { skip } { parsep } {#1}
4388     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4389     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4390     ****
4391 }

```

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

```

4392 \msg_new:nnn { enumext } { list-lengths-not-nested }
4393 {
4394     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4395     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4396     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4397     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4398     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4399     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4400     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4401     \__enumext_show_length:nnn { skip } { topsep } {#1}
4402     \__enumext_show_length:nnn { skip } { parsep } {#1}
4403     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4404     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4405     ****
4406 }

```

Messages used by `ref` key.

```

4407 \msg_new:nnn { enumext } { key-ref-empty }
4408 {
4409     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4410 }

```

Messages used by `save-ans` key.

```

4411 \msg_new:nnn { enumext } { save-ans-empty }
4412 {
4413     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4414 }
4415 \msg_new:nnn { enumext } { save-ans-log }

```

```

4416 {
4417     * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
ans=#2 ~ \msg_line_context:.
4418 }
4419 \msg_new:nnn { enumext } { save-ans-log-hook }
4420 {
4421     * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
ans=#2 ~ \msg_line_context:.
4422 }
4423 \msg_new:nnn { enumext } { save-ans-hook }
4424 {
4425     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4426 }

```

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

```

4427 \msg_new:nnn { enumext } { need-save-ans }
4428 {
4429     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4430 }
4431 \msg_new:nnn { enumext } { items-same-answer }
4432 {
4433     *****\\
4434     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_
4435     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\
4436     *****
4437 }
4438 \msg_new:nnn { enumext } { item-greater-answer }
4439 {
4440     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4441     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4442     Items ~ > ~ Answers.
4443 }
4444 \msg_new:nnn { enumext } { item-less-answer }
4445 {
4446     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4447     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4448     Items ~ < ~ Answers.
4449 }

```

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

```

4450 \msg_new:nnn { enumext } { missing-starred }
4451 {
4452     Missing ~ '\c_backslash_str #1*' ~ #2.
4453 }
4454 \msg_new:nnn { enumext } { many-starred }
4455 {
4456     Many ~ '\c_backslash_str #1*' ~ #2.
4457 }

```

Messages used by `\printkeyans*` command.

```

4458 \msg_new:nnn { enumext } { print-starred }
4459 {
4460     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4461     #2 ~ environment ~ \msg_line_context:.
4462 }

```

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

```

4463 \msg_new:nnn { enumext } { list-too-deep }
4464 {
4465     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4466     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4467 }

```

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

```

4468 \msg_new:nnn { enumext } { anskey-empty-arg }
4469 {
4470     Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4471 }
4472 \msg_new:nnn { enumext } { anskey-wrong-place }
4473 {
4474     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4475     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4476 }

```

```

4477 \msg_new:nnn { enumext } { anskey-nested }
4478 {
4479   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4480 }
4481 \msg_new:nnn { enumext } { anskey-nested-env }
4482 {
4483   The ~ environment ~ anskey* ~ can't ~ be ~ nested ~ \msg_line_context:.
4484 }
4485 \msg_new:nnn { enumext } { ansPIC-wrong-place }
4486 {
4487   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4488   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4489 }
4490 \msg_new:nnn { enumext } { command-wrong-place }
4491 {
4492   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4493   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4494 }

```

Messages used by **keyans** and **keyansPIC** environment.

```

4495 \msg_new:nnn { enumext } { keyans-nested }
4496 {
4497   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4498 }
4499 \msg_new:nnn { enumext } { keyans-wrong-level }
4500 {
4501   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4502   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4503 }
4504 \msg_new:nnn { enumext } { wrong-place }
4505 {
4506   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4507   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4508 }
4509 \msg_new:nnn { enumext } { keyansPIC-nested }
4510 {
4511   The ~ environment ~ 'keyansPIC' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4512 }
4513 \msg_new:nnn { enumext } { keyansPIC-wrong-level }
4514 {
4515   Wrong ~ level ~ position ~ for ~ 'keyansPIC' ~ \msg_line_context:~ \\
4516   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4517 }

```

Messages used by **\getkeyans** command.

```

4518 \msg_new:nnn { enumext } { undefined-storage-anskey }
4519 {
4520   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4521 }

```

Messages used by **\miniright** command.

```

4522 \msg_new:nnn { enumext } { missing-miniright }
4523 {
4524   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4525   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4526 }
4527 \msg_new:nnn { enumext } { wrong-miniright-place }
4528 {
4529   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4530   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4531 }
4532 \msg_new:nnn { enumext } { wrong-miniright-use }
4533 {
4534   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4535   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4536 }

```

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

```

4537 \msg_new:nnn { enumext } { nested }
4538 {
4539   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4540 }
4541 \msg_new:nnn { enumext } { item-joined }

```

```
4542 {  
4543     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.  
4544 }  
4545 \msg_new:nnn { enumext } { item-joined-columns }  
4546 {  
4547     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.  
4548 }
```

11.42 Finish package

Finish package implementation.

```
4549 \file_input_stop:  
4550 </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>	2487
<code>*</code>	217
<code>\+</code>	209
<code>\-</code>	209
<code>\\</code>	225, 3344, 4332, 4349, 4350, 4355, 4356, 4369, 4374, 4379, 4394, 4433, 4434, 4435, 4440, 4441, 4446, 4447, 4465, 4474, 4487, 4492, 4501, 4506, 4515, 4524, 4529, 4534
A	
<code>above</code>	1386
<code>above*</code>	1386
<code>\addvspace</code>	1040, 1068, 1184, 1263, 1326, 1332, 1360, 1377, 3183, 3198, 3300, 3315, 3541, 3548, 3900, 3907
<code>after</code>	879
<code>align</code>	496
<code>\Alph</code>	35, 40
<code>\Alph</code>	448, 557, 602, 670, 4201
<code>\alph</code>	35, 40
<code>\alph</code>	449, 555, 4193
<code>\anskey</code>	12, 71, 2218, 2548, 2553
<code>anskey*</code>	13
<code>anskeyenv</code>	2434
<code>\anspic</code>	15, 94, 3322
<code>\anspic*</code>	66
<code>\arabic</code>	29, 35
<code>\arabic</code>	447, 554, 601, 4185, 4189, 4205
B	
<code>\baselineskip</code>	47
<code>\baselineskip</code>	2163, 2171
<code>before</code>	879
<code>before*</code>	879
<code>below</code>	1386
<code>below*</code>	1386
bool commands:	
<code>\bool_gset_false:N</code>	323, 324, 325, 2560, 2562, 3550, 3554, 3909
<code>\bool_gset_true:N</code>	237, 246, 982, 1878, 1884, 3533, 3551, 3892, 3910
<code>\bool_if:NTF</code>	388, 400, 417, 1408, 1422, 1435, 1446, 1457, 1468, 1479, 1490, 1543, 1560, 1565, 1573, 1600, 1638, 1643, 1650, 1654, 1676, 1681, 1689, 1696, 1727, 1735, 1828, 1952, 2036, 2046, 2125, 2149, 2156, 2184, 2222, 2232, 2260, 2286, 2403, 2414, 2418, 2515, 2589, 2604, 2679, 2690, 2694, 2807, 2837, 2911, 2927, 2989, 2999, 3029, 3034, 3117, 3167, 3181, 3189, 3228, 3285, 3298, 3306, 3324, 3529, 3538, 3542, 3619, 3629, 3712, 3717, 3725, 3729, 3744, 3773, 3888, 3897, 3901, 4041, 4045, 4069, 4078, 4082, 4088, 4102, 4125
<code>\bool_if:nTF</code>	1361, 1378, 2848, 2883, 2947, 3345, 4225
<code>\bool_if_p:N</code>	255, 269, 1707, 1708, 1716, 1717, 1841, 1863, 1875, 1876, 1881, 1882, 2271, 2312, 2313, 2337, 2346, 2347, 2359, 2375, 2501, 2666, 2667, 2704, 2705, 3090, 3103, 3105, 3352, 3353
<code>\bool_lazy_all:nTF</code>	253, 267, 1839, 1861, 2335, 2344, 2357, 2373, 3088, 3101
<code>\bool_lazy_and:nnTF</code>	233, 242, 1706, 1715, 1874, 1880, 2270, 2277, 2311, 2500, 2506, 2665
<code>\bool_lazy_or:nnTF</code>	1768, 1775, 2703, 3351
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 60, 70, 91, 96, 97, 102, 103, 106, 127, 130, 133, 134, 143, 144, 145, 153, 154, 168, 179, 181
<code>\bool_not_p:n</code>	234, 243, 2272, 2278, 2362, 2377, 2502, 2507, 3091, 3092, 3104
<code>\bool_set_eq:NN</code>	2815, 2863, 3662, 3995
<code>\bool_set_false:N</code>	397, 1813, 1814, 3204, 3236, 3318, 3383, 3401, 3614, 3659, 3945, 3992
<code>\bool_set_true:N</code>	260, 274, 379, 383, 489, 807, 1392, 1397, 1663, 1785, 1786, 2068, 2076, 2811, 2841, 2859, 2871, 3066, 3097, 3110, 3136, 3233, 3260, 3518, 3588, 3668, 3675, 3676, 3877, 4001, 4008, 4009
box commands:	
<code>\box_dp:N</code>	1080, 1084, 1088, 1099, 1103, 1114, 1123, 1129, 1139, 1152, 1158, 1164, 1195, 1196, 1197, 1200, 1210, 1214, 1223, 1230, 1235, 1243, 1272, 1273, 1276, 1283, 1296, 1304, 1310, 1318, 3413
<code>\box_new:N</code>	67, 174
<code>\box_set_wd:Nn</code>	3765, 4117
<code>\box_use:N</code>	3772, 4124
<code>\box_wd:N</code>	455
C	
<code>\c</code>	217, 218, 707, 709, 721, 723
<code>\cB</code>	218
<code>\cE</code>	218
<code>\centering</code>	1363, 1380, 3434, 3544, 3903
<code>check-ans</code>	1805
Document class:	
<code>article</code>	41
clist commands:	
<code>\clist_const:Nn</code>	186
<code>\clist_map_function:nN</code>	3421
<code>\clist_map_inline:Nn</code>	495, 749, 812, 878, 893, 974, 1402
<code>\clist_map_inline:nn</code>	45, 56, 75, 81, 93, 105, 132, 162, 185, 520, 537, 817, 988, 1508, 1752, 1819, 2015, 2033, 2065, 2332, 2598, 2765, 2976, 2979, 3006, 3016, 3019, 3039, 4307
<code>\columnbreak</code>	72
<code>\columnbreak</code>	2274
<code>columns</code>	958
<code>columns-sep</code>	958
<code>\columnsep</code>	90, 93
<code>\columnsep</code>	3161, 3282
<code>\columnseprule</code>	90, 93
<code>\columnseprule</code>	3165, 3284
Commands provide by enumext:	
<code>\anskey</code>	27, 62, 67, 69, 71, 73-75, 78, 80, 88, 100, 111, 112, 116
<code>\anspic*</code>	27, 28, 66, 69, 77-79, 94-96, 111, 112
<code>\anspic</code>	69, 94-96, 116
<code>\getkeyans</code>	69, 111, 117
<code>\item*</code>	27, 28, 66, 69, 77-79, 82, 83, 102, 108, 109, 111, 112
<code>\itemwidth</code>	97, 104
<code>\item</code>	82, 83, 97, 101, 102, 104, 107, 108
<code>\miniright</code>	26, 45, 52, 53, 89, 91-93, 117
<code>\printkeyans*</code>	111
<code>\printkeyans</code>	27, 69, 111, 112

`\setenumext` 27, 112–115
 Counters defined by `enumext`:
 `enumXiii` 25, 34
 `enumXii` 25, 34
 `enumXiv` 25, 34
 `enumXi` 25, 34
 `enumXviii` 25, 34
 `enumXvii` 25, 34, 102
 `enumXvi` 25, 34
 `enumXv` 25, 34
 cs commands:
 `\cs_generate_variant:Nn` 457, 473, 713, 729, 2117,
 2122, 2202, 2440, 2966, 3423, 4179
 `\cs_if_exist:NTF` 427
 `\cs_new:Nn` 203
 `\cs_new:Npn` . 221, 1509, 1518, 1527, 2080, 2089, 2097
 `\cs_new_eq:NN` 350, 351, 352, 356, 357, 402, 403, 406,
 407
 `\cs_new_protected:Nn` . 213, 227, 251, 282, 309, 315,
 321, 327, 333, 341, 359, 374, 578, 641, 693, 894, 898,
 902, 906, 910, 914, 918, 922, 926, 930, 934, 938, 942,
 946, 950, 954, 989, 1001, 1025, 1042, 1053, 1070, 1145,
 1169, 1186, 1248, 1265, 1287, 1322, 1328, 1403, 1417,
 1431, 1442, 1453, 1464, 1475, 1486, 1571, 1674, 1687,
 1704, 1725, 1753, 1758, 1783, 1824, 1834, 1872, 1887,
 1894, 1903, 1908, 1913, 1918, 1927, 1932, 1937, 1942,
 2123, 2147, 2154, 2182, 2189, 2230, 2323, 2441, 2462,
 2473, 2498, 2540, 2558, 2587, 2602, 2630, 2663, 2699,
 2711, 2719, 2770, 2774, 2793, 2844, 2879, 2895, 2905,
 2921, 3059, 3086, 3115, 3122, 3145, 3175, 3187, 3226,
 3250, 3268, 3293, 3304, 3341, 3385, 3399, 3419, 3424,
 3440, 3508, 3527, 3579, 3601, 3608, 3617, 3627, 3644,
 3784, 3799, 3867, 3886, 3936, 3958, 3964, 3977, 4033,
 4137
 `\cs_new_protected:Npn` 191, 195, 199, 410, 425, 442,
 452, 458, 558, 603, 675, 700, 714, 1350, 1369, 1539,
 1558, 1628, 1661, 1763, 1963, 2034, 2044, 2066, 2074,
 2109, 2118, 2245, 2257, 2400, 2412, 2434, 2566, 2640,
 2684, 2803, 2821, 2855, 2867, 2935, 2969, 3009, 3069,
 3246, 3394, 3459, 3591, 3650, 3657, 3673, 3681, 3686,
 3698, 3818, 3951, 3983, 3990, 4006, 4014, 4028, 4158,
 4171, 4218, 4304, 4316
 `\cs_new_protected_nopar:Nn` ... 3637, 3760, 3970,
 4112
 `\cs_new_protected_nopar:Npn` 3704, 4061
 `\cs_set:Nn` 2405
 `\cs_set:Npn` 2333, 2371, 4164
 `\cs_set_eq:NN` . . 3569, 3570, 3706, 3926, 3927, 4063
 `\cs_set_protected:Nn` 818, 834, 846, 858
 `\cs_set_protected:Npn` . 41, 50, 68, 76, 88, 94, 124,
 158, 166, 474, 496, 525, 538, 585, 730, 750, 794, 813,
 870, 879, 958, 975, 1386, 1497, 1744, 1805, 1980, 2016,
 2052, 2325, 2591, 2754, 2967, 3007
 `\cs_to_str:N` 444, 467
 `\cs_undefine:N` 1957, 1958

D

`\d` 209
`\DeclareDocumentEnvironment` 363
 dim commands:
 `\dim_abs:n` 2940, 2945
 `\dim_add:Nn` 3404
 `\dim_compare:nNnTF` . 820, 836, 848, 860, 1352, 1371,
 2937, 2942, 2948, 2954, 2956, 2958, 3127, 3150, 3254,
 3272, 3396, 3442, 3510, 3801, 3869
 `\dim_compare:nTF` 2296, 2528

`\dim_gset_eq:NN` 3519, 3878
`\dim_gzero:N` 2564, 3553, 3912
`\dim_new:N` 64, 71, 72, 73, 90, 139, 175, 176, 182
`\dim_set:Nn` . . 455, 808, 2835, 2940, 2945, 2947, 2950,
 2951, 2955, 2957, 2960, 2961, 2963, 3130, 3153, 3256,
 3274, 3426, 3444, 3451, 3494, 3512, 3700, 3803, 3810,
 3853, 3871
`\dim_set_eq:NN` 545, 592, 663, 667, 2830, 2978, 3018,
 3161, 3282, 3501, 3504, 3505, 3691, 3860, 3863, 3864
`\dim_use:N` 821, 829, 1353, 1359, 2192, 2195, 2200, 2900,
 2902, 3128, 3133, 3134, 3141, 3151, 3155, 3156, 3158
`\dim_zero:N` 3165, 3284, 3405, 3406, 3407
`\dim_zero_new:N` 3457, 3816
`\c_zero_dim` 823, 837, 849, 861, 1353, 1371, 2298, 2530,
 2937, 2942, 2948, 2955, 3128, 3151, 3254, 3272, 3442,
 3510, 3801, 3869

E

`\end` . . 1356, 1374, 2151, 2186, 3180, 3197, 3297, 3314, 3531,
 3547, 3890, 3906, 4227, 4235, 4241
`\endlist` 32
`\endlist` 351
`\endlrbox` 3763, 4115
`\endminipage` 32
`\endminipage` 357
`enumext` 5, 3040
 enumext internal commands:
 `\l__enumext__check_start_line_env_tl` ... 31
 `\l__enumext__ref_the_count_tl` 37
 `\l__enumext__resume_name_tl` 58
 `__enumext_add_pre_parsep:` . 46, 999, 1001, 1001
 `__enumext_after_args_exec:` . 44, 894, 906, 3052
 `__enumext_after_args_exec_v:` . 44, 45, 910, 922,
 3219
 `__enumext_after_args_exec_vii:` . . . 926, 950
 `__enumext_after_args_exec_viii:` 954
 `__enumext_after_env:nn` 65, 91, 104, 195, 195, 2479,
 3207, 3536, 3798, 3895
 `__enumext_after_hyperref:` . . . 33, 372, 374, 374
 `__enumext_after_list:` . 91, 100, 107, 3057, 3187,
 3187
 `\l__enumext_after_list_args_v_tl` 924
 `\l__enumext_after_list_args_vii_tl` 952, 3754
 `\l__enumext_after_list_args_viii_tl` 956, 4098
 `__enumext_after_list_v:` . . 93, 3224, 3304, 3304
 `__enumext_after_list_vii:` . . . 3577, 3608, 3608
 `__enumext_after_list_viii:` . . 3934, 3964, 3964
 `__enumext_after_star_env:nn` 98
 `__enumext_after_stop_list:` . . . 44, 45, 894, 902,
 3202
 `__enumext_after_stop_list_v:` 44, 910, 918, 3319
 `\l__enumext_after_stop_list_v_tl` 920
 `__enumext_after_stop_list_vii:` 926, 942, 3611
 `\l__enumext_after_stop_list_vii_tl` ... 944
 `__enumext_after_stop_list_viii:` . 946, 3967
 `\l__enumext_after_stop_list_viii_tl` ... 948
 `\l__enumext_align_label_vii_str` . . 3746, 3750
 `\l__enumext_align_label_viii_str` . 4090, 4094
 `\l__enumext_align_label_X_str` 166
 `\c__enumext_all_envs_clist` . . 186, 495, 749, 812,
 878, 893, 974, 1402
 `\c__enumext_all_families_seq` . . 113, 4266, 4275,
 4297
 `__enumext_anskey_env_clean:` . 2441, 2493, 2558
 `__enumext_anskey_env_define_keys:` 2441, 2477

```

\__enumext_anskey_env_exec: .. 2438, 2441, 2473
\__enumext_anskey_env_keys: .. 2441, 2489, 2498
\__enumext_anskey_env_store: . 2441, 2491, 2540
\__enumext_anskey_env_undefine_keys: . 2462,
    2495
\l__enumext_anskey_level_int .. 28, 2251, 2252
\__enumext_anskey_safe_inner:n 71, 2225, 2230,
    2245
\__enumext_anskey_safe_outer: . 71, 2220, 2230,
    2230
\__enumext_anskey_wrapper:n ..... 1984, 2410
\__enumext_at_begin_document:n .. 32, 191, 191,
    348, 354
\__enumext_before_args_exec: 43, 894, 894, 3125
\__enumext_before_args_exec_v: .. 44, 910, 910,
    3253
\__enumext_before_args_exec_vii: .. 926, 926,
    3605
\__enumext_before_args_exec_viii: 930, 3961
\__enumext_before_env:nn ..... 195, 199, 2475
\__enumext_before_keys_exec: 43, 894, 898, 3050
\__enumext_before_keys_exec_v: .. 44, 910, 914,
    3217
\__enumext_before_keys_exec_vii ..... 926
\__enumext_before_keys_exec_vii: 44, 934, 3565
\__enumext_before_keys_exec_viii: .. 44, 938,
    3922
\__enumext_before_list: ... 89, 3044, 3122, 3122
\__enumext_before_list_v: . 92, 3212, 3250, 3250
\__enumext_before_list_vii: ... 100, 3560, 3601,
    3601
\__enumext_before_list_viii: .. 107, 3918, 3958,
    3958
\l__enumext_before_no_starred_key_v_tl 916
\l__enumext_before_no_starred_key_vii_-
    tl ..... 936
\l__enumext_before_no_starred_key_viii_-
    tl ..... 940
\l__enumext_before_starred_key_v_tl ... 912
\l__enumext_before_starred_key_vii_tl . 928
\l__enumext_before_starred_key_viii_tl 932
\__enumext_calc_hspace:NNNNNNN 86, 2935, 2935,
    2966, 2971, 3011
\__enumext_check_ans_active: 63, 89, 1824, 1824,
    3126, 3604
\g__enumext_check_ans_item_tl ..... 79
\g__enumext_check_ans_key_bool 64, 65, 143, 323,
    1878, 1884, 1952
\l__enumext_check_ans_key_bool ... 64, 82, 143,
    1809, 1814, 1875, 1881
\__enumext_check_ans_key_hook: 64, 1872, 1872,
    3201, 3612
\__enumext_check_ans_level: 63, 1824, 1830, 1834
\__enumext_check_ans_log: 64, 65, 1918, 1918, 1956
\__enumext_check_ans_log_msg_greater: 1918,
    1924, 1937
\__enumext_check_ans_log_msg_less: 1918, 1922,
    1927
\__enumext_check_ans_log_msg_same_ok: 1918,
    1923, 1932
\__enumext_check_ans_msg_greater: 1894, 1900,
    1913
\__enumext_check_ans_msg_less: 1894, 1898, 1903
\__enumext_check_ans_msg_same_ok: 1894, 1899,
    1908
\__enumext_check_ans_show: .. 64, 65, 1894, 1894,
    1954
\g__enumext_check_ans_show_bool ..... 91
\l__enumext_check_answers_bool 62, 63, 71, 143,
    1786, 1813, 1828, 2125, 2149, 2156, 2184, 2222, 2679,
    2807, 2837, 3717
\__enumext_check_starred_cmd:n 31, 66, 79, 1963,
    1963, 3222, 3380, 3932
\g__enumext_check_starred_cmd_int 143, 1966,
    1972, 1977, 2877, 3350, 4040
\l__enumext_check_start_line_env_tl 143, 288,
    295, 302, 1969, 1975, 1978
\l__enumext_columns_sep_v_dim 3272, 3274, 3282
\l__enumext_columns_sep_vii_dim .. 3442, 3444,
    3453, 3498, 3782
\l__enumext_columns_sep_viii_dim . 3801, 3803,
    3812, 3857, 4135
\l__enumext_columns_v_int 1191, 3270, 3278, 3290,
    3295
\l__enumext_columns_vii_int .. 3447, 3450, 3454,
    3462, 3466, 3469, 3475, 3481, 3485, 3777, 3788
\l__enumext_columns_viii_int . 3806, 3809, 3813,
    3821, 3825, 3828, 3834, 3840, 3844, 4130, 4143
\l__enumext_counter_i_tl ..... 41, 434
\l__enumext_counter_ii_tl ..... 41, 435
\l__enumext_counter_iii_tl ..... 41, 436
\l__enumext_counter_iv_tl ..... 41, 437
\c__enumext_counter_style_tl ..... 29, 46, 215
\g__enumext_counter_styles_tl . 26, 35, 64, 445,
    463
\l__enumext_counter_v_tl ..... 41, 438, 683
\l__enumext_counter_vi_tl ..... 41, 439
\l__enumext_counter_vii_tl ..... 41, 440, 613
\l__enumext_counter_viii_tl ..... 41, 441, 630
\l__enumext_current_widest_dim 26, 64, 469, 546,
    593, 664, 668
\__enumext_default_item:n ... 2803, 2803, 2852
\__enumext_define_counters:Nn 25, 425, 425, 434,
    435, 436, 437, 438, 439, 440, 441
\__enumext_endminipage: . 32, 354, 357, 369, 3436,
    3762, 4114
\g__enumext_envir_name_tl 30, 148, 261, 275, 331,
    1756, 1761, 1771, 1906, 1911, 1916, 1930, 1935, 1940
\__enumext_execute_after_env: 31, 32, 61, 64, 65,
    1942, 1942, 3207, 3798
\__enumext_fake_item: ..... 818, 818, 2998
\l__enumext_fake_item_indent_v_dim 837, 842
\l__enumext_fake_item_indent_v_tl 839, 2860,
    2864, 2872
\l__enumext_fake_item_indent_vii_dim 849, 854
\l__enumext_fake_item_indent_vii_tl 851, 3758
\l__enumext_fake_item_indent_viii_dim . 861,
    866, 4106
\l__enumext_fake_item_indent_viii_tl .. 863,
    4104, 4109
\l__enumext_fake_item_indent_X_tl ..... 94
\__enumext_fake_item_vii: .... 818, 846, 3028
\__enumext_fake_item_viii: .... 818, 858, 3033
\__enumext_filter_save_key:n .. 68, 2041, 2049,
    2072, 2078, 2080, 2080, 4183, 4187, 4191, 4195, 4199,
    4203
\__enumext_filter_save_key_key:n .. 68, 2080,
    2085, 2089
\__enumext_filter_save_key_pair:nn 68, 2080,
    2086, 2097

```

`__enumext_filter_series:n` [56](#), [1509](#), [1509](#), [1551](#), [1563](#), [1568](#)
`__enumext_filter_series_key:n` [56](#), [1509](#), [1514](#), [1518](#)
`__enumext_filter_series_pair:nn` [56](#), [1509](#), [1515](#), [1527](#)
`\g__enumext_footnote_arg_seq` [163](#), [2776](#), [2789](#), [2799](#)
`\g__enumext_footnote_int` [163](#), [2783](#), [2786](#), [2788](#), [2790](#)
`\g__enumext_footnote_int_seq` [163](#), [2777](#), [2790](#), [2795](#), [2798](#)
`__enumext_footnotes_key_bool` [33](#)
`\l__enumext_footnotes_key_bool` [28](#), [33](#), [103](#), [153](#), [383](#), [388](#), [397](#), [3725](#), [3773](#), [4078](#), [4125](#)
`__enumext_footnotetext:nn` [2770](#), [2770](#), [2800](#)
`__enumext_getkeyans:nn` [111](#), [4167](#), [4171](#), [4171](#)
`__enumext_getkeyans_aux:n` [111](#), [4155](#), [4158](#), [4158](#)
`\l__enumext_hyperref_bool` [28](#), [33](#), [153](#), [379](#), [400](#), [417](#), [2313](#), [2667](#), [3712](#), [4069](#)
`__enumext_hypertarget:nn` [33](#), [374](#), [402](#), [406](#), [422](#)
`__enumext_if_is_int:n` [207](#)
`__enumext_if_is_int:nTF` [207](#), [702](#), [716](#)
`__enumext_internal_mini_page:` [33](#), [75](#), [359](#), [359](#), [3061](#), [3581](#)
`__enumext_is_not_nested:` [30](#), [88](#), [227](#), [227](#), [3062](#), [3582](#)
`__enumext_is_on_first_level:` [30](#), [88](#), [227](#), [251](#), [3067](#), [3589](#)
`\g__enumext_item_anskey_int` [71](#), [79](#), [143](#), [318](#), [345](#), [346](#), [1891](#), [2224](#), [2681](#)
`__enumext_item_answer_diff:` [64](#), [65](#), [1887](#), [1887](#), [1949](#)
`\g__enumext_item_answer_diff_int` [64](#), [65](#), [152](#), [319](#), [1889](#), [1896](#), [1920](#)
`\l__enumext_item_column_pos_vii_int` [101](#), [3469](#), [3475](#), [3481](#), [3485](#), [3492](#), [3640](#), [3777](#), [3780](#)
`\l__enumext_item_column_pos_viii_int` [107](#), [3828](#), [3834](#), [3840](#), [3844](#), [3851](#), [3973](#), [4130](#), [4133](#)
`\l__enumext_item_column_pos_X_int` [166](#)
`\g__enumext_item_count_all_vii_int` [101](#), [3493](#), [3641](#), [3788](#), [3795](#)
`\g__enumext_item_count_all_viii_int` [107](#), [3852](#), [3974](#), [4142](#), [4150](#)
`\g__enumext_item_count_all_X_int` [166](#)
`\g__enumext_item_number_int` [63](#), [143](#), [317](#), [344](#), [346](#), [1845](#), [1849](#), [1852](#), [1855](#), [1867](#), [1891](#), [2809](#), [2839](#), [3719](#)
`__enumext_item_peek_args_vii:` [101](#), [3642](#), [3644](#), [3644](#)
`__enumext_item_peek_args_viii:` [107](#), [3975](#), [3977](#), [3977](#)
`__enumext_item_starred:` [84](#), [2895](#), [2895](#), [2913](#)
`\l__enumext_item_starred_vii_bool` [3659](#), [3675](#), [3729](#)
`\l__enumext_item_starred_viii_bool` [3992](#), [4008](#), [4082](#), [4102](#)
`\l__enumext_item_starred_X_bool` [166](#)
`__enumext_item_std:w` [32](#), [82](#), [83](#), [96](#), [348](#), [352](#), [2812](#), [2818](#), [2842](#), [2860](#), [2864](#), [2872](#), [3417](#)
`\g__enumext_item_symbol_aux_vii_tl` [3683](#), [3731](#), [3734](#), [3738](#), [3740](#)
`\g__enumext_item_symbol_aux_X_tl` [166](#)
`\l__enumext_item_symbol_sep_vii_dim` [3692](#), [3700](#), [3737](#), [3739](#)
`\g__enumext_item_symbol_tl` [25](#), [82](#), [57](#), [2827](#), [2901](#), [2918](#)
`\l__enumext_item_symbol_vii_tl` [3734](#)
`\l__enumext_item_text_vii_box` [3724](#), [3765](#), [3772](#)
`\l__enumext_item_text_viii_box` [4077](#), [4117](#), [4124](#)
`\l__enumext_item_text_X_box` [166](#)
`\l__enumext_item_width_vii_dim` [3451](#), [3496](#), [3504](#), [3505](#)
`\l__enumext_item_width_viii_dim` [3810](#), [3855](#), [3863](#), [3864](#)
`\l__enumext_item_width_X_dim` [166](#)
`\l__enumext_itemindent_X_dim` [68](#)
`\l__enumext_itemsep_vii_skip` [3794](#)
`\l__enumext_itemsep_viii_skip` [4149](#)
`\l__enumext_joined_item_aux_vii_int` [3490](#), [3491](#), [3492](#), [3493](#), [3499](#)
`\l__enumext_joined_item_aux_viii_int` [3849](#), [3850](#), [3851](#), [3852](#), [3858](#)
`\l__enumext_joined_item_aux_X_int` [166](#)
`__enumext_joined_item_vii:w` [101](#), [3647](#), [3648](#), [3650](#), [3650](#)
`\l__enumext_joined_item_vii_int` [3461](#), [3462](#), [3465](#), [3467](#), [3473](#), [3478](#), [3483](#), [3488](#), [3490](#), [3496](#)
`__enumext_joined_item_viii:w` [107](#), [108](#), [3980](#), [3981](#), [3983](#), [3983](#)
`\l__enumext_joined_item_viii_int` [3820](#), [3821](#), [3824](#), [3826](#), [3832](#), [3837](#), [3842](#), [3847](#), [3849](#), [3855](#)
`\l__enumext_joined_item_X_int` [166](#)
`\l__enumext_joined_width_vii_dim` [3494](#), [3501](#), [3504](#), [3755](#), [3767](#)
`\l__enumext_joined_width_viii_dim` [3853](#), [3860](#), [3863](#), [4099](#), [4119](#)
`\l__enumext_joined_width_X_dim` [166](#)
`__enumext_keyans_addto_prop:n` [77](#), [2566](#), [2566](#), [2874](#), [3347](#)
`__enumext_keyans_addto_seq:n` [79](#), [2640](#), [2640](#), [2876](#), [3349](#)
`__enumext_keyans_addto_seq_link:` [2640](#), [2661](#), [2663](#), [4039](#)
`__enumext_keyans_anspic_code:nnn` [94](#), [95](#), [3338](#), [3341](#), [3341](#)
`__enumext_keyans_default_item:n` [83](#), [2855](#), [2855](#), [2891](#)
`\l__enumext_keyans_env_bool` [34](#), [3091](#), [3104](#), [3233](#), [3318](#)
`__enumext_keyans_fake_item:` [818](#), [834](#), [2988](#)
`\l__enumext_keyans_item_opt_tl` [108](#), [106](#), [2688](#), [2701](#), [2707](#), [4024](#)
`\l__enumext_keyans_level_h_int` [28](#), [623](#), [650](#), [2618](#), [3938](#), [3939](#)
`\l__enumext_keyans_level_int` [28](#), [1344](#), [2236](#), [2613](#), [3232](#), [3237](#), [3332](#)
`__enumext_keyans_make_label:` [36](#), [85](#), [2921](#), [2921](#), [2986](#)
`__enumext_keyans_mini_addvspace:` [51](#), [92](#), [1248](#), [1248](#), [3262](#)
`__enumext_keyans_mini_right_cmd:n` [53](#), [1346](#), [1369](#), [1369](#)
`__enumext_keyans_mini_set_vskip:` [50](#), [1186](#), [1186](#), [1250](#)
`__enumext_keyans_multi_addvspace:` [93](#), [1042](#), [1053](#), [3287](#)
`__enumext_keyans_multi_set_vskip:` [47](#), [1042](#), [1042](#), [1055](#)
`__enumext_keyans_multicols_start:` [92](#), [93](#),

3266 , 3268 , 3268	\l__enumext_label_ii_tl 538
__enumext_keyans_multicols_stop : . 93 , 1373 , 3293 , 3293 , 3317	\l__enumext_label_iii_tl 538
__enumext_keyans_parse_keys:n 3211 , 3246 , 3246	\l__enumext_label_iv_tl 538
\l__enumext_keyans_pic_above_int . 138 , 3427 , 3428 , 3430	__enumext_label_style:Nnn 25 , 35 , 458 , 458 , 473 , 543 , 590 , 661 , 665
\l__enumext_keyans_pic_above_skip . . 96 , 138 , 3371 , 3411	\l__enumext_label_v_tl . . 77 , 79 , 658 , 2574 , 2648 , 2713 , 2748 , 2869 , 2873 , 3214 , 3355 , 3357
__enumext_keyans_pic_arg_two : 96 , 3369 , 3399 , 3399	\l__enumext_label_vi_tl . 77 , 79 , 658 , 2571 , 2645 , 3355 , 3357 , 3361
\l__enumext_keyans_pic_below_int . 138 , 3427 , 3428 , 3431	\l__enumext_label_vii_tl . 585 , 3670 , 3695 , 3702
\l__enumext_keyans_pic_body_seq . . 94 – 96 , 138 , 3336 , 3376 , 3435	\l__enumext_label_viii_tl 585 , 4003 , 4031 , 4035
__enumext_keyans_pic_do:n 96 , 3376 , 3378 , 3419 , 3419 , 3423	\l__enumext_label_width_by_box . . 64 , 454 , 455
\l__enumext_keyans_pic_level_int . . 28 , 1336 , 2240 , 2569 , 2608 , 2643 , 2721 , 3387 , 3388	__enumext_label_width_by_box:Nn 35 , 452 , 452 , 457 , 469 , 726
__enumext_keyans_pic_row:n 96 , 3421 , 3424 , 3424	\l__enumext_labelsep_i_dim . . . 2716 , 2751 , 4043 , 4058
__enumext_keyans_pic_safe_exec : . . 95 , 3365 , 3385 , 3385	\l__enumext_labelsep_v_dim 3277
__enumext_keyans_pic_skip_abs:N . . 96 , 3394 , 3394 , 3410	\l__enumext_labelsep_vii_dim . 3446 , 3455 , 3497 , 3693 , 3753 , 3769
\l__enumext_keyans_pic_width_dim . 138 , 3426 , 3433	\l__enumext_labelsep_viii_dim 3805 , 3814 , 3856 , 4097 , 4106 , 4121
__enumext_keyans_redefine_item : . . 84 , 2879 , 2879 , 2985	\l__enumext_labelwidth_i_dim . 2716 , 2751 , 4043 , 4058
__enumext_keyans_ref : 39 , 675 , 693 , 2987	\l__enumext_labelwidth_v_dim 3277
__enumext_keyans_ref:n 39 , 672 , 675 , 675	\l__enumext_labelwidth_vii_dim . . 3446 , 3454 , 3497 , 3746 , 3750 , 3768
__enumext_keyans_safe_exec : . 3210 , 3226 , 3226	\l__enumext_labelwidth_viii_dim . . 3805 , 3813 , 3856 , 4090 , 4094 , 4120
__enumext_keyans_save_start_line : . 31 , 282 , 282 , 3234 , 3392 , 3943	\l__enumext_leftmargin_tmp_v_bool . 96 , 3401
__enumext_keyans_show_ans : . . 2684 , 2692 , 2711	\l__enumext_leftmargin_tmp_X_bool 68
__enumext_keyans_show_item_opt : . 2684 , 2699 , 2872 , 3361 , 4105	\l__enumext_leftmargin_tmp_X_dim 68
__enumext_keyans_show_left:n . 83 , 2684 , 2684 , 2870 , 3356	\l__enumext_leftmargin_X_dim 68
__enumext_keyans_show_pos : . . 2684 , 2696 , 2719	__enumext_level : 203 , 203 , 567 , 570 , 571 , 580 , 582 , 821 , 825 , 829 , 896 , 900 , 904 , 908 , 991 , 993 , 995 , 997 , 1030 , 1032 , 1034 , 1036 , 1040 , 1073 , 1076 , 1095 , 1104 , 1110 , 1115 , 1119 , 1130 , 1134 , 1135 , 1140 , 1176 , 1180 , 1353 , 1359 , 1406 , 1408 , 1410 , 1413 , 1420 , 1422 , 1424 , 1427 , 2036 , 2038 , 2040 , 2068 , 2069 , 2071 , 2127 , 2135 , 2139 , 2143 , 2405 , 2408 , 2409 , 2811 , 2812 , 2816 , 2817 , 2818 , 2825 , 2827 , 2831 , 2832 , 2835 , 2841 , 2842 , 2897 , 2900 , 2902 , 2909 , 2910 , 2911 , 2914 , 2917 , 3047 , 3049 , 3097 , 3110 , 3117 , 3128 , 3130 , 3133 , 3134 , 3136 , 3141 , 3148 , 3151 , 3153 , 3155 , 3156 , 3157 , 3158 , 3161 , 3167 , 3172 , 3178 , 3181 , 3183 , 3189
__enumext_keyans_store_ref : . . 78 , 2587 , 2587 , 2875 , 3348 , 4037	\l__enumext_level_h_int . . 28 , 235 , 257 , 270 , 606 , 643 , 1842 , 1858 , 2361 , 2378 , 3583 , 3584
__enumext_keyans_store_ref_aux_i : 78 , 2587 , 2599 , 2602	\l__enumext_level_int . 88 , 28 , 205 , 244 , 256 , 271 , 361 , 1003 , 1147 , 1340 , 1836 , 1864 , 1944 , 2338 , 2348 , 2354 , 2360 , 2368 , 2376 , 2383 , 3001 , 3063 , 3064 , 3074 , 3081 , 3095 , 3108 , 3163 , 3241 , 3328 , 3621 , 3631 , 3946
__enumext_keyans_store_ref_aux_ii : 79 , 2587 , 2628 , 2630	__enumext_list_arg_two_i : 2967
\l__enumext_keyans_tmpa_tl . 27 , 106 , 2869 , 2873	__enumext_list_arg_two_ii : 2967
__enumext_keyans_wrapper_opt:n . . 1987 , 2707	__enumext_list_arg_two_iii : 2967
\l__enumext_label_copy_i_tl . . 2367 , 2606 , 2611 , 2616 , 2621	__enumext_list_arg_two_iv : 2967
\l__enumext_label_copy_v_tl 2616	__enumext_list_arg_two_v : . 84 , 2967 , 3216 , 3402
\l__enumext_label_copy_vi_tl 2611	__enumext_list_arg_two_vii : 3007 , 3564
\l__enumext_label_copy_vii_tl 2342 , 2353 , 2384 , 2606	__enumext_list_arg_two_viii : 3007 , 3921
\l__enumext_label_copy_viii_tl 2621	\l__enumext_listoffset_v_dim 3279
\l__enumext_label_copy_X_tl 155	\l__enumext_listparindent_vii_dim 3756
\l__enumext_label_fill_left_v_tl 2925	\l__enumext_listparindent_viii_dim . . . 4100
\l__enumext_label_fill_left_X_tl 94	__enumext_log_answer_vars : . 32 , 333 , 341 , 1951
\l__enumext_label_fill_right_v_tl 2932	__enumext_log_global_vars : . 32 , 333 , 333 , 1950
\l__enumext_label_fill_right_X_tl 94	__enumext_make_label : 36 , 82 , 84 , 2905 , 2905 , 2996
\l__enumext_label_font_style_v_tl 2926 , 3360	\l__enumext_mark_answer_sym_tl . 70 , 133 , 1993 ,
\l__enumext_label_font_style_vii_tl . . 3743	
\l__enumext_label_font_style_viii_tl . . 4087	
\l__enumext_label_i_tl 538	

2197, 2420, 2723, 2736, 4047
 \l__enumext_mark_position_str 133, 1997, 1998,
 2021, 2022, 2195
 \l__enumext_mark_ref_sym_tl . . 133, 2007, 2318,
 2675
 __enumext_mini_addvspace: . . 50, 89, 1169, 1169,
 3138
 __enumext_mini_addvspace_vii: 52, 1322, 1322,
 3522
 __enumext_mini_addvspace_viii: 52, 1322, 1328,
 3881
 __enumext_mini_env* 359
 __enumext_mini_right_cmd:n 53, 1348, 1350, 1350
 __enumext_mini_set_vskip: . 48, 1070, 1070, 1171
 __enumext_mini_set_vskip_vii: 51, 1265, 1265,
 1324
 __enumext_mini_set_vskip_viii: 51, 1265, 1287,
 1330
 __enumext_minipage:w 32, 354, 356, 365, 3433, 3755,
 4099
 \l__enumext_minipage_active_v_bool . . 92, 93,
 3260, 3285, 3298, 3306
 \g__enumext_minipage_active_vii_bool . . . 98,
 3533, 3538, 3550
 \l__enumext_minipage_active_vii_bool . 3518,
 3529
 \g__enumext_minipage_active_viii_bool 3892,
 3897, 3909
 \l__enumext_minipage_active_viii_bool 3877,
 3888
 \g__enumext_minipage_active_X_bool . . . 166
 \l__enumext_minipage_active_X_bool 82
 \g__enumext_minipage_after_skip 82, 1269, 1281,
 3548, 3907
 \l__enumext_minipage_after_skip 48, 49, 91, 93,
 82, 1086, 1101, 1121, 1137, 1152, 1158, 1164, 1178,
 1188, 1197, 1200, 1212, 1230, 1241, 1257, 1289, 1302,
 1316, 3198, 3315
 \g__enumext_minipage_center_vii_bool . 3542,
 3551
 \g__enumext_minipage_center_viii_bool 3901,
 3910
 \g__enumext_minipage_center_X_bool . . . 166
 \l__enumext_minipage_hsep_v_dim . . . 92, 3258
 \l__enumext_minipage_hsep_vii_dim . . . 3516
 \l__enumext_minipage_hsep_viii_dim . . . 3875
 \l__enumext_minipage_left_skip 48, 92, 82, 1078,
 1093, 1112, 1127, 1174, 1184, 1189, 1195, 1204, 1221,
 1233, 1253, 1263, 1267, 1272, 1276, 1290, 1294, 1308,
 1326, 1332
 \l__enumext_minipage_left_v_dim 92, 3256, 3264
 \l__enumext_minipage_left_vii_dim 3512, 3524
 \l__enumext_minipage_left_viii_dim 3871, 3883
 \l__enumext_minipage_left_X_dim 82
 \g__enumext_minipage_right_skip 82, 1268, 1273,
 1277, 3541, 3900
 \l__enumext_minipage_right_skip . 48, 82, 1082,
 1097, 1117, 1132, 1190, 1196, 1208, 1226, 1237, 1291,
 1298, 1312, 1360, 1377
 \l__enumext_minipage_right_v_dim . . 92, 1371,
 1376, 3254, 3258
 \g__enumext_minipage_right_vii_dim 98, 3520,
 3540, 3553
 \l__enumext_minipage_right_vii_dim 98, 3510,
 3515, 3521
 \g__enumext_minipage_right_viii_dim . . 3879,
 3899, 3912
 \l__enumext_minipage_right_viii_dim . . 3869,
 3874, 3880
 \g__enumext_minipage_right_X_dim 166
 \g__enumext_minipage_right_X_skip 166
 \g__enumext_minipage_stat_int . 89, 92, 82, 1365,
 1382, 3137, 3191, 3196, 3261, 3308, 3313
 \g__enumext_miniright_code_vii_tl . 99, 3546,
 3552
 \g__enumext_miniright_code_viii_tl 3905, 3911
 \g__enumext_miniright_code_X_tl 166
 __enumext_multi_addvspace: . 47, 90, 1025, 1025,
 3169
 __enumext_multi_set_vskip: . 46, 989, 989, 1027
 \l__enumext_multicols_above_ii_skip . . . 1008
 \l__enumext_multicols_above_iii_skip . . 1014
 \l__enumext_multicols_above_iv_skip . . . 1020
 \l__enumext_multicols_above_v_skip 1044, 1058,
 1068
 \l__enumext_multicols_above_X_skip 76
 \l__enumext_multicols_below_v_skip 1048, 1062,
 3300
 \l__enumext_multicols_below_X_skip 76
 __enumext_multicols_start: . 89, 90, 3143, 3145,
 3145
 __enumext_multicols_stop: 90, 1355, 3175, 3175,
 3200
 __enumext_newlabel:nn 28, 34, 74, 410, 410, 2394,
 2634
 \l__enumext_newlabel_arg_one_tl 28, 34, 74, 78,
 155, 2317, 2387, 2395, 2623, 2635, 2673
 \l__enumext_newlabel_arg_two_tl 28, 34, 73, 155,
 2341, 2351, 2365, 2381, 2396, 2610, 2615, 2620, 2636
 __enumext_parse_keys:n . . . 57, 3043, 3069, 3069
 __enumext_parse_keys_vii:n 57, 3559, 3591, 3591
 __enumext_parse_keys_viii:n . 3917, 3951, 3951
 __enumext_parse_save_key:n 68, 2061, 2066, 2066
 __enumext_parse_save_key_vii:n 68, 2056, 2066,
 2074
 __enumext_parse_serie:n 100
 __enumext_parse_series:n . . 57, 88, 1539, 1539,
 3077, 3597
 __enumext_parse_store_keys:n 88
 \l__enumext_parsep_i_skip 1006, 1008, 1150, 1198
 \l__enumext_parsep_ii_skip . . . 1012, 1014, 1156
 \l__enumext_parsep_iii_skip . . 1018, 1020, 1162
 \l__enumext_parsep_vii_skip 3757
 \l__enumext_parsep_viii_skip 4101
 \l__enumext_partopsep_v_skip . 1060, 1064, 1224,
 1228, 1235, 1239, 1255, 1259
 \l__enumext_partopsep_viii_skip 1300
 __enumext_phantomsection: 33, 374, 403, 407, 423
 __enumext_print_footnote: . . 2770, 2793, 3775,
 4127
 __enumext_print_keyans_box:NN 70, 2189, 2189,
 2202, 2407, 2715, 2750, 4043, 4058
 \l__enumext_print_keyans_i_tl . . . 4188, 4210
 \l__enumext_print_keyans_ii_tl . . 4192, 4211
 \l__enumext_print_keyans_iii_tl . . 4196, 4212
 \l__enumext_print_keyans_iv_tl . . . 4200, 4213
 \l__enumext_print_keyans_starred_tl 111, 112,
 123, 4184, 4232
 \l__enumext_print_keyans_vii_tl 111, 4204, 4214

```

\l__enumext_print_keyans_X_tl . . . . . 123
\__enumext_printkeyans:nnn 112, 4215, 4218, 4218
\__enumext_redefine_item: . 83, 2844, 2844, 2995
\l__enumext_ref_key_arg_tl 37, 46, 218, 560, 561,
    574, 605, 608, 619, 625, 636, 677, 678, 689
\l__enumext_ref_the_count_tl . 37, 46, 567, 570,
    573, 613, 615, 618, 630, 632, 635, 683, 685, 688
\__enumext_regex_counter_style: . . 29, 37, 213,
    213, 568, 614, 631, 684
\__enumext_register_counter_style:Nn . . 442,
    442, 447, 448, 449, 450, 451
\__enumext_remove_extra_parsep_vii: . . 3574,
    3784, 3784
\__enumext_remove_extra_parsep_viii: . 3931,
    4137, 4137
\__enumext_renew_footnote: . . . 2770, 2774, 3727,
    4080
\l__enumext_renew_the_count_v_tl 686, 695, 697
\l__enumext_renew_the_count_vii_tl 616, 645,
    647
\l__enumext_renew_the_count_viii_tl 633, 652,
    654
\l__enumext_renew_the_count_X_tl . . . . . 46
\__enumext_reset_global_bool: . . 309, 312, 321
\__enumext_reset_global_int: . . . 309, 311, 315
\__enumext_reset_global_tl: . . . 309, 313, 327
\__enumext_reset_global_vars: . 31, 65, 309, 309,
    1960
\l__enumext_resume_active_bool 57, 59, 57, 1543,
    1663
\__enumext_resume_counter: . . 58, 59, 1661, 1667,
    1674
\__enumext_resume_counter:n . 57, 59, 1632, 1637,
    1661, 1661, 1731, 1739
\__enumext_resume_counter_save_ans: . . 59, 60,
    1661, 1672, 1704
\__enumext_resume_counter_series: 59, 60, 1661,
    1670, 1687
\g__enumext_resume_int . . . 57, 1584, 1678, 1679
\__enumext_resume_last:n . . 57, 1539, 1545, 1558
\l__enumext_resume_name_tl 57, 1580, 1588, 1591,
    1607, 1615, 1618, 1664, 1665, 1693, 1700
\__enumext_resume_save_counter: 57, 1571, 1571,
    3205, 3615
\__enumext_resume_series:n . 58, 1503, 1628, 1628
\__enumext_resume_starred: . 60, 1504, 1725, 1725
\g__enumext_resume_vii_int . 100, 57, 1611, 1683,
    1684
\__enumext_safe_exec: 33, 75, 88, 3042, 3059, 3059
\__enumext_safe_exec_vii: 33, 75, 3558, 3579, 3579
\__enumext_safe_exec_viii: . . . 3916, 3936, 3936
\__enumext_scontents_anskey:n . 75, 1788, 2434,
    2434, 2440
\__enumext_scontents_anskey_keys: . . . . . 76
\l__enumext_series_name_tl . . . . . 59
\l__enumext_series_str . 58, 88, 1501, 1541, 1549,
    1550, 1552, 1554, 1575, 1578, 1582, 1602, 1605, 1609,
    3073, 3595
\__enumext_set_error:nn . . . . . 4304, 4314, 4316
\__enumext_set_parse:n . . . . . 4287, 4304, 4304
\l__enumext_setkey_tmpa_int . . . 118, 4280, 4284
\l__enumext_setkey_tmpa_seq . . 118, 4278, 4288,
    4294, 4296, 4298, 4311
\l__enumext_setkey_tmpa_tl . . . 118, 4286, 4290
\l__enumext_setkey_tmpb_seq . . 118, 4279, 4282,
    4286, 4287
\l__enumext_setkey_tmpb_tl 118, 4306, 4308, 4309
\l__enumext_show_answer_bool . 133, 2001, 2025,
    2414, 2690, 2704, 3352, 4041
\__enumext_show_length:nnn . . 43, 221, 221, 4380,
    4381, 4382, 4383, 4384, 4385, 4386, 4387, 4388, 4389,
    4395, 4396, 4397, 4398, 4399, 4400, 4401, 4402, 4403,
    4404
\l__enumext_show_position_bool 133, 2004, 2028,
    2418, 2694, 2705, 3353, 4045
\g__enumext_standar_bool 30, 88, 34, 234, 237, 255,
    324, 1573, 1638, 1650, 1676, 1689, 1727, 1863, 1876
\l__enumext_standar_bool . 88, 91, 34, 2346, 2359,
    2375, 3066, 3204
\l__enumext_standar_first_bool 30, 88, 34, 260,
    1560, 1707, 1769, 1776
\__enumext_standar_item_vii:w . 101, 3655, 3657,
    3657
\__enumext_standar_item_viii:w 108, 3988, 3990,
    3990
\__enumext_standar_ref: . . . . 37, 558, 578, 2997
\__enumext_standar_ref:n . . . . 37, 550, 558, 558
\g__enumext_standar_series_tl . 57, 1562, 1563,
    1729, 1732
\g__enumext_starred_bool 30, 99, 100, 34, 243, 246,
    269, 325, 1600, 1643, 1654, 1681, 1696, 1735, 1841,
    1882, 2337, 2347, 2377, 2604, 3092, 3105, 3554
\l__enumext_starred_bool 99, 100, 34, 2272, 2278,
    2362, 2403, 2502, 2507, 3588, 3614
\__enumext_starred_columns_set_vii: . . 3440,
    3440, 3567
\__enumext_starred_columns_set_viii: . 3799,
    3799, 3924
\l__enumext_starred_first_bool . . . 30, 34, 274,
    1565, 1716, 1769, 1776
\__enumext_starred_item:nn . . . 2821, 2821, 2850
\__enumext_starred_item_exec: . 109, 4033, 4033,
    4084
\__enumext_starred_item_vii:w . 101, 102, 3654,
    3673, 3673
\__enumext_starred_item_vii_aux_i:w . 3673,
    3678, 3681
\__enumext_starred_item_vii_aux_ii:w . 3673,
    3679, 3684, 3686
\__enumext_starred_item_vii_aux_iii:w 3673,
    3689, 3698
\__enumext_starred_item_viii:w 108, 3987, 4006,
    4006
\__enumext_starred_item_viii_aux_i:w . . 108,
    4006, 4011, 4014
\__enumext_starred_item_viii_aux_ii:w . 108,
    4006, 4012, 4026, 4028
\__enumext_starred_joined_item_vii:n 97, 101,
    3459, 3459, 3652
\__enumext_starred_joined_item_viii:n . 104,
    108, 3818, 3818, 3985
\__enumext_starred_ref: . . . . 38, 603, 641, 3025
\__enumext_starred_ref:n . . . . 38, 597, 603, 603
\g__enumext_starred_series_tl . 57, 1567, 1568,
    1737, 1740
\__enumext_start_from:NNn 40, 700, 700, 713, 735
\l__enumext_start_i_int . . . . 1679, 1691, 1710
\__enumext_start_item_tmp_vii: 99, 3570, 3637,
    3637

```

__enumext_start_item_tmp_viii: .. 106, 3927, 3970, 3970
 __enumext_start_item_vii:w 101, 102, 3665, 3670, 3695, 3702, 3704, 3704
 __enumext_start_item_viii:w .. 108, 3998, 4003, 4031, 4061, 4061
 \g__enumext_start_line_tl 30, 143, 262, 276, 330, 1906, 1911, 1916, 1930, 1935, 1940
 __enumext_start_list:nn 32, 85, 96, 348, 350, 3046, 3213, 3366, 3562, 3919
 __enumext_start_mini_vii: 100, 3508, 3508, 3606
 __enumext_start_mini_viii: ... 107, 3867, 3867, 3962
 __enumext_start_save_ans_msg: 61, 1753, 1753, 1778
 __enumext_start_store_level: . 88, 3045, 3086, 3086
 __enumext_start_store_level_vii: 100, 3561, 3617, 3617
 \l__enumext_start_vii_int ... 1684, 1698, 1719
 \l__enumext_start_X_int 94, 730
 __enumext_stop_item_tmp_vii: 99, 101, 102, 3569, 3573, 3639, 3706
 __enumext_stop_item_tmp_viii: 106, 107, 3926, 3930, 3972, 4063
 __enumext_stop_item_vii: 102, 103, 3706, 3760, 3760
 __enumext_stop_item_viii: 110, 4063, 4112, 4112
 __enumext_stop_list: .. 32, 348, 351, 3055, 3223, 3379, 3575, 3933
 __enumext_stop_mini_vii: . 98, 100, 3527, 3527, 3610
 __enumext_stop_mini_viii: 107, 3867, 3886, 3966
 __enumext_stop_save_ans_msg: . 61, 1753, 1758, 1948
 __enumext_stop_store_level: .. 88, 3056, 3086, 3115
 __enumext_stop_store_level_vii: . 100, 3576, 3617, 3627
 \l__enumext_store_active_bool . 27, 62, 88, 100, 106, 1708, 1717, 1785, 2232, 3090, 3103, 3228, 3236, 3324, 3383, 3619, 3629, 3945
 __enumext_store_active_keys:n 67, 2034, 2034, 3083
 __enumext_store_active_keys_vii:n . 67, 100, 2034, 2044, 3598
 __enumext_store_addto_prop:n 69, 77, 2109, 2109, 2117, 2259, 2585, 4036
 __enumext_store_addto_seq:n 69, 79, 2118, 2118, 2122, 2129, 2143, 2151, 2160, 2178, 2186, 2321, 2678
 \l__enumext_store_anskey_arg_tl .. 27, 72, 106, 2269, 2274, 2276, 2281, 2288, 2291, 2301, 2306, 2309, 2315, 2321
 __enumext_store_anskey_code:nn . 71, 72, 2226, 2257, 2257
 \l__enumext_store_anskey_env_tl .. 106, 2481, 2484, 2487, 2543, 2548, 2553
 \l__enumext_store_anskey_opt_tl 76, 106, 2482, 2504, 2510, 2517, 2523, 2533, 2545, 2551
 __enumext_store_anskey_safe_outer: 71
 \l__enumext_store_anskey_seq 106
 __enumext_store_anskey_show_left:n 75, 2264, 2412, 2412
 __enumext_store_anskey_show_wrap:n 74, 2400, 2400, 2416, 2431
 \g__enumext_store_columns_break_bool . 2445, 2501, 2560
 \l__enumext_store_columns_break_bool . 2205, 2271
 \l__enumext_store_columns_join_int ... 106
 __enumext_store_internal_ref: .. 72, 73, 2262, 2323, 2323
 \g__enumext_store_item_join_int .. 2448, 2508, 2512, 2561
 \l__enumext_store_item_join_int .. 2208, 2279, 2283
 \g__enumext_store_item_star_bool . 2450, 2515, 2562
 \l__enumext_store_item_star_bool . 2210, 2286
 \g__enumext_store_item_symbol_sep_dim 2455, 2530, 2535, 2564
 \l__enumext_store_item_symbol_sep_dim 2215, 2298, 2303
 \g__enumext_store_item_symbol_tl . 2453, 2521, 2525, 2563
 \l__enumext_store_item_symbol_tl . 2213, 2289, 2293
 \l__enumext_store_keyans_item_opt_sep_tl 1990, 2579, 2581, 2652, 2656, 4019, 4021
 \l__enumext_store_keyans_item_opt_tl .. 106
 \l__enumext_store_keyans_label_tl 27, 77, 79, 108, 106, 2568, 2571, 2574, 2581, 2583, 2585, 2642, 2645, 2648, 2654, 2659, 2669, 2678, 4016, 4021, 4022, 4035, 4036, 4038
 __enumext_store_level_close: . 69, 2123, 2147, 3119
 __enumext_store_level_close_vii: 2154, 2182, 3633
 __enumext_store_level_open: .. 69, 2123, 2123, 3098, 3111
 __enumext_store_level_open_vii: . 2154, 2154, 3623
 \g__enumext_store_name_tl 27, 91, 106, 329, 336, 337, 338, 339, 1761, 1787, 1905, 1910, 1915, 1929, 1934, 1939, 1946, 2484, 2485
 \l__enumext_store_name_tl 27, 61, 63, 106, 1594, 1597, 1621, 1624, 1712, 1721, 1756, 1765, 1766, 1787, 1788, 1789, 1791, 1792, 1794, 1796, 1797, 1799, 1801, 1802, 1826, 2111, 2113, 2120, 2389, 2390, 2426, 2625, 2626, 2729, 2742, 4053
 \l__enumext_store_ref_key_bool 72, 2010, 2260, 2312, 2589, 2666
 \l__enumext_store_save_key_vii_bool .. 2046, 2076
 \l__enumext_store_save_key_vii_tl 2048, 2049, 2077, 2078, 2158, 2168, 2174, 2178
 \l__enumext_store_save_key_X_bool 67
 \l__enumext_store_save_key_X_tl 67, 123
 \l__enumext_store_upper_level_X_bool .. 123
 \l__enumext_store_write_aux_file_tl 28, 74, 79, 155, 2392, 2398, 2632, 2638
 __enumext_storing_exec: 61, 62, 1763, 1779, 1783
 __enumext_storing_set:n .. 61, 1748, 1763, 1763
 \l__enumext_the_counter_v_tl 685
 \l__enumext_the_counter_vii_tl 615
 \l__enumext_the_counter_viii_tl 632
 \l__enumext_the_counter_X_tl 46
 __enumext_tmp:n 41, 45, 50, 56, 68, 75, 76, 81, 88, 93, 94, 105, 124, 132, 158, 162, 166, 185, 813, 817, 1497,

1508, 1744, 1752, 1805, 1823, 1980, 2015, 2016, 2033, 2052, 2065, 2325, 2332, 2333, 2354, 2368, 2371, 2383, 2591, 2598, 2967, 3006, 3007, 3039	<code>\l__enumext_wrap_label_X_bool</code> 94
<code>__enumext_tmp:nn</code> 474, 495, 496, 524, 525, 537, 730, 749, 794, 812, 870, 878, 879, 893, 958, 974, 975, 988, 1386, 1402, 2754, 2769	<code>__enumext_wrapper_label_v:n</code> 2929, 3361
<code>__enumext_tmp:nnn</code> 538, 554, 555, 556, 557, 585, 601, 602	<code>__enumext_wrapper_label_vii:n</code> 3747
<code>__enumext_tmp:nnnnn</code> 750, 775, 778, 781, 783, 785, 788, 791	<code>__enumext_wrapper_label_viii:n</code> 4091
<code>__enumext_tmp:w</code> 4164, 4167	<code>__enumext_zero_parsep:</code> . . . 49, 1090, 1145, 1145
<code>\l__enumext_tmpa_vii_int</code> 3450, 3453	<code>enumext*</code> 5, 3556
<code>\l__enumext_tmpa_viii_int</code> 3809, 3812	<code>enumXi</code> 434
<code>\l__enumext_tmpa_X_int</code> 166	<code>enumXii</code> 434
<code>\l__enumext_topsep_v_skip</code> 1046, 1050, 1193, 1206, 1214, 1219, 1239, 1243, 3382, 3414	<code>enumXiii</code> 434
<code>\l__enumext_topsep_vii_skip</code> . . 1270, 1279, 1283	<code>enumXiv</code> 434
<code>\l__enumext_topsep_viii_skip</code> . 1292, 1314, 1318	<code>enumXv</code> 434
<code>\l__enumext_vspace_a_star_v_bool</code> 1435	<code>enumXvi</code> 434
<code>\l__enumext_vspace_a_star_vii_bool</code> . . . 1457	<code>enumXvii</code> 434
<code>\l__enumext_vspace_a_star_viii_bool</code> . . . 1468	<code>enumXviii</code> 434
<code>\l__enumext_vspace_a_star_X_bool</code> 94	Environments provide by <code>enumext</code> :
<code>__enumext_vspace_above:</code> . . 54, 1403, 1403, 3124	<code>anskeyenv</code> 75
<code>__enumext_vspace_above_v:</code> . 55, 1431, 1431, 3252	<code>enumext*</code> . . 24, 25, 27–30, 33, 34, 37, 38, 40–45, 51, 52, 55–58, 60–63, 65–76, 78, 81, 87, 88, 99, 100, 102, 104, 105, 107, 109, 111, 112, 115, 117
<code>\l__enumext_vspace_above_v_skip</code> . . 1433, 1437, 1439	<code>enumext</code> . 24, 25, 27, 29, 30, 33–41, 43–51, 53, 54, 56–58, 60–63, 65–76, 78, 81–88, 91, 95, 97, 98, 100, 111, 113, 115, 116
<code>__enumext_vspace_above_vii:</code> . . 55, 1453, 1453, 3603	<code>keyans*</code> 24, 25, 27–31, 34, 37–45, 51, 52, 55, 62, 66, 69, 78, 81, 87, 106, 107, 115, 117
<code>\l__enumext_vspace_above_vii_skip</code> 1455, 1459, 1461	<code>keyanspic</code> . . 24, 25, 27, 28, 31, 34, 36, 39, 53, 62, 66, 69, 77–80, 94–96, 117
<code>__enumext_vspace_above_viii:</code> . 55, 1453, 1464, 3960	<code>keyans</code> 24, 25, 27, 28, 30, 31, 34–36, 39–41, 43–45, 47, 50, 51, 53–55, 62, 66, 69, 77–80, 84–86, 91, 92, 94–96, 98, 107, 115, 117
<code>\l__enumext_vspace_above_viii_skip</code> 1466, 1470, 1472	Environments:
<code>\l__enumext_vspace_b_star_v_bool</code> 1446	<code>list</code> 29, 32, 85, 87
<code>\l__enumext_vspace_b_star_vii_bool</code> . . . 1479	<code>lrbox</code> 97, 103, 109, 110
<code>\l__enumext_vspace_b_star_viii_bool</code> . . . 1490	<code>minipage</code> . . . 29, 32, 33, 45, 47, 48, 75, 94–97, 103, 110
<code>\l__enumext_vspace_b_star_X_bool</code> 94	<code>multicols</code> 46–48, 53, 89–93
<code>__enumext_vspace_below:</code> . . 54, 1417, 1417, 3203	<code>scontents</code> 75
<code>__enumext_vspace_below_v:</code> . 55, 1442, 1442, 3320	exp commands:
<code>\l__enumext_vspace_below_v_skip</code> . . 1444, 1448, 1450	<code>\exp_after:wN</code> 4167
<code>__enumext_vspace_below_vii:</code> . . 55, 1475, 1475, 3613	<code>\exp_args:Ne</code> 2547, 2552, 3080, 4155
<code>\l__enumext_vspace_below_vii_skip</code> 1477, 1481, 1483	<code>\exp_not:N</code> . 54, 465, 573, 618, 635, 688, 827, 841, 842, 853, 854, 865, 866, 2317, 2423, 2424, 2671, 2726, 2727, 2739, 2740, 4050, 4051, 4164
<code>__enumext_vspace_below_viii:</code> . 55, 1475, 1486, 3968	<code>\exp_not:n</code> 264, 278, 290, 297, 304, 573, 574, 618, 619, 635, 636, 688, 689, 828, 1525, 1537, 2095, 2107, 2283, 2293, 2303, 2317, 2318, 2395, 2512, 2525, 2535, 2635, 2673, 2675
<code>\l__enumext_vspace_below_viii_skip</code> 1488, 1492, 1494	F
<code>__enumext_widest_from:nNNn</code> . . 40, 714, 714, 729, 741	<code>\fbox</code> 1985
<code>\g__enumext_widest_label_tl</code> 26, 35, 64, 462, 466, 470	file commands:
<code>\l__enumext_wrap_label_opt_v_bool</code> 2863	<code>\file_input_stop:</code> 4549
<code>\l__enumext_wrap_label_opt_vii_bool</code> 101, 3664	<code>first</code> 879
<code>\l__enumext_wrap_label_opt_viii_bool</code> . . 108, 3997	<code>font</code> 474
<code>\l__enumext_wrap_label_opt_X_bool</code> 94	<code>\footnote</code> 81
<code>\l__enumext_wrap_label_v_bool</code> 2859, 2863, 2871, 2927	<code>\footnote</code> 81, 2778
<code>\l__enumext_wrap_label_vii_bool</code> . . 101, 3663, 3668, 3676, 3744	<code>\footnotemark</code> 2788
<code>\l__enumext_wrap_label_viii_bool</code> . 108, 3996, 4001, 4009, 4088	<code>\footnotesize</code> 2424, 2727, 2740, 4051
	<code>\footnotetext</code> 2772
	G
	<code>\getkeyans</code> 15, 111, 4153
	group commands:
	<code>\group_begin:</code> . . 2221, 2422, 2542, 2725, 2738, 3723, 3742, 4049, 4076, 4086, 4175, 4209
	<code>\group_end:</code> 2228, 2429, 2556, 2732, 2745, 3752, 3764, 4056, 4096, 4116, 4177, 4216

H

\hbadness	3771, 4123
hbox commands:	
\hbox_set:Nn	454
\hfill	504, 508, 513, 514, 1357, 1375, 2317, 2671, 3532, 3891
hook commands:	
\hook_gput_code:nnn	9, 193, 197, 201, 372
\hook_gset_rule:nnnn	373
\hspace	3782, 4135
\hyperlink	73, 77, 79
\hyperlink	2317, 2671
\hypertarget	33
\hypertarget	402

I

\IfHyperBoolean	380
\IfPackageLoadedTF	11, 19, 376, 390
\ignorespaces	830
\inputlineno	264, 278, 290, 297, 304
int commands:	
\int_add:Nn	3492, 3851
\int_case:nn	1003, 1147, 1836, 1858, 1896, 1920
\int_compare:nNnTF	361, 606, 623, 643, 650, 1072, 1191, 1336, 1340, 1344, 1944, 1965, 1971, 2236, 2240, 2252, 2569, 2608, 2613, 2618, 2643, 2721, 3064, 3074, 3095, 3108, 3147, 3163, 3177, 3191, 3237, 3241, 3270, 3295, 3308, 3328, 3332, 3388, 3462, 3472, 3488, 3584, 3621, 3631, 3777, 3786, 3821, 3831, 3847, 3939, 3946, 4129, 4139, 4284
\int_compare_p:nNn	235, 244, 256, 257, 270, 271, 1842, 1864, 2279, 2338, 2348, 2360, 2361, 2376, 2378, 2508
\int_decr:N	3491, 3850
\int_eval:n	346, 2113, 2390, 2424, 2626, 2727, 2740, 2982, 3024, 3480, 3839, 4051
\int_from_alph:n	708, 722
\int_from_roman:n	710, 724
\int_gadd:Nn	3493, 3852
\int_gdecr:N	1845, 1849, 1852, 1855, 1867
\int_gincr:N	1678, 1683, 2224, 2681, 2809, 2839, 2877, 3137, 3261, 3350, 3641, 3719, 3974, 4040
\int_gset:Nn	1889, 2786
\int_gset_eq:NN	1577, 1584, 1590, 1596, 1604, 1611, 1617, 1623, 2783
\int_gzero:N	317, 318, 319, 1365, 1382, 1977, 2561, 3196, 3313, 3795, 4150
\int_if_exist:NTF	1552, 1588, 1594, 1615, 1621, 1799
\int_incr:N	2251, 3063, 3232, 3387, 3583, 3640, 3938, 3973
\int_mod:nn	3788, 4141
\int_new:N	28, 29, 30, 31, 32, 33, 57, 58, 82, 98, 113, 120, 140, 141, 149, 150, 151, 152, 163, 169, 170, 171, 172, 173, 1554, 1802
\int_set:Nn	704, 708, 710, 1691, 1698, 1710, 1719, 3427, 3428, 3450, 3461, 3467, 3483, 3771, 3809, 3820, 3826, 3842, 4123, 4280
\int_set_eq:NN	1679, 1684, 3490, 3849
\int_sign:n	1891
\int_step_function:nnN	2354, 2368, 2383
\int_step_inline:nnn	3429
\int_to_roman:n	205, 2334, 2372
\int_use:N	339, 344, 345, 1073, 1693, 1700, 1712, 1721, 2982, 3001, 3024, 3081, 3148, 3157, 3172, 3178, 3465, 3466, 3478, 3824, 3825, 3837
\int_zero:N	3780, 4133
\c_one_int	3450, 3469, 3475, 3481, 3485, 3488, 3809, 3828, 3834, 3840, 3844, 3847

\c_zero_int	2338, 2348, 2360, 2361, 2376, 2378, 3621, 3631, 3791, 4146
\item	32, 44, 45, 70, 82, 94, 96, 97, 99, 106
\item	82, 83, 101, 102, 107, 109, 352, 2131, 2137, 2162, 2170, 2276, 2645, 2648, 2846, 2881, 3568, 3570, 3925, 3927, 4038
\item*	5, 14, 66, 2879
item-pos*	2754
item-sym*	2754
\itemindent	26, 86
\itemindent	85
itemindent	794
\itemsep	95, 96
\itemsep	3403, 3409
\itemwidth	3457, 3501, 3505, 3816, 3860, 3864

K

keyans	13, 3208
keyans*	13, 3914
keyanspic	14, 3363

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

break-col	71, 72, 76
item-join	71, 72, 76
item-pos*	71, 72, 76
item-star	71, 72, 76
item-sym*	71, 72, 76

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

above*	26, 54, 55
above	26, 54, 55, 89, 92, 100, 107
after	43-45, 91, 94, 100, 107
align	26, 36, 84, 103
before*	43, 44, 89, 100, 107
before	43, 44, 92
below*	26, 54, 55
below	26, 54, 55, 91, 94, 100, 107
check-ans	27-30, 61, 62, 64-66, 68, 79, 82, 83, 89, 91, 104, 116
columns-sep	45, 90, 93
columns	26, 45, 48, 54, 90, 93
first	43-45, 103
font	35, 84, 103
item-pos*	81
item-sym*	25, 81, 82
item*-sep	82
itemindent	26, 41, 42, 84, 103
itemsep	41, 87
labelsep	35, 82, 86, 103
labelwidth	35, 37, 39, 40, 86
label	25, 26, 35, 36, 40, 97
lisparindent	87
list-indent	26, 41, 42, 96
list-offset	41, 42
listparindent	41, 103
mark-ans	27, 66, 68, 75
mark-pos	66
mark-ref	27, 66, 68, 73, 77
mini-env	26, 33, 45, 53, 54, 68, 75, 81, 89, 92, 98, 100, 105, 107
mini-right*	26, 45, 68, 99
mini-right	26, 45, 52, 68, 99
mini-rigth*	29
mini-rigth	29
mini-sep	26, 45, 68, 89, 92
no-store	28, 61-63, 68, 71
noitemsep	41, 49

nosep	41, 49
parindent	87
parsep	41, 87, 103
partopsep	41
ref	25, 29, 36, 37, 39, 115
resume*	25, 56, 57, 60–62, 68, 91
resume	25, 32, 56–62, 68, 91, 100
rightmargin	41
save-ans	27, 32, 56–61, 63–65, 67–69, 71, 77, 79, 83, 88, 91, 94, 100, 107, 109, 111, 115
save-key	27, 56, 67, 88
save-pos	68
save-ref	28, 34, 66, 68, 72, 73, 77–79, 84, 109
save-sep	66, 68, 108
series	25, 56–60, 68, 88, 91
show-ans	27, 66, 68, 70, 72, 75, 83, 109
show-length	30, 43, 115
show-pos	27, 66, 70, 72, 75, 80, 83, 109
start	26, 29, 40, 56
store-key	67
topsep	41
widest	26, 29, 40
wrap-ans	66, 68, 70, 74
wrap-label*	35, 82, 84, 101, 103, 108
wrap-label	35, 84, 101, 103, 108
wrap-opt	66, 68
keys commands:	
\keys_define:nn	476, 498, 527, 540, 587, 658, 732, 752, 796, 815, 872, 881, 960, 977, 1388, 1499, 1746, 1807, 1982, 2018, 2054, 2059, 2203, 2443, 2464, 2756, 4180, 4249
\l_keys_key_str	4365
\keys_precompile:nnN	112, 4179, 4182, 4186, 4190, 4194, 4198, 4202
\keys_set:nn	490, 983, 1393, 1398, 1640, 1645, 1732, 1740, 2267, 2551, 3076, 3080, 3248, 3596, 3955, 4251, 4252, 4253, 4254, 4255, 4256, 4257, 4258, 4259, 4260, 4261, 4262, 4263, 4301
keyval commands:	
\keyval_parse:NNn	1513, 2084
L	
label	538, 585, 658
Labels provide by enumext:	
\Alph*	35
\Roman*	35
\alph*	35
\arabic*	29, 35
\roman*	35
\labelsep	96
\labelsep	3404, 3407
labelsep	474
\labelwidth	35, 96
\labelwidth	3404, 3405
labelwidth	474
\leftmargin	26, 86
\leftmargin	85, 3404
legacy commands:	
\legacy_if:nTF	3707, 3710, 4064, 4067
\legacy_if_gset_false:n	366
\legacy_if_set_false:n	3709, 4066
\legacy_if_set_true:n	3669, 3694, 3701, 3714, 4002, 4030, 4071
\linewidth	89, 92
\linewidth	3132, 3258, 3426, 3453, 3514, 3812, 3873

\list	32
\list	350
list-indent	794
list-offset	794
\listparindent	3406
listparindent	794
\lrbox	3724, 4077

M

\makebox	97
\makebox	2193, 2195, 2901, 3738, 3746, 3750, 4090, 4094
\makelabel	82, 84, 85, 97
\makelabel	84, 85, 2907, 2923
\makesavenoteenv	396
mark-ans	1980
mark-pos	1980, 2016
mark-ref	1980
mini-env	958
mini-sep	958
\minipage	32
\minipage	356
\miniright	10, 52, 1334, 3194, 3311
\miniright*	10
mode commands:	
\mode_if_vertical:TF	1028, 1056, 1172, 1251
\mode_leave_vertical:	827, 841, 853, 865, 2162, 2170, 2191, 2899, 3736
msg commands:	
\msg_error:nn	2249, 2254, 3239, 3243, 3330, 3390, 3586, 3941, 3948, 4264
\msg_error:nnn	563, 610, 627, 680, 1338, 1342, 1367, 1384, 1652, 1656, 1771, 4169, 4174, 4246, 4317
\msg_error:nnnn	2234, 2238, 2242, 3230, 3326, 3334, 4229
\msg_fatal:nn	3065
\msg_fatal:nnn	428
\msg_info:nnn	13, 16, 21, 24, 378, 392
\msg_line_context:	4337, 4341, 4345, 4369, 4374, 4379, 4394, 4409, 4413, 4417, 4421, 4425, 4429, 4435, 4441, 4447, 4461, 4465, 4470, 4474, 4479, 4483, 4487, 4492, 4497, 4501, 4506, 4511, 4515, 4520, 4524, 4529, 4534, 4539, 4543, 4547
\msg_log:nnn	1791, 1796, 1801
\msg_log:nnnnn	343, 1929, 1934, 1939
\msg_log:nnnnnn	335
\msg_new:nnn	4318, 4322, 4326, 4330, 4335, 4339, 4343, 4347, 4353, 4359, 4363, 4367, 4372, 4377, 4392, 4407, 4411, 4415, 4419, 4423, 4427, 4431, 4438, 4444, 4450, 4454, 4458, 4463, 4468, 4472, 4477, 4481, 4485, 4490, 4495, 4499, 4504, 4509, 4513, 4518, 4522, 4527, 4532, 4537, 4541, 4545
\msg_term:nnnn	1755, 1760, 2991, 3001, 3030, 3035
\msg_term:nnnnn	1910
\msg_warning:nn	3193, 3310
\msg_warning:nnnn	1968, 1974, 2939, 2944, 3464, 3477, 3823, 3836
\msg_warning:nnnnn	1905, 1915
\multicolsep	90, 93
\multicolsep	3162, 3283

N

\NeedsTeXFormat	3
\newcounter	431
\NewDocumentCommand	1334, 2218, 3322, 4153, 4207, 4271
\NewDocumentEnvironment	3040, 3208, 3363, 3556, 3914

<code>\newenvsc</code>	2437
<code>\newlabel</code>	34
<code>\newlabel</code>	414
<code>no-store</code>	1805
<code>\noindent</code>	99, 106
<code>\noindent</code>	3139, 3263, 3523, 3569, 3779, 3882, 3926, 4132
<code>\nointerlineskip</code>	3139, 3263, 3523, 3882
<code>noitemsep</code>	750
<code>\nopagebreak</code>	1039, 1067, 1183, 1262, 1325, 1331
<code>\normalfont</code>	2423, 2726, 2739, 4050
<code>nosep</code>	750

P

Packages:

<code>enumext</code>	24, 36, 61, 85, 94, 114
<code>enumitem</code>	34, 35
<code>expl3</code>	97
<code>footnotehyper</code>	33
<code>hyperref</code>	28, 29, 33, 34, 73, 77, 79, 102, 114
<code>lua-visual-debug</code>	48
<code>multicol</code>	24, 114
<code>scontents</code>	24
<code>shortlst</code>	97
<code>\par</code>	1039, 1067, 1183, 1262, 1325, 1331, 1360, 1377, 2402, 3183, 3198, 3300, 3315, 3438, 3541, 3548, 3779, 3793, 3900, 3907, 4132, 4148
<code>\parindent</code>	3756, 4100
<code>\parsep</code>	46, 49, 95, 96
<code>\parsep</code>	2163, 2171, 3021, 3403, 3410, 3415
<code>parsep</code>	750
<code>\parskip</code>	3757, 4101
<code>\partopsep</code>	96
<code>\partopsep</code>	3022, 3408
<code>partopsep</code>	750
peek commands:	
<code>\peek_meaning:N</code>	3646, 3660, 3677, 3688, 3979, 3993, 4010
<code>\peek_meaning_remove:N</code>	3653, 3986
<code>\peek_remove_spaces:n</code>	2885
<code>\phantomsection</code>	33
<code>\phantomsection</code>	403
prg commands:	
<code>\prg_do_nothing:</code>	407
<code>\prg_new_protected_conditional:Npnn</code>	207
<code>\prg_replicate:nn</code>	224
<code>\prg_return_false:</code>	211
<code>\prg_return_true:</code>	210
<code>\printkeyans</code>	15, 111, 4207
prop commands:	
<code>\prop_count:N</code>	337, 2113, 2390, 2426, 2626, 2729, 2742, 4053
<code>\prop_gput_if_not_in:Nnn</code>	2111
<code>\prop_if_exist:N</code>	1789, 4173
<code>\prop_item:Nn</code>	4176
<code>\prop_new:N</code>	1792
<code>\ProvidesExplPackage</code>	4

R

<code>\raggedcolumns</code>	3171, 3289
<code>\ref</code>	73, 78
<code>ref</code>	538, 585, 658
<code>\refstepcounter</code>	3716, 4073
regex commands:	
<code>\regex_match:nnTF</code>	209, 707, 709, 721, 723
<code>\regex_replace_all:nnN</code>	2487

<code>\regex_replace_once:nnN</code>	217
<code>\renewcommand</code>	573, 618, 635, 688
<code>\RenewDocumentCommand</code>	2778, 2846, 2881, 2907, 2923
<code>\RequirePackage</code>	17, 25
<code>resume</code>	1497
<code>resume*</code>	1497
<code>rightmargin</code>	794
<code>\Roman</code>	35, 40
<code>\Roman</code>	450
<code>\roman</code>	35, 40
<code>\roman</code>	451, 556, 4197

S

<code>\s</code>	2487
<code>save-ans</code>	1744
<code>save-key</code>	2052
<code>save-ref</code>	1980
<code>save-sep</code>	1980
scan commands:	
<code>\scan_stop:</code>	96, 3417, 3568, 3925, 4164, 4167
scontents internal commands:	
<code>__scontents_anskeyenv_env_begin:</code>	1957
<code>__scontents_anskeyenv_env_end:</code>	1958
<code>__scontents_rescan_tokens:n</code>	2548, 2553
seq commands:	
<code>\seq_clear:N</code>	4278
<code>\seq_const_from_clist:Nn</code>	4266
<code>\seq_count:N</code>	338, 3376, 4282
<code>\seq_gclear:N</code>	2776, 2777
<code>\seq_gpop_right:NNTF</code>	2483
<code>\seq_gput_right:Nn</code>	2120, 2789, 2790
<code>\seq_if_empty:N</code>	2795, 4222, 4296
<code>\seq_if_exist:N</code>	1794, 4220
<code>\seq_if_in:NnTF</code>	4227
<code>\seq_item:Nn</code>	2485, 3435
<code>\seq_map_function:NN</code>	4287
<code>\seq_map_inline:Nn</code>	4234, 4240, 4275, 4297, 4298
<code>\seq_map_pairwise_function:NNN</code>	2797
<code>\seq_new:N</code>	109, 121, 122, 138, 164, 165, 1797
<code>\seq_pop_left:NN</code>	4286
<code>\seq_put_right:Nn</code>	3336, 4294, 4311
<code>\seq_set_from_clist:Nn</code>	4279
<code>\seq_set_map_e:NNn</code>	4288
<code>\seq_show:N</code>	4224
<code>series</code>	1497
<code>\setcounter</code>	718, 722, 724, 2982, 3024, 3381
<code>\setenumext</code>	6, 113, 4271
<code>\setlength</code>	2164, 2172
<code>show-ans</code>	1980, 2016
<code>show-length</code>	870
<code>show-pos</code>	2016
skip commands:	
<code>\skip_add:Nn</code>	1008, 1014, 1020, 1030, 1034, 1058, 1062, 1152, 1158, 1164, 1174, 1178, 1200, 1253, 1257, 3403
<code>\skip_eval:n</code>	2163, 2171
<code>\skip_gset:Nn</code>	1273, 1277, 1281
<code>\skip_gzero_new:N</code>	1268, 1269
<code>\skip_horizontal:N</code>	842, 854, 866, 3739, 3753, 4097
<code>\skip_horizontal:n</code>	828, 2192, 2200, 2900, 2902, 3737, 4106
<code>\skip_if_eq:nnTF</code>	1006, 1012, 1018, 1075, 1109, 1150, 1156, 1162, 1193, 1198, 1219, 1270, 1292, 1405, 1419, 1433, 1444, 1455, 1466, 1477, 1488
<code>\skip_new:N</code>	78, 79, 83, 84, 85, 86, 87, 142, 183

\skip_set:Nn	991, 995, 1044, 1048, 1078, 1082, 1086, 1093, 1097, 1101, 1112, 1117, 1121, 1127, 1132, 1137, 1195, 1196, 1197, 1204, 1208, 1212, 1221, 1226, 1230, 1233, 1237, 1241, 1272, 1276, 1294, 1298, 1302, 1308, 1312, 1316, 3397, 3411
\skip_set_eq:NN	2980, 3020, 3021, 3756, 3757, 4100, 4101
\skip_use:N	993, 997, 1032, 1036, 1040, 1060, 1064, 1076, 1095, 1104, 1110, 1115, 1119, 1130, 1134, 1135, 1140, 1176, 1180, 1206, 1406, 1410, 1413, 1420, 1424, 1427, 3183
\skip_zero:N	3022, 3162, 3283, 3408, 3409
\skip_zero_new:N	1188, 1189, 1190, 1267, 1289, 1290, 1291
\c_zero_skip	1006, 1012, 1018, 1076, 1110, 1150, 1156, 1162, 1193, 1198, 1219, 1270, 1292, 1406, 1420, 1433, 1444, 1455, 1466, 1477, 1488
\small	4185, 4189, 4193, 4197, 4201, 4205
\star	2760
start	730
\stepcounter	2782, 3343
str commands:	
\c_backslash_str	4337, 4341, 4345, 4349, 4350, 4351, 4355, 4356, 4452, 4456, 4460, 4474, 4475, 4479, 4487, 4488, 4492, 4493, 4524, 4525, 4529, 4534, 4535
\c_colon_str	2389, 2625, 4164
\c_left_brace_str	4417, 4421, 4434, 4440, 4446
\c_right_brace_str	4417, 4421, 4434, 4440, 4446
\str_case:nn	229, 284
\str_case:nnTF	1520, 1529, 2091, 2099
\str_clear:N	3073, 3595
\str_count:n	224
\str_if_empty:NTF	1541, 1582, 1609
\str_if_eq:nnTF	2983, 3026
\str_if_in:nnTF	4160
\str_new:N	137, 178
\str_set:Nn	530, 531, 532, 1997, 1998, 2021, 2022
\string	396
\strutbox	1080, 1084, 1088, 1099, 1103, 1114, 1123, 1129, 1139, 1152, 1158, 1164, 1195, 1196, 1197, 1200, 1210, 1214, 1223, 1230, 1235, 1243, 1272, 1273, 1276, 1283, 1296, 1304, 1310, 1318, 3413

T

TeX and L^AT_EX 2_ε commands:

\@auxout	412
\@currentenv	229, 284
\protected@write	412

text commands:

\text_expand:n	4156
\textasteriskcentered	1994, 2008
\thepage	418

tl commands:

\c_space_tl	2707, 4379, 4394, 4417, 4421
\tl_clear:N	503, 509, 1978, 2038, 2048, 2069, 2077, 2269, 2481, 2482, 2568, 2642, 4016
\tl_clear_new:N	460
\tl_const:Nn	46, 444
\tl_gclear:N	329, 330, 331, 1562, 1567, 2563, 2918, 3552, 3740, 3911
\tl_gclear_new:N	1549
\tl_gput_right:Nn	445
\tl_greplace_all:Nnn	466
\tl_gset:Nn	261, 262, 275, 276, 1550, 1563, 1568, 1787, 3683
\tl_gset_eq:NN	462, 2827, 3733
\tl_if_blank:nTF	3731

\tl_if_empty:NTF	561, 580, 608, 625, 645, 652, 678, 695, 1575, 1580, 1602, 1607, 1665, 1729, 1737, 1766, 1826, 1946, 2127, 2158, 2289, 2521, 2543, 2545, 2579, 2652, 2701, 2897, 4019, 4309
\tl_if_empty:nTF	1630, 2247
\tl_if_exist:NTF	1635
\tl_if_novalue:nTF	2265, 2576, 2650, 2686, 2780, 2805, 2823, 2828, 2857, 3071, 3374, 3593, 3953, 4017, 4273
\tl_map_inline:Nn	215, 463
\tl_new:N	43, 48, 49, 52, 53, 59, 61, 62, 63, 65, 66, 99, 100, 101, 107, 108, 110, 111, 112, 114, 115, 116, 117, 118, 119, 123, 126, 128, 129, 135, 136, 146, 147, 148, 155, 156, 157, 160, 177, 180
\tl_put_left::Ne	2510
\tl_put_left:Nn	2135, 2168, 2274, 2504, 2517, 2523, 2533, 2713, 2748, 4035, 4038
\tl_put_right:Nn	461, 571, 616, 633, 686, 2139, 2174, 2276, 2281, 2288, 2291, 2301, 2306, 2309, 2315, 2341, 2351, 2365, 2381, 2387, 2392, 2571, 2574, 2581, 2583, 2610, 2615, 2620, 2623, 2632, 2645, 2648, 2654, 2659, 2669, 4021, 4022
\tl_remove_all:Nn	4308
\tl_remove_once:Nn	2329, 2595
\tl_replace_all:Nnn	465
\tl_reverse:N	2328, 2330, 2594, 2596
\tl_set:Nn	54, 288, 295, 302, 430, 504, 508, 513, 514, 560, 605, 677, 825, 839, 851, 863, 1664, 1765, 2039, 2049, 2070, 2078, 2420, 2688, 2723, 2736, 2825, 4024, 4047, 4306
\tl_set_eq:NN	471, 566, 569, 613, 615, 630, 632, 683, 685, 2327, 2593, 2606, 2869, 2873, 3355, 3357
\tl_to_str:n	1635, 1641, 1646, 4156
\tl_trim_spaces:n	461, 4294, 4306, 4312
\tl_use:N	467, 470, 582, 647, 654, 697, 896, 900, 904, 908, 912, 916, 920, 924, 928, 932, 936, 940, 944, 948, 952, 956, 2197, 2334, 2342, 2353, 2367, 2372, 2384, 2812, 2818, 2842, 2860, 2864, 2872, 2909, 2910, 2917, 2925, 2926, 2932, 3047, 3214, 3360, 3546, 3743, 3754, 3758, 3905, 4087, 4098, 4104, 4109, 4210, 4211, 4212, 4213, 4214, 4232, 4290

token commands:

\token_to_str:N	414
\topsep	2164, 2172
topsep	750
\typeout	382, 385, 395, 396

U

\u	218, 2487
use commands:	
\use:N	225, 2914, 3049
\use:n	1511, 2082, 4162
\use_none:nn	406
\usecounter	2981, 3023

V

\value	1578, 1584, 1591, 1597, 1605, 1611, 1618, 1624
\vspace	367, 1410, 1413, 1424, 1427, 1437, 1439, 1448, 1450, 1459, 1461, 1470, 1472, 1481, 1483, 1492, 1494, 2163, 2171, 3371, 3382, 3794, 4149

W

widest	730
wrap-ans	1980
wrap-label	474
wrap-label*	474
wrap-opt	1980