

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	118
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 / 130

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.

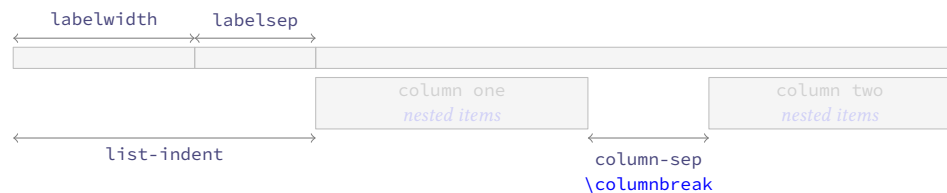


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 L^AT_EX, `\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`

<code>\setenumext</code>	<code>\setenumext{\langle key = val \rangle}</code>	<code>\setenumext[\langle keyans* \rangle]{\langle key = val \rangle}</code>
	<code>\setenumext[\langle enumext, level \rangle]{\langle key = val \rangle}</code>	<code>\setenumext[\langle print, level \rangle]{\langle key = val \rangle}</code>
	<code>\setenumext[\langle enumext* \rangle]{\langle key = val \rangle}</code>	<code>\setenumext[\langle print, * \rangle]{\langle key = val \rangle}</code>
	<code>\setenumext[\langle keyans \rangle]{\langle key = val \rangle}</code>	<code>\setenumext[\langle print* \rangle]{\langle key = val \rangle}</code>

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 \TeX 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.

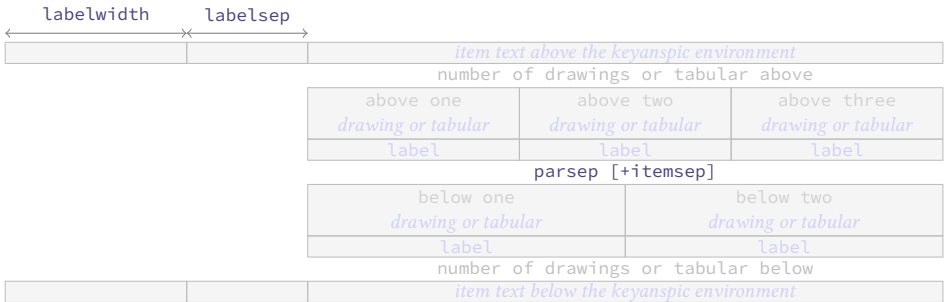


Figure 6: Representation of the keyanspic environment with optional argument [3,2] in enumext.

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 \LaTeXe\ is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1]

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

(a)

~~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

3. Third type of questions

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

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

A

value

B

value

C

value

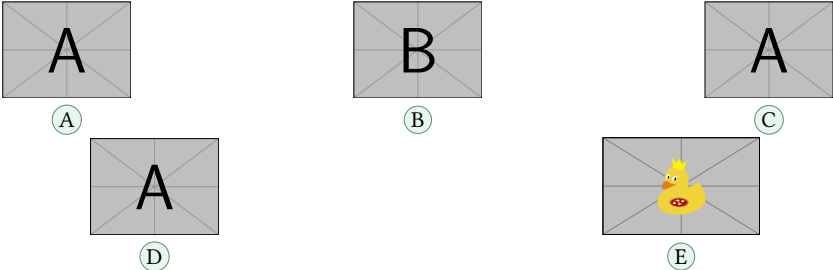
D

value

E

value

4. Question with image and label below:



5. Question with image on left side:
- A

value
- B

value
- C

value
- D

correct
- E

value



Test keys

1. B, $x = 5$

2. D

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

5. D, other note

Example 4

A “simple worksheet” using ducks :) 🦆.



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

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.

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:

article

2

book

2

exam

2

letter

2

report

2

\columnbreak

4, 12

\columnsep

10

Commands provide by enumext:

\anskey

4, 10–13

\anspic*

4, 10–12, 14, 15

\anspic

14, 15

\getkeyans

4, 12, 15

\item*

4–7, 10–14

\itemwidth

5

\item

5–7, 9, 10, 12, 13

\miniright

4, 10

\printkeyans

4, 6, 11, 15

\setenumext

4–7, 11, 13, 14, 16

Counters defined by enumext:

enumXiii

4

enumXii

4

enumXiv

4

enumXi

4

enumXviii

4

enumXvii

4

enumXvi

4

enumXv

4

E

Environments provide by enumext:

anskey*

13

enumext*

4–16

enumext

4–16, 19

keyans*

4–14

keyanspic

4, 6, 8, 10–14, 19

keyans

4–15, 19

Environments:

enumerate

1, 3, 5, 20

figure

5

list

3, 9, 20

minipage

3–5, 10, 20

multicols

3, 4, 10

table

5

task

5

F

\footnote

5

I

\item

3, 5

\itemsep

8

K

Keys for command provide by enumext:

break-col

12

item-join

12

item-pos*

12

item-star

12

item-sym*

12

L

Keys for environments provide by enumext:

above*

8

above

8

after

9, 10

align

7, 20

before*

9

before

9

below*

8

below

8

check-ans

12

columns-sep

4, 10

columns

4, 8, 10

first

9

font

7

item-pos*

5, 6

item-sym*

5, 6

itemindent

8

itemsep

8, 15

labelsep

3, 5–10, 12, 19, 20

labelwidth

3, 6, 7, 9, 10, 12, 19, 20

labelwith

5

label

7, 9, 14, 19, 20

list-indent

3, 9

list-offset

3, 9, 20

listparindent

9

mark-ans

11, 12

mark-pos

12

mark-ref

11

mini-env

4, 8, 10

mini-right*

6, 10

mini-right

6, 10

mini-sep

4, 10

no-store

11–13

noitemsep

8

nosep

8, 19

parsep

8, 15

partopsep

8

ref

4, 7

resume*

6, 9, 10

resume

6, 9, 10

rightmargin

8

save-ans

4, 6, 9–15

save-key

9, 11, 16

save-ref

4, 7, 11–13, 15

save-sep

11

series

6, 9, 10

show-ans

11

show-length

7

show-pos

11, 12, 15

start

9

topsep

8

widest

7

wrap-ans

11

wrap-label*

7, 20

wrap-label

7

wrap-opt

11

\label

4

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 { multicol }
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_arg_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 \tl_new:N \l__enumext_store_anskey_arg_tl
110 \tl_new:N \l__enumext_store_anskey_opt_tl
111 \int_new:N \l__enumext_store_columns_join_int
112 \tl_new:N \l__enumext_store_keyans_label_tl
113 \tl_new:N \l__enumext_store_keyans_item_opt_tl
114 \tl_new:N \l__enumext_keyans_item_opt_tl
115 \tl_new:N \l__enumext_keyans_tmpa_tl
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpp_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmppb_seq
```

Internal variables used by the command `\setenumext` (§11.40).

```
116 \tl_new:N \l__enumext_setkey_tmpa_tl
117 \tl_new:N \l__enumext_setkey_tmppb_tl
118 \int_new:N \l__enumext_setkey_tmpa_int
119 \seq_new:N \l__enumext_setkey_tmpa_seq
120 \seq_new:N \l__enumext_setkey_tmppb_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.

```
121 \tl_new:N \l__enumext_print_keyans_starred_tl
122 \cs_set_protected:Npn \__enumext_tmp:n #1
123 {
124   \tl_new:c { \l__enumext_store_save_key_#1_tl }
125   \bool_new:c { \l__enumext_store_save_key_#1_bool }
126   \tl_new:c { \l__enumext_store_active_keys_#1_tl }
127   \tl_new:c { \l__enumext_print_keyans_#1_tl }
128   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
129 }
130 \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).

```
131 \bool_new:N \l__enumext_show_answer_bool
132 \bool_new:N \l__enumext_show_position_bool
133 \tl_new:N \l__enumext_mark_ref_sym_tl
134 \tl_new:N \l__enumext_mark_answer_sym_tl
135 \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).

```
136 \seq_new:N \l__enumext_keyans_pic_body_seq
137 \dim_new:N \l__enumext_keyans_pic_width_dim
138 \int_new:N \l__enumext_keyans_pic_above_int
139 \int_new:N \l__enumext_keyans_pic_below_int
140 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

```

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

```

Internal variables used by “*check answer*” mechanism (§11.23.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

141 \bool_new:N \l__enumext_check_answers_bool
142 \bool_new:N \l__enumext_check_ans_key_bool
143 \bool_new:N \g__enumext_check_ans_key_bool
144 \tl_new:N \l__enumext_check_start_line_env_tl
145 \tl_new:N \g__enumext_start_line_tl
146 \tl_new:N \g__enumext_envir_name_tl
147 \int_new:N \g__enumext_check_starred_cmd_int
148 \int_new:N \g__enumext_item_anskey_int
149 \int_new:N \g__enumext_item_number_int
150 \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`.

```

151 \bool_new:N \l__enumext_hyperref_bool
152 \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.

```

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

```

161 \int_new:N \g__enumext_footnote_int
162 \seq_new:N \g__enumext_footnote_arg_seq
163 \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.

```

164 \cs_set_protected:Npn \__enumext_tmp:n #1
165 {
166   \bool_new:c { l__enumext_item_starred_#1_bool }
167   \int_new:c { l__enumext_item_column_pos_#1_int }
168   \int_new:c { g__enumext_item_count_all_#1_int }
169   \int_new:c { l__enumext_joined_item_#1_int }
170   \int_new:c { l__enumext_joined_item_aux_#1_int }
171   \int_new:c { l__enumext_tmpa_#1_int }
172   \box_new:c { l__enumext_item_text_#1_box }
173   \dim_new:c { l__enumext_joined_width_#1_dim }
174   \dim_new:c { l__enumext_item_width_#1_dim }
175   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
176   \str_new:c { l__enumext_align_label_#1_str }
177   \bool_new:c { g__enumext_minipage_active_#1_bool }
178   \tl_new:c { g__enumext_miniright_code_#1_tl }
179   \bool_new:c { g__enumext_minipage_center_#1_bool }
180   \dim_new:c { g__enumext_minipage_right_#1_dim }
181   \skip_new:c { g__enumext_minipage_right_#1_skip }
182 }
183 \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`.

```
184 \clist_const:Nn \c__enumext_all_envs_clist
185 {
186   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
187   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
188 }
```

(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.

```
189 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
190 {
191   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
192 }
```

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

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

```
193 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
194 {
195   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
196 }
```

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

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

```
197 \cs_new:Nn \__enumext_level:
198 {
199   \int_to_roman:n { \__enumext_level_int }
200 }
```

(End of definition for `__enumext_level:`.)

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

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

```
201 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
202 {
203   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
204   { \prg_return_true: }
205   { \prg_return_false: }
206 }
```

(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.

```
207 \cs_new_protected:Nn \__enumext_regex_counter_style:
208 {
209   \tl_map_inline:Nn \c__enumext_counter_style_tl
210   {
211     \regex_replace_once:nnN { \c{##1}\* }
212     { \c{##1}\cB{\u{ \__enumext_ref_the_count_tl }\cE} } \__enumext_ref_key_arg_tl
213   }
214 }
```

(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.

```

215 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
216 {
217   * ~ #2
218   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
219   = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
220 }
```

(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.

```

221 \cs_new_protected:Nn \__enumext_is_not_nested:
222 {
223   \str_case:en { \@currentvir }
224   {
225     {enumext}
226     {
227       \bool_lazy_and:nnT
228       { \bool_not_p:n { \g__enumext_standar_bool } }
229       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
230       {
231         \bool_gset_true:N \g__enumext_standar_bool
232       }
233     }
234     {enumext*}
235     {
236       \bool_lazy_and:nnT
237       { \bool_not_p:n { \g__enumext_starred_bool } }
238       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
239       {
240         \bool_gset_true:N \g__enumext_starred_bool
241       }
242     }
243   }
244 }
```

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.

```

245 \cs_new_protected:Nn \__enumext_is_on_first_level:
246 {
247   \bool_lazy_all:nT
248   {
249     { \bool_if_p:N \g__enumext_standar_bool }
250     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
251     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
252   }
253   {
254     \bool_set_true:N \l__enumext_standar_first_bool
255     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
256     \tl_gset:Nn \g__enumext_start_line_tl
257     {
258       on ~ line ~ \exp_not:V \inputlineno
259     }
260   }
261   \bool_lazy_all:nT
262   {
263     { \bool_if_p:N \g__enumext_starred_bool }
264     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
265     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
266   }
267   {
268     \bool_set_true:N \l__enumext_starred_first_bool
269     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
270     \tl_gset:Nn \g__enumext_start_line_tl
```

```

271         {
272             on ~ line ~ \exp_not:V \inputlineno
273         }
274     }
275 }

```

(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.

```

276 \cs_new_protected:Nn \__enumext_keyans_save_start_line:
277 {
278     \str_case:en { \@currenvir }
279     {
280         {keyans}
281         {
282             \tl_set:Nc \l__enumext_check_start_line_env_tl
283             {
284                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
285             }
286         }
287         {keyans*}
288         {
289             \tl_set:Nc \l__enumext_check_start_line_env_tl
290             {
291                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
292             }
293         }
294         {keyanspic}
295         {
296             \tl_set:Nc \l__enumext_check_start_line_env_tl
297             {
298                 in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
299             }
300         }
301     }
302 }

```

(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.

```

303 \cs_new_protected:Nn \__enumext_reset_global_vars:
304 {
305     \__enumext_reset_global_int:
306     \__enumext_reset_global_bool:
307     \__enumext_reset_global_tl:
308 }
309 \cs_new_protected:Nn \__enumext_reset_global_int:
310 {
311     \int_gzero:N \g__enumext_item_number_int
312     \int_gzero:N \g__enumext_item_anskey_int
313     \int_gzero:N \g__enumext_item_answer_diff_int
314 }
315 \cs_new_protected:Nn \__enumext_reset_global_bool:
316 {
317     \bool_gset_false:N \g__enumext_check_ans_key_bool
318     \bool_gset_false:N \g__enumext_standar_bool
319     \bool_gset_false:N \g__enumext_starred_bool
320 }
321 \cs_new_protected:Nn \__enumext_reset_global_tl:
322 {
323     \tl_gclear:N \g__enumext_store_name_tl
324     \tl_gclear:N \g__enumext_start_line_tl
325     \tl_gclear:N \g__enumext_envir_name_tl
326 }

```

(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.

```

327 \cs_new_protected:Nn \__enumext_log_global_vars:
328 {
329   \msg_log:nneeee { enumext } { prop-seq-int-hook }
330   { \g__enumext_store_name_tl }
331   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
332   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
333   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
334 }

```

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.

```

335 \cs_new_protected:Nn \__enumext_log_answer_vars:
336 {
337   \msg_log:nneeee { enumext } { item-answer-hook }
338   { \int_use:N \g__enumext_item_number_int }
339   { \int_use:N \g__enumext_item_anskey_int }
340   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
341 }

```

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

```

342 \__enumext_at_begin_document:n
343 {
344   \cs_new_eq:NN \__enumext_start_list:nn \list
345   \cs_new_eq:NN \__enumext_stop_list: \endlist
346   \cs_new_eq:NN \__enumext_item_std:w \item
347 }

```

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

```

348 \__enumext_at_begin_document:n
349 {
350   \cs_new_eq:NN \__enumext_minipage:w \minipage
351   \cs_new_eq:NN \__enumext_endminipage: \endminipage
352 }

```

(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)

```

353 \cs_new_protected:Nn \__enumext_internal_mini_page:
354 {
355   \int_compare:nNtT { \__enumext_level_int } = { 0 }
356   {
357     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
358     {
359       \__enumext_minipage:w [ t ] { ##1 }
360       \legacy_if_gset_false:n { @minipage }
361       \vspace { 0pt }
362     }
363     { \__enumext_endminipage: }
364   }
365 }
```

(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.

```

366 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
367 \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”.

```

368 \cs_new_protected:Nn \__enumext_after_hyperref:
369 {
370   \IfPackageLoadedTF { hyperref }
371   {
372     \msg_info:nnn { enumext } { package-load } { hyperref }
373     \bool_set_true:N \l__enumext_hyperref_bool
374     \IfHyperBoolean{hyperfootnotes}
375     {
376       \typeout{hyperfootnotes=true}
377       \bool_set_true:N \l__enumext_footnotes_key_bool
378     }
379     { \typeout{hyperfootnotes=false} }
380   }
381   { }
```

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`.

```

382 \bool_if:NT \l__enumext_footnotes_key_bool
383 {
384   \IfPackageLoadedTF { footnotehyper }
385   {
386     \msg_info:nnn { enumext } { package-load } { footnotehyper }
387   }
388   {
389     \typeout{No ~ footnotehyper ~ load}
390     \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
391     \bool_set_false:N \l__enumext_footnotes_key_bool
392   }
393 }
```

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.

```

394 \bool_if:NTF \l__enumext_hyperref_bool
395 {
396   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
397   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
398 }
399 {
400   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
401   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
402 }
403 }

```

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

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

#1: `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

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

```

404 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
405 {
406   \protected@write \@auxout { }
407   {
408     \token_to_str:N \newlabel {#1}
409     {
410       {#2}
411       \bool_if:NT \l__enumext_hyperref_bool
412       { { \thepage } {#2} {#1} }
413       { }
414     }
415   }
416   \__enumext_hypertarget:nn {#1} { }
417   \__enumext_phantomsection:
418 }

```

(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 `\l__enumext_counter_X_tl` for “store” the counter’s name.

#2: The counter’s name.

```

419 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
420 {
421   \cs_if_exist:cTF { c@ #2 }
422   { \msg_fatal:nnn { enumext } { counters } { #2 } }
423   {
424     \tl_set:Nn #1 { #2 }
425     \newcounter { #2 }
426   }
427 }

```

(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 428 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXv 429 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXvi 430 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii 431 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii 432 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
433 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
434 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
435 \__enumext_define_counters:Nn \l__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.

```

436 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
437 {
438   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
439   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
440 }
441 \__enumext_register_counter_style:Nn \arabic { 0 }
442 \__enumext_register_counter_style:Nn \Alph { M }
443 \__enumext_register_counter_style:Nn \alph { m }
444 \__enumext_register_counter_style:Nn \Roman { VIII }
445 \__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.

```

446 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
447 {
448   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
449   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
450 }
451 \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.

```

452 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
453 {
454   \tl_clear_new:N #1
455   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
456   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
457   \tl_map_inline:Nn \g__enumext_counter_styles_tl
458   {
459     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
460     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
461     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
462   }
463   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
464   { \tl_use:N \g__enumext_widest_label_tl }
465   \tl_set_eq:cN { the #2 } #1
466 }
467 \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.

```

468 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
469 {
470   \keys_define:nn { enumext / #1 }
471   {
472     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
473     font      .value_required:n = true,
474     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
475     labelsep  .initial:n   = {0.3333em},
476     labelsep  .value_required:n = true,
477     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
478     labelwidth .value_required:n = true,

```

```

479     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
480     wrap-label .initial:n = {##1},
481     wrap-label .value_required:n = true,
482     wrap-label* .code:n = {
483         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
484         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
485     },
486     wrap-label* .value_required:n = true,
487 }
488 }
489 \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.

```

490 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
491 {
492     \keys_define:nn { enumext / #1 }
493     {
494         align .choice:,
495         align / left .code:n =
496             {
497                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
498                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
499             },
500         align / right .code:n =
501             {
502                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
503                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
504             },
505         align / center .code:n =
506             {
507                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
508                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
509             },
510         align .initial:n = left,
511         align .value_required:n = true,
512     }
513 }
514 \clist_map_inline:nn
515 {
516     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
517 }
518 { \__enumext_tmp:nn #1 }

519 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
520 {
521     \keys_define:nn { enumext / #1 }
522     {
523         align .choice:,
524         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
525         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
526         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
527         align .initial:n = left,
528         align .value_required:n = true,
529     }
530 }
531 \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
ref `labelwidth` key and `ref` key.

```

\l__enumext_label_i_tl 532 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 533 {
\l__enumext_label_iii_tl 534 \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 535 {
536     label .code:n = {
537         \__enumext_label_style:cvn { l__enumext_label_#2_tl }
538         { l__enumext_counter_#2_tl } {##1}
539         \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
540         \l__enumext_current_widest_dim
541     },
542     label .initial:n = #3,
543     label .value_required:n = true,
544     ref .code:n = \__enumext_standar_ref:n {##1},
545     ref .value_required:n = true,
546 }
547 }
548 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
549 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
550 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
551 \__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_-
__enumext_standar_ref: 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.

```

552 \cs_new_protected:Npn \__enumext_standar_ref:n #1
553 {
554     \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
555     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
556     {
557         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
558     }
559     {
560         \tl_set_eq:Nc
561         \__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
562         \__enumext_regex_counter_style:
563         \tl_set_eq:Nc
564         \__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
565         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
566         {
567             \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
568             { \exp_not:V \__enumext_ref_key_arg_tl }
569         }
570     }
571 }

```

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

```

572 \cs_new_protected:Nn \__enumext_standar_ref:
573 {
574     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
575     {
576         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
577     }
578 }

```

(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
ref for `labelwidth` key and `ref` key.

```

\l__enumext_label_vii_tl 579 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 580 {

```

```

581 \keys_define:nn { enumext / #1 }
582 {
583   label .code:n = {
584     \__enumext_label_style:cvn { l__enumext_label_#2_tl }
585     { l__enumext_counter_#2_tl } {##1}
586     \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
587     \l__enumext_current_widest_dim
588   },
589   label .initial:n = #3,
590   label .value_required:n = true,
591   ref .code:n = \__enumext_starred_ref:n {##1},
592   ref .value_required:n = true,
593 }
594 }
595 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
596 \__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:`

```

597 \cs_new_protected:Npn \__enumext_starred_ref:n #1
598 {
599   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
600   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
601   {
602     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
603     {
604       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
605     }
606     {
607       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
608       \__enumext_regex_counter_style:
609       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
610       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
611       {
612         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
613         { \exp_not:V \l__enumext_ref_key_arg_tl }
614       }
615     }
616   }
617   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
618   {
619     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
620     {
621       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
622     }
623     {
624       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
625       \__enumext_regex_counter_style:
626       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
627       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
628       {
629         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
630         { \exp_not:V \l__enumext_ref_key_arg_tl }
631       }
632     }
633   }
634 }

```

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.

```

635 \cs_new_protected:Npn \__enumext_starred_ref:
636 {
637   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
638   {
639     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
640     {
641       \tl_use:N \l__enumext_renew_the_count_vii_tl
642     }
643   }

```



```

644 \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
645 {
646   \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
647   {
648     \tl_use:N \l__enumext_renew_the_count_viii_tl
649   }
650 }
651 }

```

(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
652 \keys_define:nn { enumext / keyans }
653 {
654   label .code:n = {
655     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
656     { \l__enumext_counter_v_tl } {#1}
657     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
658     \l__enumext_current_widest_dim
659     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
660     { \l__enumext_counter_vi_tl } {#1}
661     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
662     \l__enumext_current_widest_dim
663   },
664   label .initial:n = \Alph*,
665   label .value_required:n = true,
666   ref .code:n = \l__enumext_keyans_ref:n {#1},
667   ref .value_required:n = true,
668 }

```

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

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

```

669 \cs_new_protected:Npn \l__enumext_keyans_ref:n #1
670 {
671   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
672   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
673   {
674     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
675   }
676   {
677     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
678     \__enumext_regex_counter_style:
679     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
680     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
681     {
682       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
683       { \exp_not:V \l__enumext_ref_key_arg_tl }
684     }
685   }
686 }

```

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

```

687 \cs_new_protected:Nn \l__enumext_keyans_ref:
688 {
689   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
690   {
691     \tl_use:N \l__enumext_renew_the_count_v_tl
692   }
693 }

```

(End of definition for `\l__enumext_keyans_ref:n` and `\l__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.

```
694 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
695 {
696   \__enumext_if_is_int:nTF { #3 }
697   {
698     \int_set:Nn #2 {#3}
699   }
700   {
701     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
702     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
703     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
704     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
705   }
706 }
707 \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.

```
708 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
709 {
710   \__enumext_if_is_int:nTF {#4}
711   {
712     \setcounter{enumX#1} { #4 }
713   }
714   {
715     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
716     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
717     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
718     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
719   }
720   \__enumext_label_width_by_box:cv
721   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
722 }
723 \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.

```
724 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
725 {
726   \keys_define:nn { enumext / #1 }
727   {
728     start .code:n = {
729       \__enumext_start_from:ccn
730       { \l__enumext_label_#2_tl }
731       { \l__enumext_start_#2_int } {##1}
732     },
733     start .initial:n = 1,
734     widest .code:n = {
735       \__enumext_widest_from:nccn {#2}
736       { \l__enumext_label_#2_tl }
737       { \l__enumext_labelwidth_#2_dim } {##1}
738     },

```

```

739         widest .value_required:n = true,
740         start .value_required:n = true,
741     }
742 }
743 \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
744 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
745 {
746     \keys_define:nn { enumext / #1 }
747     {
748         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
749         topsep .initial:n = {#3},
750         topsep .value_required:n = true,
751         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
752         partopsep .initial:n = {#4},
753         partopsep .value_required:n = true,
754         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
755         parsep .initial:n = {#5},
756         parsep .value_required:n = true,
757         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
758         itemsep .initial:n = {#6},
759         itemsep .value_required:n = true,
760         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
761         noitemsep .value_forbidden:n = true,
762         nosep .meta:n = {
763             itemsep = 0pt, parsep = 0pt,
764             topsep = 0pt, partopsep = 0pt,
765         },
766         nosep .value_forbidden:n = true,
767     }
768 }

```

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

```

769 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
770 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
771 { 4.0pt plus 2.0pt minus 1.0pt }
772 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
773 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
774 { 2.0pt plus 1.0pt minus 1.0pt }
775 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
776 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
777 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
778 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
779 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
780 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
781 { 2.0pt plus 1.0pt minus 1.0pt }
782 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
783 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
784 { 4.0pt plus 2.0pt minus 1.0pt }
785 \__enumext_tmp:nnnnnn { keyans* } { viii } { 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 }

```

(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.

```

788 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
789 {
790     \keys_define:nn { enumext / #1 }
791     {
792         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
793         itemindent .value_required:n = true,
794         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },

```

```

795     rightmargin .value_required:n = true,
796     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
797     listparindent .value_required:n = true,
798     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
799     list-offset .value_required:n = true,
800     list-indent .code:n =
801         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
802         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
803     list-indent .value_required:n = true,
804 }
805 }
806 \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.

```

807 \cs_set_protected:Npn \__enumext_tmp:n #1
808 {
809     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
810 }
811 \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.

```

812 \cs_set_protected:Nn \__enumext_fake_item:
813 {
814     \dim_compare:nNnT
815     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
816     >
817     { \c_zero_dim }
818     {
819         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
820         {
821             \exp_not:N \mode_leave_vertical:
822             \exp_not:n { \skip_horizontal:n }
823             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
824             \ignorespaces
825         }
826     }
827 }
828 \cs_set_protected:Nn \__enumext_keyans_fake_item:
829 {
830     \dim_compare:nNnT
831     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
832     {
833         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
834         {
835             \exp_not:N \mode_leave_vertical:
836             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
837         }
838     }
839 }
840 \cs_set_protected:Nn \__enumext_fake_item_vii:
841 {
842     \dim_compare:nNnT
843     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
844     {
845         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
846         {
847             \exp_not:N \mode_leave_vertical:
848             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
849         }
850     }
851 }
852 \cs_set_protected:Nn \__enumext_fake_item_viii:
853 {
854     \dim_compare:nNnT

```

```

855     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
856     {
857         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
858         {
859             \exp_not:N \mode_leave_vertical:
860             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
861         }
862     }
863 }

```

(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.

```

864 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
865 {
866     \keys_define:nn { enumext / #1 }
867     {
868         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
869         show-length .initial:n = false,
870     }
871 }
872 \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.

```

873 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
874 {
875     \keys_define:nn { enumext / #1 }
876     {
877         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
878         before .value_required:n = true,
879         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
880         before* .value_required:n = true,
881         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
882         after .value_required:n = true,
883         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
884         first .value_required:n = true,
885     }
886 }
887 \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.

```

888 \cs_new_protected:Nn \__enumext_before_args_exec:
889 {
890     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
891 }

```

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 $\langle keys \rangle$.

```

892 \cs_new_protected:Nn \__enumext_before_keys_exec:
893 {
894     \tl_use:c { \l__enumext_before_no_starred_key_ \__enumext_level: _tl }
895 }

```

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

```
896 \cs_new_protected:Nn \__enumext_after_stop_list:
897 {
898     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
899 }
```

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

```
900 \cs_new_protected:Nn \__enumext_after_args_exec:
901 {
902     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
903 }
```

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

```
904 \cs_new_protected:Nn \__enumext_before_args_exec_v:
905 {
906     \tl_use:N \l__enumext_before_starred_key_v_tl
907 }
```

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

```
908 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
909 {
910     \tl_use:N \l__enumext_before_no_starred_key_v_tl
911 }
```

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

```
912 \cs_new_protected:Nn \__enumext_after_stop_list_v:
913 {
914     \tl_use:N \l__enumext_after_stop_list_v_tl
915 }
```

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`.

```
916 \cs_new_protected:Nn \__enumext_after_args_exec_v:
917 {
918     \tl_use:N \l__enumext_after_list_args_v_tl
919 }
```

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

```
920 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
921 {
922     \tl_use:N \l__enumext_before_starred_key_vii_tl
923 }
924 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
925 {
926     \tl_use:N \l__enumext_before_starred_key_viii_tl
927 }
```

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

```
928 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
929 {
930     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
931 }
932 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
933 {
```



```

934     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
935   }

```

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

```

936 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
937 {
938   \tl_use:N \l__enumext_after_stop_list_vii_tl
939 }
940 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
941 {
942   \tl_use:N \l__enumext_after_stop_list_viii_tl
943 }

```

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`.

```

944 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
945 {
946   \tl_use:N \l__enumext_after_list_args_vii_tl
947 }
948 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
949 {
950   \tl_use:N \l__enumext_after_list_args_viii_tl
951 }

```

(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.

```

952 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
953 {
954   \keys_define:nn { enumext / #1 }
955   {
956     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
957     mini-env .value_required:n = true,
958     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
959     mini-sep .initial:n = 0.3333em,
960     mini-sep .value_required:n = true,
961     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
962     columns-sep .value_required:n = true,
963     columns .int_set:c = { l__enumext_columns_#2_int },
964     columns .initial:n = 1,
965     columns .value_required:n = true,
966   }
967 }
968 \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.

```

969 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
970 {
971   \keys_define:nn { enumext / #1 }
972   {
973     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
974     mini-right .value_required:n = true,
975     mini-right* .code:n = {
976       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
977       \keys_set:nn { enumext / #1 } { miniright = {##1} }
978     },
979     mini-right* .value_required:n = true,
980   }
981 }
982 \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.

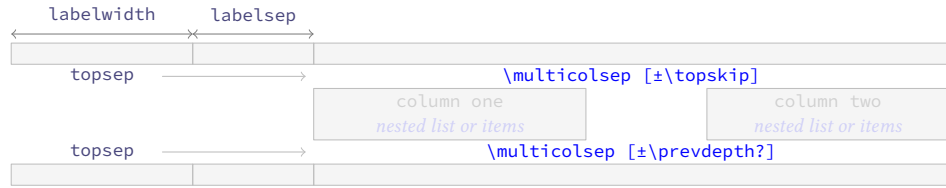


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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*.

```

983 \cs_new_protected:Nn \__enumext_multi_set_vskip:
984 {
985   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip {
986     {
987       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
988         }
989     }
990   \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip {
991     {
992       \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip {
993         }
994   }
995   \__enumext_add_pre_parsep:
996 }

```

(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*.

```

995 \cs_new_protected:Nn \__enumext_add_pre_parsep:
996 {
997   \int_case:nn { \__enumext_level_int }
998   {
999     { 2 } {
1000       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip } {
1001         {
1002           \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
1003         }
1004       }
1005     { 3 } {
1006       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip } {
1007         {
1008           \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
1009         }
1010       }
1011     { 4 } {
1012       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip } {
1013         {
1014           \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
1015         }
1016       }
1017     }
1018   }
1019 }

```

```

1017     }
1018 }

```

(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)*.

```

1019 \cs_new_protected:Nn \__enumext_multi_addvspace:
1020 {
1021   \__enumext_multi_set_vskip:
1022   \mode_if_vertical:T
1023   {
1024     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
1025     {
1026       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1027     }
1028     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
1029     {
1030       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1031     }
1032   }
1033   \par\nopagebreak
1034   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
1035 }

```

(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:`

```

1036 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1037 {
1038   \skip_set:Nn \__enumext_multicols_above_v_skip
1039   {
1040     \__enumext_topsep_v_skip
1041   }
1042   \skip_set:Nn \__enumext_multicols_below_v_skip
1043   {
1044     \__enumext_topsep_v_skip
1045   }
1046 }
1047 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1048 {
1049   \__enumext_keyans_multi_set_vskip:
1050   \mode_if_vertical:T
1051   {
1052     \skip_add:Nn \__enumext_multicols_above_v_skip
1053     {
1054       \skip_use:N \__enumext_partopsep_v_skip
1055     }
1056     \skip_add:Nn \__enumext_multicols_below_v_skip
1057     {
1058       \skip_use:N \__enumext_partopsep_v_skip
1059     }
1060   }
1061   \par\nopagebreak
1062   \addvspace{ \__enumext_multicols_above_v_skip }
1063 }

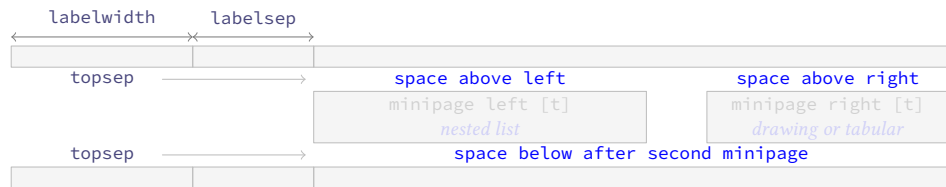
```

(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

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`.

```

1064 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1065 {
1066   \int_compare:nNnTF
1067     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1068     {

```

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.

```

1069   \skip_if_eq:nTF
1070     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1071     {
1072       \skip_set:Nn \l__enumext_minipage_left_skip
1073       {
1074         -0.150\box_dp:N \strutbox
1075       }
1076       \skip_set:Nn \l__enumext_minipage_right_skip
1077       {
1078         0.695\box_dp:N \strutbox
1079       }
1080       \skip_set:Nn \l__enumext_minipage_after_skip
1081       {
1082         \box_dp:N \strutbox
1083       }
1084       \__enumext_zero_parsep:
1085     }
1086   {
1087     \skip_set:Nn \l__enumext_minipage_left_skip
1088     {
1089       \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1090     }
1091     \skip_set:Nn \l__enumext_minipage_right_skip
1092     {
1093       0.695\box_dp:N \strutbox
1094     }
1095     \skip_set:Nn \l__enumext_minipage_after_skip
1096     {
1097       1.85\box_dp:N \strutbox
1098       + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1099     }

```

```

1100     }
1101   }
1102   {

```

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.

```

1103     \skip_if_eq:nnTF
1104     { \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip } } { \c_zero_skip }
1105     {
1106       \skip_set:Nn \l__enumext_minipage_left_skip
1107       {
1108         0.5\box_dp:N \strutbox
1109         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1110       }
1111       \skip_set:Nn \l__enumext_minipage_right_skip
1112       {
1113         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1114       }
1115       \skip_set:Nn \l__enumext_minipage_after_skip
1116       {
1117         1.6\box_dp:N \strutbox
1118       }
1119     }
1120     {
1121       \skip_set:Nn \l__enumext_minipage_left_skip
1122       {
1123         0.5875\box_dp:N \strutbox
1124         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1125       }
1126       \skip_set:Nn \l__enumext_minipage_right_skip
1127       {
1128         + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1129         + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1130       }
1131       \skip_set:Nn \l__enumext_minipage_after_skip
1132       {
1133         0.325\box_dp:N \strutbox
1134         + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1135       }
1136     }
1137   }
1138 }

```

(End of definition for `__enumext_mini_set_vskip:`)

`__enumext_zero_parsep:`

The function `__enumext_zero_parsep:` “adjusted” the value of `\l__enumext_minipage_after_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces* and this is noticeable when using the `nosep` or `noitemsep` keys.

```

1139 \cs_new_protected:Nn \__enumext_zero_parsep:
1140 {
1141   \int_case:nn { \l__enumext_level_int }
1142   {
1143     { 2 } {
1144       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1145       {
1146         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1147       }
1148     }
1149     { 3 } {
1150       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1151       {
1152         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1153       }
1154     }
1155     { 4 } {
1156       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1157       {
1158         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1159       }
1160     }

```

```

1161     }
1162 }

```

(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*.

```

1163 \cs_new_protected:Nn \__enumext_mini_addvspace:
1164 {
1165     \__enumext_mini_set_vskip:
1166     \mode_if_vertical:T
1167     {
1168         \skip_add:Nn \l__enumext_minipage_left_skip
1169         {
1170             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1171         }
1172         \skip_add:Nn \l__enumext_minipage_after_skip
1173         {
1174             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1175         }
1176     }
1177     \par\nopagebreak
1178     \addvspace { \l__enumext_minipage_left_skip }
1179 }

```

(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`.

```

1180 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1181 {
1182     \skip_zero_new:N \l__enumext_minipage_after_skip
1183     \skip_zero_new:N \l__enumext_minipage_left_skip
1184     \skip_zero_new:N \l__enumext_minipage_right_skip
1185     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1186     {
1187         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1188         {
1189             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1190             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1191             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1192             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1193             {
1194                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1195             }
1196         }
1197     }
1198     \skip_set:Nn \l__enumext_minipage_left_skip
1199     {
1200         \skip_use:N \l__enumext_topsep_v_skip
1201     }
1202     \skip_set:Nn \l__enumext_minipage_right_skip
1203     {
1204         0.705\box_dp:N \strutbox
1205     }
1206     \skip_set:Nn \l__enumext_minipage_after_skip
1207     {
1208         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1209     }
1210 }
1211 {
1212     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1213     {
1214         \skip_set:Nn \l__enumext_minipage_left_skip
1215         {

```



```

1217         0.5\box_dp:N \strutbox
1218         + \l__enumext_partopsep_v_skip
1219     }
1220     \skip_set:Nn \l__enumext_minipage_right_skip
1221     {
1222         \l__enumext_partopsep_v_skip
1223     }
1224     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1225 }
1226 {
1227     \skip_set:Nn \l__enumext_minipage_left_skip
1228     {
1229         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1230     }
1231     \skip_set:Nn \l__enumext_minipage_right_skip
1232     {
1233         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1234     }
1235     \skip_set:Nn \l__enumext_minipage_after_skip
1236     {
1237         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1238     }
1239 }
1240 }
1241 }

```

(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`.

```

1242 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1243 {
1244     \l__enumext_keyans_mini_set_vskip:
1245     \mode_if_vertical:T
1246     {
1247         \skip_add:Nn \l__enumext_minipage_left_skip
1248         {
1249             \l__enumext_partopsep_v_skip
1250         }
1251         \skip_add:Nn \l__enumext_minipage_after_skip
1252         {
1253             \l__enumext_partopsep_v_skip
1254         }
1255     }
1256     \par\nopagebreak
1257     \addvspace { \l__enumext_minipage_left_skip }
1258 }

```

(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*`.

```

1259 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1260 {
1261     \skip_zero_new:N \l__enumext_minipage_left_skip
1262     \skip_gzero_new:N \g__enumext_minipage_right_skip
1263     \skip_gzero_new:N \g__enumext_minipage_after_skip
1264     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1265     {
1266         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1267         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1268     }
1269     {
1270         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1271         \skip_gset:Nn \g__enumext_minipage_right_skip

```

```

1272         {
1273             \l__enumext_topsep_vii_skip
1274         }
1275         \skip_gset:Nn \g__enumext_minipage_after_skip
1276         {
1277             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1278         }
1279     }
1280 }
1281 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1282 {
1283     \skip_zero_new:N \l__enumext_minipage_after_skip
1284     \skip_zero_new:N \l__enumext_minipage_left_skip
1285     \skip_zero_new:N \l__enumext_minipage_right_skip
1286     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1287     {
1288         \skip_set:Nn \l__enumext_minipage_left_skip
1289         {
1290             0.5\box_dp:N \strutbox
1291         }
1292         \skip_set:Nn \l__enumext_minipage_right_skip
1293         {
1294             \l__enumext_partopsep_viii_skip
1295         }
1296         \skip_set:Nn \l__enumext_minipage_after_skip
1297         {
1298             1.6\box_dp:N \strutbox
1299         }
1300     }
1301 }
1302 \skip_set:Nn \l__enumext_minipage_left_skip
1303 {
1304     0.5875\box_dp:N \strutbox
1305 }
1306 \skip_set:Nn \l__enumext_minipage_right_skip
1307 {
1308     \l__enumext_topsep_viii_skip
1309 }
1310 \skip_set:Nn \l__enumext_minipage_after_skip
1311 {
1312     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1313 }
1314 }
1315 }

```

(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 $\langle\textit{horizontal mode}\rangle$ or $\langle\textit{vertical mode}\rangle$, since `\partopsep` is equal to `0pt` in both environments.

```

1316 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1317 {
1318     \__enumext_mini_set_vskip_vii:
1319     \par\nopagebreak
1320     \addvspace { \l__enumext_minipage_left_skip }
1321 }
1322 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1323 {
1324     \__enumext_mini_set_vskip_viii:
1325     \par\nopagebreak
1326     \addvspace { \l__enumext_minipage_left_skip }
1327 }

```

(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.

```

1328 \NewDocumentCommand \miniright { s }
1329 {
1330   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1331   {
1332     \msg_error:nnn { enumext } { wrong-miniright-place }
1333   }
1334   \int_compare:nNt { \__enumext_level_int } = { 0 }
1335   {
1336     \msg_error:nnn { enumext } { wrong-miniright-place }
1337   }
1338   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1339   {
1340     \__enumext_keyans_mini_right_cmd:n {#1}
1341   }
1342   { \__enumext_mini_right_cmd:n {#1} }
1343 }

```

(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 `__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.

```

1344 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1345 {
1346   \dim_compare:nNtF
1347   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1348   {
1349     \__enumext_multicols_stop:
1350     \end{__enumext_mini_env*}
1351     \hfill
1352     \begin{__enumext_mini_env*}
1353     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1354     \par\addvspace { \l__enumext_minipage_right_skip }
1355     \bool_if:nF {#1}
1356     {
1357       \centering
1358     }
1359     \int_gzero:N \g__enumext_minipage_stat_int
1360   }
1361   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1362 }

```

(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.

```

1363 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1364 {
1365   \dim_compare:nNtF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1366   {
1367     \__enumext_keyans_multicols_stop:
1368     \end{__enumext_mini_env*}
1369     \hfill
1370     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1371     \par\addvspace { \l__enumext_minipage_right_skip }
1372     \bool_if:nF {#1}
1373     {
1374       \centering
1375     }

```

```

1376         \int_gzero:N \g__enumext_minipage_stat_int
1377     }
1378     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1379 }

```

(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* 1380 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below* 1381 {
below* 1382   \keys_define:nn { enumext / #1 }
1383   {
1384     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1385     above .value_required:n = true,
1386     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1387                       \keys_set:nn { enumext / #1 } { above = {##1} },
1388     above* .value_required:n = true,
1389     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1390     below .value_required:n = true,
1391     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1392                       \keys_set:nn { enumext / #1 } { below = {##1} },
1393     below* .value_required:n = true,
1394   }
1395 }
1396 \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.

```

1397 \cs_new_protected:Nn \__enumext_vspace_above:
1398 {
1399   \skip_if_eq:nnF
1400   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1401   {
1402     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1403     {
1404       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1405     }
1406     {
1407       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1408     }
1409   }
1410 }

```

(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.

```

1411 \cs_new_protected:Nn \__enumext_vspace_below:
1412 {
1413   \skip_if_eq:nnF
1414   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1415   {
1416     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1417     {
1418       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1419     }
1420     {
1421       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1422     }
1423   }
1424 }

```

(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.

```

1425 \cs_new_protected:Nn \__enumext_vspace_above_v:
1426 {
1427   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1428   {
1429     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1430     {
1431       \vspace*{ \l__enumext_vspace_above_v_skip }
1432     }
1433     { \vspace { \l__enumext_vspace_above_v_skip } }
1434   }
1435 }

```

(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.

```

1436 \cs_new_protected:Nn \__enumext_vspace_below_v:
1437 {
1438   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1439   {
1440     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1441     {
1442       \vspace*{ \l__enumext_vspace_below_v_skip }
1443     }
1444     { \vspace { \l__enumext_vspace_below_v_skip } }
1445   }
1446 }

```

(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:`

```

1447 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1448 {
1449   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1450   {
1451     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1452     {
1453       \vspace*{ \l__enumext_vspace_above_vii_skip }
1454     }
1455     { \vspace { \l__enumext_vspace_above_vii_skip } }
1456   }
1457 }
1458 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1459 {
1460   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1461   {
1462     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1463     {
1464       \vspace*{ \l__enumext_vspace_above_viii_skip }
1465     }
1466     { \vspace { \l__enumext_vspace_above_viii_skip } }
1467   }
1468 }

```

(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:`

```

1469 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1470 {
1471   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1472   {
1473     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1474     {

```

```

1475         \vspace*{ \l__enumext_vspace_below_vii_skip }
1476     }
1477     { \vspace { \l__enumext_vspace_below_vii_skip } }
1478 }
1479 }
1480 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1481 {
1482     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1483     {
1484         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1485         {
1486             \vspace*{ \l__enumext_vspace_below_viii_skip }
1487         }
1488         { \vspace { \l__enumext_vspace_below_viii_skip } }
1489     }
1490 }

```

(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](#)

We define the keys `series`, `resume` and `resume*` only for the “first level” of `enumext` and `enumext*`.

```

series
resume
resume*
1491 \cs_set_protected:Npn \__enumext_tmp:n #1
1492 {
1493     \keys_define:nn { enumext / #1 }
1494     {
1495         series .str_set:N = \l__enumext_series_str,
1496         series .value_required:n = true,
1497         resume .code:n = \__enumext_resume_series:n {##1},
1498         resume* .code:n = \__enumext_resume_starred:,
1499         resume* .value_forbidden:n = true,
1500     }
1501 }
1502 \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

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.

```

1503 \cs_new:Npn \__enumext_filter_series:n #1
1504 {
1505     \use:e
1506     {
1507         \keyval_parse:NNn
1508         \__enumext_filter_series_key:n
1509         \__enumext_filter_series_pair:nn {##1}
1510     }
1511 }

```

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.

```

1512 \cs_new:Npn \__enumext_filter_series_key:n #1
1513 {
1514     \str_case:nnF {##1}
1515     {
1516         { resume } {}
1517         { resume* } {}
1518     }
1519     { , { \exp_not:n {##1} } }
1520 }

```

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.

```

1521 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1522 {
1523     \str_case:nnF {##1}

```



```

1524     {
1525         { series } {}
1526         { resume } {}
1527         { start } {}
1528         { save-ans } {}
1529         { save-key } {}
1530     }
1531     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1532 }

```

(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).

```

1533 \cs_new_protected:Npn \__enumext_parse_series:n #1
1534 {
1535     \str_if_empty:NTF \l__enumext_series_str
1536     {
1537         \bool_if:NF \l__enumext_resume_active_bool
1538         {
1539             \__enumext_resume_last:n {#1}
1540         }
1541     }
1542     {
1543         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1544         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1545         { \__enumext_filter_series:n {#1} }
1546         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1547         {
1548             \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1549         }
1550     }
1551 }

```

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.

```

1552 \cs_new_protected:Npn \__enumext_resume_last:n #1
1553 {
1554     \bool_if:NT \l__enumext_standar_first_bool
1555     {
1556         \tl_gclear:N \g__enumext_standar_series_tl
1557         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1558     }
1559     \bool_if:NT \l__enumext_starred_first_bool
1560     {
1561         \tl_gclear:N \g__enumext_starred_series_tl
1562         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1563     }
1564 }

```

(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 $\langle series\ name \rangle$ but are executed at different moments, the integer variable with `__enumext_series_str` sets the value when execute `series=\langle series\ name \rangle` and the integer variable with `__enumext__resume_name_tl` sets the subsequent values when use `resume=\langle series\ name \rangle`. This function is passed to the `enumext` environment definition (§11.33) and the `enumext*` environment definition (§11.36).

```

1565 \cs_new_protected:Nn \__enumext_resume_save_counter:
1566 {
1567   \bool_if:NT \g__enumext_standar_bool
1568   {
1569     \tl_if_empty:NF \__enumext_series_str
1570     {
1571       \int_gset_eq:cN
1572       { g__enumext_series_ \__enumext_series_str_int } \value{enumXi}
1573     }
1574     \tl_if_empty:NTF \__enumext_resume_name_tl
1575     {
1576       \str_if_empty:NT \__enumext_series_str
1577       {
1578         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1579       }
1580     }
1581     {
1582       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1583       {
1584         \int_gset_eq:cN
1585         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXi}
1586       }
1587     }
1588     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1589     {
1590       \int_gset_eq:cN
1591       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXi}
1592     }
1593   }
1594   \bool_if:NT \g__enumext_starred_bool
1595   {
1596     \tl_if_empty:NF \__enumext_series_str
1597     {
1598       \int_gset_eq:cN
1599       { g__enumext_series_ \__enumext_series_str_int } \value{enumXvii}
1600     }
1601     \tl_if_empty:NTF \__enumext_resume_name_tl
1602     {
1603       \str_if_empty:NT \__enumext_series_str
1604       {
1605         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1606       }
1607     }
1608     {
1609       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1610       {
1611         \int_gset_eq:cN
1612         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXvii}
1613       }
1614     }
1615     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1616     {
1617       \int_gset_eq:cN
1618       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXvii}
1619     }
1620   }
1621 }

```

(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=\langle series\ name \rangle` 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.

```

1622 \cs_new_protected:Npn \__enumext_resume_series:n #1
1623 {
1624   \tl_if_empty:NTF {#1}
1625   {
1626     \__enumext_resume_counter:n { }
1627   }
1628   {
1629     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1630     {
1631       \__enumext_resume_counter:n {#1}
1632       \bool_if:NT \g__enumext_standar_bool
1633       {
1634         \keys_set:nv { enumext / level-1 }
1635         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1636       }
1637       \bool_if:NT \g__enumext_starred_bool
1638       {
1639         \keys_set:nv { enumext / enumext* }
1640         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1641       }
1642     }
1643     {
1644       \bool_if:NT \g__enumext_standar_bool
1645       {
1646         \msg_error:nnn { enumext } { unknown-series } {#1}
1647       }
1648       \bool_if:NT \g__enumext_starred_bool
1649       {
1650         \msg_error:nnn { enumext } { unknown-series } {#1}
1651       }
1652     }
1653   }
1654 }

```

(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`.

```

1655 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1656 {
1657   \bool_set_true:N \l__enumext_resume_active_bool
1658   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1659   \tl_if_empty:NTF \l__enumext_resume_name_tl
1660   {
1661     \__enumext_resume_counter:
1662   }
1663   {
1664     \__enumext_resume_counter_series:
1665   }
1666   \__enumext_resume_counter_save_ans:
1667 }

```

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.

```

1668 \cs_new_protected:Nn \__enumext_resume_counter:
1669 {
1670   \bool_if:NT \g__enumext_standar_bool
1671   {
1672     \int_gincr:N \g__enumext_resume_int
1673     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1674   }
1675   \bool_if:NT \g__enumext_starred_bool
1676   {
1677     \int_gincr:N \g__enumext_resume_vii_int

```

```

1678     \int_set_eq:Nn \l__enumext_start_vii_int \g__enumext_resume_vii_int
1679   }
1680 }

```

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.

```

1681 \cs_new_protected:Nn \__enumext_resume_counter_series:
1682 {
1683   \bool_if:NT \g__enumext_standar_bool
1684   {
1685     \int_set:Nn \l__enumext_start_i_int
1686     {
1687       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1688     }
1689   }
1690   \bool_if:NT \g__enumext_starred_bool
1691   {
1692     \int_set:Nn \l__enumext_start_vii_int
1693     {
1694       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1695     }
1696   }
1697 }

```

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.

```

1698 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1699 {
1700   \bool_lazy_and:nnT
1701   { \bool_if_p:N \l__enumext_standar_first_bool }
1702   { \bool_if_p:N \l__enumext_store_active_bool }
1703   {
1704     \int_set:Nn \l__enumext_start_i_int
1705     {
1706       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1707     }
1708   }
1709   \bool_lazy_and:nnT
1710   { \bool_if_p:N \l__enumext_starred_first_bool }
1711   { \bool_if_p:N \l__enumext_store_active_bool }
1712   {
1713     \int_set:Nn \l__enumext_start_vii_int
1714     {
1715       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1716     }
1717   }
1718 }

```

(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.

```

1719 \cs_new_protected:Nn \__enumext_resume_starred:
1720 {
1721   \bool_if:NT \g__enumext_standar_bool
1722   {
1723     \tl_if_empty:NF \g__enumext_standar_series_tl
1724     {
1725       \__enumext_resume_counter:n { }
1726       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1727     }
1728   }
1729   \bool_if:NT \g__enumext_starred_bool
1730   {
1731     \tl_if_empty:NF \g__enumext_starred_series_tl
1732     {

```

```

1733         \__enumext_resume_counter:n { }
1734         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1735     }
1736 }
1737 }

```

(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*`.

```

1738 \cs_set_protected:Npn \__enumext_tmp:n #1
1739 {
1740     \keys_define:nn { enumext / #1 }
1741     {
1742         save-ans .code:n = \__enumext_storing_set:n {##1},
1743         save-ans .value_required:n = true,
1744     }
1745 }
1746 \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:`.

```

1747 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1748 {
1749     \msg_term:nnVV { enumext } { save-ans-log }
1750     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1751 }
1752 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1753 {
1754     \msg_term:nnVV { enumext } { save-ans-log-hook }
1755     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1756 }

```

(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.

```

1757 \cs_new_protected:Npn \__enumext_storing_set:n #1
1758 {
1759     \tl_set:Nx \l__enumext_store_name_tl {#1}
1760     \tl_if_empty:NTF \l__enumext_store_name_tl
1761     {
1762         \bool_lazy_or:nnT
1763         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1764         {
1765             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1766         }
1767     }
1768     {
1769         \bool_lazy_or:nnT
1770         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1771         {
1772             \__enumext_start_save_ans_msg:
1773             \__enumext_storing_exec:
1774         }
1775     }
1776 }

```

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*`.

```

1777 \cs_new_protected:Nn \__enumext_storing_exec:
1778 {
1779   \bool_set_true:N \l__enumext_store_active_bool
1780   \bool_set_true:N \l__enumext_check_answers_bool
1781   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1782   \__enumext_scontents_anskey:V \l__enumext_store_name_tl
1783   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1784   {
1785     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1786     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1787   }
1788   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1789   {
1790     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1791     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1792   }
1793   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1794   {
1795     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1796     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1797   }
1798 }

```

(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 1799 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1800 {
1801   \keys_define:nn { enumext / #1 }

```

```

1802     {
1803         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1804         check-ans .initial:n = false,
1805         check-ans .value_required:n = true,
1806         no-store .code:n = {
1807             \bool_set_false:N \l__enumext_check_answers_bool
1808             \bool_set_false:N \l__enumext_check_ans_key_bool
1809         },
1810         no-store .value_forbidden:n = true,
1811     }
1812 }
1813 \clist_map_inline:nn
1814 {
1815     level-1, level-2, level-3, level-4, enumext*
1816 }
1817 { \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.

```

1818 \cs_new_protected:Nn \l__enumext_check_ans_active:
1819 {
1820     \tl_if_empty:NF \l__enumext_store_name_tl
1821     {
1822         \bool_if:NT \l__enumext_check_answers_bool
1823         {
1824             \l__enumext_check_ans_level:
1825         }
1826     }
1827 }

```

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.

```

1828 \cs_new_protected:Nn \l__enumext_check_ans_level:
1829 {
1830     \int_case:nn { \l__enumext_level_int }
1831     {
1832         { 1 }{
1833             \bool_lazy_all:nT
1834             {
1835                 { \bool_if_p:N \g__enumext_starred_bool }
1836                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1837             }
1838             {
1839                 \int_gdecr:N \g__enumext_item_number_int
1840             }
1841         }
1842         { 2 }{
1843             \int_gdecr:N \g__enumext_item_number_int
1844         }
1845         { 3 }{
1846             \int_gdecr:N \g__enumext_item_number_int
1847         }
1848         { 4 }{
1849             \int_gdecr:N \g__enumext_item_number_int
1850         }
1851     }

```

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.

```

1852     \int_case:nn { \l__enumext_level_h_int }
1853     {
1854         { 1 }{
1855             \bool_lazy_all:nT
1856             {

```



```

1857         { \bool_if_p:N \g__enumext_standar_bool }
1858         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1859     }
1860     {
1861         \int_gdecr:N \g__enumext_item_number_int
1862     }
1863 }
1864 }
1865 }

```

(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.

```

1866 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1867 {
1868     \bool_lazy_and:nnT
1869     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1870     { \bool_if_p:N \g__enumext_standar_bool }
1871     {
1872         \bool_gset_true:N \g__enumext_check_ans_key_bool
1873     }
1874     \bool_lazy_and:nnT
1875     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1876     { \bool_if_p:N \g__enumext_starred_bool }
1877     {
1878         \bool_gset_true:N \g__enumext_check_ans_key_bool
1879     }
1880 }

```

(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:`.

```

1881 \cs_new_protected:Nn \__enumext_item_answer_diff:
1882 {
1883     \int_gset:Nn \g__enumext_item_answer_diff_int
1884     {
1885         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1886     }
1887 }

```

(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:`.

```

1888 \cs_new_protected:Nn \__enumext_check_ans_show:
1889 {
1890     \int_case:nn { \g__enumext_item_answer_diff_int }
1891     {
1892         { -1 } { \__enumext_check_ans_msg_less: }
1893         { 0 } { \__enumext_check_ans_msg_same_ok: }
1894         { 1 } { \__enumext_check_ans_msg_greater: }
1895     }
1896 }
1897 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1898 {
1899     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1900     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1901 }
1902 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1903 {
1904     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1905     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1906 }

```

```

1907 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1908 {
1909     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1910     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1911 }

```

(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:.

```

1912 \cs_new_protected:Nn \__enumext_check_ans_log:
1913 {
1914     \int_case:nn { \g__enumext_item_answer_diff_int }
1915     {
1916         { -1 } { \__enumext_check_ans_log_msg_less: }
1917         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1918         { 1 } { \__enumext_check_ans_log_msg_greater: }
1919     }
1920 }
1921 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1922 {
1923     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1924     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1925 }
1926 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1927 {
1928     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1929     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1930 }
1931 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1932 {
1933     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1934     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1935 }

```

(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.

```

1936 \cs_new_protected:Nn \__enumext_execute_after_env:
1937 {
1938     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1939     {
1940         \tl_if_empty:NF \g__enumext_store_name_tl
1941         {
1942             \__enumext_stop_save_ans_msg:
1943             \__enumext_item_answer_diff:
1944             \__enumext_log_global_vars:
1945             \__enumext_log_answer_vars:
1946             \bool_if:NTF \g__enumext_check_ans_key_bool
1947             {
1948                 \__enumext_check_ans_show:
1949             }
1950             { \__enumext_check_ans_log: }
1951             \cs_undefine:c { __scontents_anskey*_env_begin: }
1952             \cs_undefine:c { __scontents_anskey*_env_end: }
1953         }
1954         \__enumext_reset_global_vars:
1955     }
1956 }

```

(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.

```

1957 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1958 {
1959   \int_compare:nNtT
1960     { \g__enumext_check_starred_cmd_int } = { 0 }
1961     {
1962       \msg_warning:nnnV
1963         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1964     }
1965   \int_compare:nNtT
1966     { \g__enumext_check_starred_cmd_int } > { 1 }
1967     {
1968       \msg_warning:nnnV
1969         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1970     }
1971   \int_gzero:N \g__enumext_check_starred_cmd_int
1972   \tl_clear:N \l__enumext_check_start_line_env_tl
1973 }

```

(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*`.

```

1974 \cs_set_protected:Npn \__enumext_tmp:n #1
1975 {
1976   \keys_define:nn { enumext / #1 }
1977   {
1978     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1979     wrap-ans .initial:n = \fbox{##1},
1980     wrap-ans .value_required:n = true,
1981     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1982     wrap-opt .initial:n = [{##1}],
1983     wrap-opt .value_required:n = true,
1984     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1985     save-sep .initial:n = {, ~ },
1986     save-sep .value_required:n = true,
1987     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1988     mark-ans .initial:n = \textasteriskcentered,
1989     mark-ans .value_required:n = true,
1990     mark-pos .choice:,
1991     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1992     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1993     mark-pos .initial:n = right,
1994     mark-pos .value_required:n = true,
1995     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1996     show-ans .initial:n = false,
1997     show-ans .value_required:n = true,
1998     show-pos .bool_set:N = \l__enumext_show_position_bool,
1999     show-pos .initial:n = false,
2000     show-pos .value_required:n = true,
2001     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2002     mark-ref .initial:n = \textasteriskcentered,
2003     mark-ref .value_required:n = true,
2004     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2005     save-ref .initial:n = false,
2006     save-ref .value_required:n = true,
2007   }
2008 }
2009 \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`.

```

2010 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

2011 {
2012   \keys_define:nn { enumext / #1 }
2013   {
2014     mark-pos .choice:,
2015     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2016     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2017     mark-pos .initial:n = right,
2018     mark-pos .value_required:n = true,
2019     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2020     show-ans .initial:n = false,
2021     show-ans .value_required:n = true,
2022     show-pos .bool_set:N = \l__enumext_show_position_bool,
2023     show-pos .initial:n = false,
2024     show-pos .value_required:n = true,
2025   }
2026 }
2027 \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⟩*.

```

2028 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2029 {
2030   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2031   {
2032     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2033     \tl_set:ce
2034       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2035       { \__enumext_filter_save_key:n {#1} }
2036   }
2037 }
2038 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2039 {
2040   \bool_if:NF \l__enumext_store_save_key_vii_bool
2041   {
2042     \tl_clear:N \l__enumext_store_save_key_vii_tl
2043     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2044   }
2045 }

```

(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.

```

2046 \cs_set_protected:Npn \__enumext_tmp:n #1
2047 {
2048   \keys_define:nn { enumext / enumext* }
2049   {
2050     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2051     save-key .value_required:n = true,
2052   }
2053   \keys_define:nn { enumext / #1 }
2054   {

```

```

2055         save-key .code:n = \__enumext_parse_save_key:n {##1},
2056         save-key .value_required:n = true,
2057     }
2058 }
2059 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

```

\__enumext_parse_save_key:n
\__enumext_parse_save_key_vii:n

```

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

```

2060 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2061 {
2062     \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2063     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2064     \tl_set:ce
2065         { \l__enumext_store_save_key_ \__enumext_level: _tl }
2066         { \__enumext_filter_save_key:n {#1} }
2067 }
2068 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2069 {
2070     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2071     \tl_clear:N \l__enumext_store_save_key_vii_tl
2072     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2073 }

```

(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

```

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

```

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

```

2074 \cs_new:Npn \__enumext_filter_save_key:n #1
2075 {
2076     \use:e
2077     {
2078         \keyval_parse:NNn
2079         \__enumext_filter_save_key_key:n
2080         \__enumext_filter_save_key_pair:nn {#1}
2081     }
2082 }

```

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.

```

2083 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2084 {
2085     \str_case:nnF {#1}
2086     {
2087         { resume } {} { resume* } {} { no-store } {}
2088     }
2089     { , { \exp_not:n {#1} } }
2090 }

```

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.

```

2091 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2092 {
2093     \str_case:nnF {#1}
2094     {
2095         { series } {} { resume } {} { save-ans } {}
2096         { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2097         { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2098         { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2099         { mini-right } {} { mini-right* } {}
2100     }
2101     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2102 }

```

(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.

```
2103 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2104 {
2105   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2106   {
2107     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2108   }
2109   { #1 }
2110 }
2111 \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.

```
2112 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2113 {
2114   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2115 }
2116 \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.

```
2117 \cs_new_protected:Nn \__enumext_store_level_open:
2118 {
2119   \bool_if:NT \l__enumext_check_answers_bool
2120   {
2121     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2122     {
2123       \__enumext_store_addto_seq:n
2124       {
2125         \item \begin{enumext}
2126       }
2127     }
2128     {
2129       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2130       {
2131         \item \begin{enumext} [
2132       ]
2133       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2134       {
2135       ]
2136       }
2137       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2138     }
2139   }
2140 }
2141 \cs_new_protected:Nn \__enumext_store_level_close:
2142 {
2143   \bool_if:NT \l__enumext_check_answers_bool
2144   {
2145     \__enumext_store_addto_seq:n { \end{enumext} }
2146   }
2147 }
```

(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.

```

2148 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2149 {
2150   \bool_if:NT \l__enumext_check_answers_bool
2151   {
2152     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2153     {
2154       \__enumext_store_addto_seq:n
2155       {
2156         \item \mode_leave_vertical:
2157         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2158         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2159       }
2160     }
2161     {
2162       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2163       {
2164         \item \mode_leave_vertical:
2165         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2166         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2167       ]
2168       }
2169       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2170       {
2171       }
2172       \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2173     }
2174   }
2175 }
2176 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2177 {
2178   \bool_if:NT \l__enumext_check_answers_bool
2179   {
2180     \__enumext_store_addto_seq:n { \end{enumext*} }
2181   }
2182 }

```

(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`

```

2183 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2184 {
2185   \mode_leave_vertical:
2186   \skip_horizontal:n { -\dim_use:N #2 }
2187   \makebox[\opt][ r ]
2188   {
2189     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2190     {
2191       \tl_use:N \l__enumext_mark_answer_sym_tl
2192     }
2193   }
2194   \skip_horizontal:n { \dim_use:N #2 }
2195 }
2196 \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*.

```

2197 \keys_define:nn { enumext / anskey }
2198 {
2199     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2200     break-col .default:n = true,
2201     break-col .value_forbidden:n = true,
2202     item-join .int_set:N = \l__enumext_store_item_join_int,
2203     item-join .value_required:n = true,
2204     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2205     item-star .default:n = true,
2206     item-star .value_forbidden:n = true,
2207     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2208     item-sym* .value_required:n = true,
2209     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2210     item-pos* .value_required:n = true,
2211 }

```

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.

```

2212 \NewDocumentCommand \anskey { o +m }
2213 {
2214     \__enumext_anskey_safe_outer:
2215     \group_begin:
2216         \bool_if:NT \l__enumext_check_answers_bool
2217         {
2218             \int_gincr:N \g__enumext_item_anskey_int
2219             \__enumext_anskey_safe_inner:n {#2}
2220             \__enumext_store_anskey_code:nn {#1} {#2}
2221         }
2222     \group_end:
2223 }

```

(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.

```

2224 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2225 {
2226     \bool_if:NF \l__enumext_store_active_bool
2227     {
2228         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2229     }
2230     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2231     {
2232         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2233     }
2234     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2235     {
2236         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2237     }
2238 }

```

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.

```

2239 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2240 {
2241     \tl_if_empty:nT {#1}

```

```

2242     {
2243         \msg_error:nn { enumext } { anskey-empty-arg }
2244     }
2245     \int_incr:N \l__enumext_anskey_level_int
2246     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2247     {
2248         \msg_error:nn { enumext } { anskey-nested }
2249     }
2250 }

```

(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.

```

2251 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2252 {
2253     \__enumext_store_addto_prop:n {#2}
2254     \bool_if:NT \l__enumext_store_ref_key_bool
2255     {
2256         \__enumext_store_internal_ref:
2257     }
2258     \__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`.

```

2259     \tl_if_novalue:nF {#1}
2260     {
2261         \keys_set:nn { enumext / anskey } {#1}
2262     }
2263     \tl_clear:N \l__enumext_store_anskey_arg_tl
2264     \bool_lazy_and:nnT
2265     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2266     { \bool_not_p:n { \l__enumext_starred_bool } }
2267     {
2268         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2269     }
2270     \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`.

```

2271     \bool_lazy_and:nnT
2272     { \bool_not_p:n { \l__enumext_starred_bool } }
2273     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2274     {
2275         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2276         {
2277             ( \exp_not:V \l__enumext_store_item_join_int )
2278         }
2279     }

```

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⟩*.

```

2280     \bool_if:NTF \l__enumext_store_item_star_bool
2281     {
2282         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2283         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2284         {
2285             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2286             {
2287                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2288             }
2289         }
2290         \dim_compare:nT
2291         {
2292             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2293         }
2294         {

```

```

2295         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2296         {
2297             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2298         }
2299     }
2300     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2301 }
2302 {
2303     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2304 }

```

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`.

```

2305     \bool_lazy_and:nnT
2306     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2307     { \bool_if_p:N \l__enumext_hyperref_bool }
2308     {
2309         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2310         {
2311             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2312             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2313         }
2314     }
2315     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2316 }

```

(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`.

```

2317 \cs_new_protected:Nn \__enumext_store_internal_ref:
2318 {
2319     \cs_set_protected:Npn \__enumext_tmp:n ##1
2320     {
2321         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2322         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2323         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2324         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2325     }
2326     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2327     \cs_set:Npn \__enumext_tmp:n ##1
2328     { . \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.

```

2329     \bool_lazy_all:nT
2330     {
2331         { \bool_if_p:N \g__enumext_starred_bool }
2332         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2333     }
2334     {
2335         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2336         { \tl_use:N \l__enumext_label_copy_vii_tl }
2337     }
2338     \bool_lazy_all:nT
2339     {
2340         { \bool_if_p:N \l__enumext_standar_bool }
2341         { \bool_if_p:N \g__enumext_starred_bool }
2342         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2343     }
2344     {
2345         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2346         {
2347             \tl_use:N \l__enumext_label_copy_vii_tl
2348             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2349         }
2350     }

```

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.

```

2351 \bool_lazy_all:nT
2352 {
2353   { \bool_if_p:N \l__enumext_standar_bool }
2354   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2355   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2356   { \bool_not_p:n { \l__enumext_starred_bool } }
2357 }
2358 {
2359   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2360   {
2361     \tl_use:N \l__enumext_label_copy_i_tl
2362     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2363   }
2364 }
2365 \cs_set:Npn \__enumext_tmp:n ##1
2366 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2367 \bool_lazy_all:nT
2368 {
2369   { \bool_if_p:N \l__enumext_standar_bool }
2370   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2371   { \bool_not_p:n { \g__enumext_starred_bool } }
2372   { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2373 }
2374 {
2375   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2376   {
2377     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2378     . \tl_use:N \l__enumext_label_copy_vii_tl
2379   }
2380 }

```

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

```

2381 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2382 {
2383   \l__enumext_store_name_tl \c_colon_str
2384   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2385 }

```

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.

```

2386 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2387 {
2388   \__enumext_newlabel:nn
2389   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2390   { \l__enumext_newlabel_arg_two_tl }
2391 }
2392 \l__enumext_store_write_aux_file_tl
2393 }

```

(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.

```

2394 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2395 {
2396   \par
2397   \bool_if:NT \l__enumext_starred_bool
2398   {
2399     \cs_set:Nn \__enumext_level: { vii }
2400   }
2401   \__enumext_print_keyans_box:cc
2402   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2403   { \l__enumext_labelsep_ \__enumext_level: _dim }
2404   \__enumext_anskey_wrapper:n { #1 }
2405 }

```

(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.

```

2406 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2407 {
2408   \bool_if:NT \l__enumext_show_answer_bool
2409   {
2410     \__enumext_store_anskey_show_wrap:n { #1 }
2411   }
2412   \bool_if:NT \l__enumext_show_position_bool
2413   {
2414     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2415     {
2416       \group_begin:
2417       \exp_not:N \normalfont
2418       \exp_not:N \footnotesize [ \int_eval:n
2419       {
2420         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2421       }
2422       ]
2423       \group_end:
2424     }
2425     \__enumext_store_anskey_show_wrap:n { #1 }
2426   }
2427 }

```

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

11.26 The environment `anskey*`

`__enumext_scontents_anskey:n`

`__enumext_scontents_anskey:V`

`anskey*`

The function `__enumext_scontents_anskey:n` creates a internal `anskey*` 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)

```

2428 \cs_new_protected:Npn \__enumext_scontents_anskey:n #1
2429 {
2430   \keys_define:nn { scontents / scontents }
2431   {
2432     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2433     break-col .default:n = true,
2434     break-col .value_forbidden:n = true,
2435     item-join .int_gset:N = \g__enumext_store_item_join_int,
2436     item-join .value_required:n = true,
2437     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2438     item-star .default:n = true,
2439     item-star .value_forbidden:n = true,
2440     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2441     item-sym* .value_required:n = true,
2442     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2443     item-pos* .value_required:n = true,
2444   }
2445   \newenvsc{anskey*}[store-env=#1,print-env=false]
2446   \__enumext_scontents_anskey_exec:
2447 }
2448 \cs_generate_variant:Nn \__enumext_scontents_anskey:n { V }

```

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

`__enumext_scontents_anskey_exec:`

`__enumext_scontents_anskey_keys:`

`__enumext_scontents_anskey_store:`

`__enumext_scontents_anskey_clean_vars:`

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.

```

2449 \cs_new_protected:Nn \__enumext_scontents_anskey_exec:
2450 {
2451   \__enumext_after_env:nn { anskeyverb }
2452   {

```

```

2453     \tl_clear:N \l__enumext_store_anskey_arg_tl
2454     \tl_clear:N \l__enumext_store_anskey_opt_tl
2455     \seq_gpop_right:cNT
2456     { g__scontents_name_ \g__enumext_store_name_tl _seq } \l__enumext_store_anskey_arg_tl
2457     { \seq_item:ce { g__scontents_name_ \g__enumext_store_name_tl _seq } { -1 } }
2458     % add to prop
2459     \regex_replace_all:nnN { \s{2,}}\u{c__scontents_hidden_space_str} } { \% } \l__enumext_store_
2460     % keys
2461     \__enumext_scontents_anskey_keys:
2462     % store
2463     \__enumext_scontents_anskey_store:
2464     % clean
2465   }
2466 }

```

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`.

```

2467 \cs_new_protected:Nn \__enumext_scontents_anskey_keys:
2468 {
2469   \bool_lazy_and:nnT
2470   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2471   { \bool_not_p:n { \l__enumext_starred_bool } }
2472   {
2473     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2474   }

```

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

```

2475   \bool_lazy_and:nnT
2476   { \bool_not_p:n { \l__enumext_starred_bool } }
2477   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2478   {
2479     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2480     {
2481       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2482     }
2483   }

```

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

```

2484   \bool_if:NT \g__enumext_store_item_star_bool
2485   {
2486     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2487     {
2488       ,item-star,
2489     }
2490     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2491     {
2492       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2493       {
2494         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2495       }
2496     }
2497     \dim_compare:nT
2498     {
2499       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2500     }
2501     {
2502       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2503       {
2504         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2505       }
2506     }
2507   }
2508 }

```

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⟩`.

```

2509 \cs_new_protected:Nn \__enumext_scontents_anskey_store:
2510 {
2511   \group_begin:

```

```

2512 \tl_if_empty:NF \l__enumext_store_anskey_arg_tl
2513 {
2514     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2515     {
2516         \exp_args:Ne
2517         \anskey{ \scontents_rescan_tokens:x { \l__enumext_store_anskey_arg_tl } }
2518     }
2519     {
2520         \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2521         \exp_args:Ne
2522         \anskey{ \scontents_rescan_tokens:x { \l__enumext_store_anskey_arg_tl } }
2523     }
2524 }
2525 \group_end:
2526 }

```

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`.

```

2527 \cs_new_protected:Nn \__enumext_scontents_anskey_clean_vars:
2528 {
2529     \bool_gset_false:N \g__enumext_store_columns_break_bool
2530     \int_gzero:N \g__enumext_store_item_join_int
2531     \bool_gset_false:N \g__enumext_store_item_star_bool
2532     \tl_gclear:N \g__enumext_store_item_symbol_tl
2533     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2534 }

```

(End of definition for `__enumext_scontents_anskey_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`.

```

2535 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2536 {
2537     \tl_clear:N \l__enumext_store_keyans_label_tl
2538     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2539     {
2540         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2541     }
2542     {
2543         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2544     }
2545     \tl_if_novalue:nF { #1 }
2546     {
2547         % Set save-sep
2548         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2549         {
2550             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2551         }
2552         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2553     }
2554     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2555 }

```

(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)`.

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`__enumext_keyans_store_ref_aux_ii:`

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


```

2556 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2557 {
2558   \bool_if:NT \l__enumext_store_ref_key_bool
2559   {
2560     \cs_set_protected:Npn \__enumext_tmp:n #1
2561     {
2562       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2563       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2564       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2565       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2566     }
2567     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2568     \__enumext_keyans_store_ref_aux_i:
2569   }
2570 }

```

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

```

2571 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2572 {
2573   \bool_if:NT \g__enumext_starred_bool
2574   {
2575     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2576   }
2577   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2578   {
2579     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2580     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2581   }
2582   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2583   {
2584     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2585     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2586   }
2587   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2588   {
2589     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2590     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2591   }
2592   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2593   {
2594     \l__enumext_store_name_tl \c_colon_str
2595     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2596   }
2597   \__enumext_keyans_store_ref_aux_ii:
2598 }

```

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.

```

2599 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2600 {
2601   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2602   {
2603     \__enumext_newlabel:nn
2604     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2605     { \l__enumext_newlabel_arg_two_tl }
2606   }
2607   \l__enumext_store_write_aux_file_tl
2608 }

```

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

```

2609 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1

```

```

2610 {
2611   \tl_clear:N \l__enumext_store_keyans_label_tl
2612   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2613   {
2614     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2615   }
2616   {
2617     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2618   }
2619   \tl_if_novalue:nF { #1 }
2620   {
2621     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2622     {
2623       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2624       {
2625         \l__enumext_store_keyans_item_opt_sep_tl
2626       }
2627     }
2628     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2629   }
2630   \__enumext_keyans_addto_seq_link:
2631 }

```

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.

```

2632 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2633 {
2634   \bool_lazy_and:nnT
2635   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2636   { \bool_if_p:N \l__enumext_hyperref_bool }
2637   {
2638     \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2639     {
2640       \hfill \exp_not:N \hyperlink
2641       {
2642         \exp_not:V \l__enumext_newlabel_arg_one_tl
2643       }
2644       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2645     }
2646   }
2647   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2648   \bool_if:NT \l__enumext_check_answers_bool
2649   {
2650     \int_gincr:N \g__enumext_item_anskey_int
2651   }
2652 }

```

(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.

```

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

```

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

```

2653 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2654 {
2655   \tl_if_novalue:nF { #1 }
2656   {
2657     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2658   }
2659   \bool_if:NT \l__enumext_show_answer_bool
2660   {
2661     \__enumext_keyans_show_ans:
2662   }
2663   \bool_if:NT \l__enumext_show_position_bool
2664   {

```

```

2665     \__enumext_keyans_show_pos:
2666   }
2667 }
2668 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2669 {
2670   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2671   {
2672     \bool_lazy_or:nnT
2673       { \bool_if_p:N \l__enumext_show_answer_bool }
2674       { \bool_if_p:N \l__enumext_show_position_bool }
2675       {
2676         \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2677       }
2678   }
2679 }
2680 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2681 {
2682   \tl_put_left:Nn \l__enumext_label_v_tl
2683   {
2684     \__enumext_print_keyans_box:NN
2685     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2686   }
2687 }
2688 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2689 {
2690   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2691   {
2692     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2693     {
2694       \group_begin:
2695       \exp_not:N \normalfont
2696       \exp_not:N \footnotesize [ \int_eval:n
2697         {
2698           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2699         }
2700       ]
2701       \group_end:
2702     }
2703   }
2704   {
2705     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2706     {
2707       \group_begin:
2708       \exp_not:N \normalfont
2709       \exp_not:N \footnotesize [ \int_eval:n
2710         {
2711           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2712         }
2713       ]
2714       \group_end:
2715     }
2716   }
2717   \tl_put_left:Nn \l__enumext_label_v_tl
2718   {
2719     \__enumext_print_keyans_box:NN
2720     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2721   }
2722 }

```

(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 $\langle symbol \rangle$ and its $\langle offset \rangle$.

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

```

2723 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2724 {
2725   \keys_define:nn { enumext / #1 }
2726   {
2727     item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },

```

```

2728         item-sym* .value_required:n = true,
2729         item-sym* .initial:n = {\$ \star$},
2730         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2731         item-pos* .value_required:n = true,
2732     }
2733 }
2734 \clist_map_inline:nn
2735 {
2736     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2737 }
2738 { \__enumext_tmp:nn #1 }

```

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

11.29 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

2739 \cs_new_protected:Nn \__enumext_footnotetext:nn
2740 {
2741     \footnotetext[#1]{#2}
2742 }
2743 \cs_new_protected:Nn \__enumext_renew_footnote:
2744 {
2745     \seq_gclear:N \g__enumext_footnote_arg_seq
2746     \seq_gclear:N \g__enumext_footnote_int_seq
2747     \RenewDocumentCommand \footnote { o +m }
2748     {
2749         \tl_if_novalue:nTF {##1}
2750         {
2751             \stepcounter{footnote}
2752             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2753         }
2754         {
2755             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2756         }
2757         \footnotemark [ \g__enumext_footnote_int ]
2758         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2759         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2760     }
2761 }
2762 \cs_new_protected:Nn \__enumext_print_footnote:
2763 {
2764     \seq_if_empty:NF \g__enumext_footnote_int_seq
2765     {
2766         \seq_map_pairwise_function:NNN
2767         \g__enumext_footnote_int_seq
2768         \g__enumext_footnote_arg_seq
2769         \__enumext_footnotetext:nn
2770     }
2771 }

```

(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).

```

2772 \cs_new_protected:Npn \__enumext_default_item:n #1

```

```

2773 {
2774   \tl_if_novalue:nTF {#1}
2775   {
2776     \bool_if:NT \l__enumext_check_answers_bool
2777     {
2778       \int_gincr:N \g__enumext_item_number_int
2779     }
2780     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2781     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2782   }
2783   {
2784     \bool_set_eq:cc
2785     { l__enumext_wrap_label_ \__enumext_level: _bool }
2786     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2787     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2788   }
2789 }

```

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

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

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

```

2790 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2791 {
2792   \tl_if_novalue:nF {#1}
2793   {
2794     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2795   }
2796   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2797   \tl_if_novalue:nTF {#2}
2798   {
2799     \dim_set_eq:cc
2800     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2801     { l__enumext_labelsep_ \__enumext_level: _dim }
2802   }
2803   {
2804     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2805   }
2806   \bool_if:NT \l__enumext_check_answers_bool
2807   {
2808     \int_gincr:N \g__enumext_item_number_int
2809   }
2810   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2811   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2812 }

```

(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).

```

2813 \cs_new_protected:Nn \__enumext_redefine_item:
2814 {
2815   \RenewDocumentCommand \item { s o o }
2816   {
2817     \bool_if:nTF {##1}
2818     {
2819       \__enumext_starred_item:nn {##2} {##3}
2820     }
2821     { \__enumext_default_item:n {##2} }
2822   }
2823 }

```

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

__enumext_keyans_default_item:n

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

```

2824 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2825 {
2826   \tl_if_novalue:nTF { #1 }
2827   {
2828     \bool_set_true:N \l__enumext_wrap_label_v_bool
2829     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2830   }
2831   {
2832     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2833     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2834   }
2835 }

```

(End of definition for __enumext_keyans_default_item:n.)

__enumext_keyans_starred_item:n

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

```

2836 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2837 {
2838   \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2839   \__enumext_keyans_show_left:n { #1 }
2840   \bool_set_true:N \l__enumext_wrap_label_v_bool
2841   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

2842   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2843   \__enumext_keyans_addto_prop:n { #1 }
2844   \__enumext_keyans_store_ref:
2845   \__enumext_keyans_addto_seq:n { #1 }
2846   \int_gincr:N \g__enumext_check_starred_cmd_int
2847 }

```

(End of definition for __enumext_keyans_starred_item:n.)

\item*
__enumext_keyans_redefine_item:

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

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

```

2848 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2849 {
2850   \RenewDocumentCommand \item { s o }
2851   {
2852     \bool_if:nTF {##1}
2853     {
2854       \peek_remove_spaces:n

```

```

2855         {
2856             \__enumext_keyans_starred_item:n {##2}
2857         }
2858     }
2859     {
2860         \__enumext_keyans_default_item:n {##2}
2861     }
2862 }
2863 }

```

(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.

```

2864 \cs_new_protected:Nn \__enumext_item_starred:
2865 {
2866     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2867     {
2868         \mode_leave_vertical:
2869         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2870         \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
2871         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2872     }
2873 }

```

(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).

```

2874 \cs_new_protected:Nn \__enumext_make_label:
2875 {
2876     \RenewDocumentCommand \makelabel { m }
2877     {
2878         \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2879         \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2880         \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2881         {
2882             \__enumext_item_starred:
2883             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2884         }
2885         { ##1 }
2886         \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2887         \tl_gclear:N \g__enumext_item_symbol_tl
2888     }
2889 }

```

(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).

```

2890 \cs_new_protected:Nn \__enumext_keyans_make_label:
2891 {
2892     \RenewDocumentCommand \makelabel { m }
2893     {
2894         \tl_use:N \l__enumext_label_fill_left_v_tl
2895         \tl_use:N \l__enumext_label_font_style_v_tl
2896         \bool_if:NTF \l__enumext_wrap_label_v_bool
2897         {
2898             \__enumext_wrapper_label_v:n { ##1 }
2899         }
2900         { ##1 }
2901         \tl_use:N \l__enumext_label_fill_right_v_tl

```



```

2902     }
2903 }

```

(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.

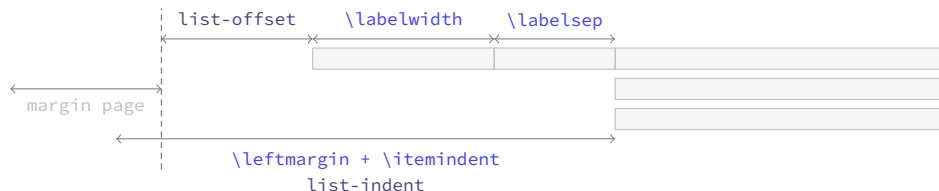


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

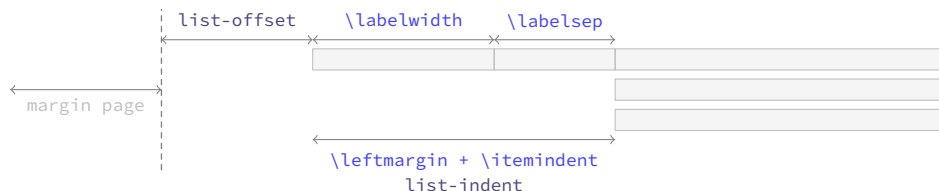


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

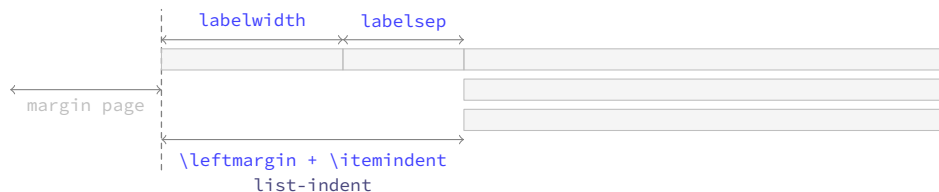


Figure 11: Default horizontal lengths in `enumext`.

```

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

```

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

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

```

2904 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2905 {
2906   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2907   {
2908     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2909     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2910   }
2911   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2912   {
2913     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2914     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2915   }

```

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

```
2916 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2917 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2918 {
2919   \dim_set:Nn #6 { #1 + #2 - #4 }
2920   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2921 }
2922 {
2923   \dim_compare:nNnT { #4 } = { #1 + #2 }
2924   { \dim_set:Nn #6 { \c_zero_dim } }
2925   \dim_compare:nNnT { #4 } < { #1 + #2 }
2926   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2927   \dim_compare:nNnT { #4 } > { #1 + #2 }
2928   {
2929     \dim_set:Nn #6 { -#1 - #2 + #4 }
2930     \dim_set:Nn #6 { #6*-1 }
2931   }
2932   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2933 }
2934 }
2935 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }
```

(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.

```
2936 \cs_set_protected:Npn \__enumext_tmp:n #1
2937 {
2938   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
2939   {
2940     \__enumext_calc_hspace:ccccc
2941     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2942     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2943     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2944     { \__enumext_leftmargin_tmp_#1_bool }
2945     \clist_map_inline:nn
2946     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2947     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2948     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2949     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2950     \usecounter { enumX#1 }
2951     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2952     \str_if_eq:nnTF {#1} { v }
2953     {
2954       \__enumext_keyans_redefine_item:
2955       \__enumext_keyans_make_label:
2956       \__enumext_keyans_ref:
2957       \__enumext_keyans_fake_item:
2958       \bool_if:cT { \__enumext_show_length_#1_bool }
2959       {
2960         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2961       }
2962     }
2963     {
2964       \__enumext_redefine_item:
2965       \__enumext_make_label:
2966       \__enumext_standar_ref:
2967       \__enumext_fake_item:
2968       \bool_if:cT { \__enumext_show_length_#1_bool }
2969       {
2970         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2971       }
2972     }
2973   }
2974 }
2975 \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: For the horizontal environments enumext* and keyans* the implementation is similar, but, the value of
\__enumext_list_arg_two_viii: \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.

2976 \cs_set_protected:Npn \__enumext_tmp:n #1
2977 {
2978   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
2979   {
2980     \__enumext_calc_hspace:ccccc
2981     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2982     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2983     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2984     { \__enumext_leftmargin_tmp_#1_bool }
2985     \clist_map_inline:nn
2986     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2987     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
2988     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2989     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
2990     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2991     \skip_zero:N \partopsep
2992     \usecounter { enumX#1 }
2993     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2994     \__enumext_starred_ref:
2995     \str_if_eq:nnTF {#1} { vii }
2996     {
2997       \__enumext_fake_item_vii:
2998       \bool_if:cT { \__enumext_show_length_vii_bool }
2999       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3000     }
3001     {
3002       \__enumext_fake_item_viii:
3003       \bool_if:cT { \__enumext_show_length_#1_bool }
3004       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3005     }
3006   }
3007 }
3008 \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.

```

3009 \NewDocumentEnvironment{enumext}{0}{ }
3010 {
3011   \__enumext_safe_exec:
3012   \__enumext_parse_keys:n {#1}
3013   \__enumext_before_list:
3014   \__enumext_start_store_level:
3015   \__enumext_start_list:nn
3016   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3017   {
3018     \use:c { \__enumext_list_arg_two_ \__enumext_level: : }
3019     \__enumext_before_keys_exec:
3020   }
3021   \__enumext_after_args_exec:
3022 }
3023 {
3024   \__enumext_stop_list:
3025   \__enumext_stop_store_level:
3026   \__enumext_after_list:
3027 }

```

(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:`.

```

3028 \cs_new_protected:Nn \__enumext_safe_exec:
3029 {
3030   \__enumext_internal_mini_page:
3031   \__enumext_is_not_nested:
3032   \int_incr:N \__enumext_level_int
3033   \int_compare:nNnT { \__enumext_level_int } > { 4 }
3034     { \msg_fatal:nn { enumext } { list-too-deep } }
3035   \bool_set_true:N \__enumext_standar_bool
3036   \__enumext_is_on_first_level:
3037 }

```

(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 `__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 `__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`.

```

3038 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3039 {
3040   \tl_if_novalue:nF {#1}
3041   {
3042     \str_clear:N \__enumext_series_str
3043     \int_compare:nNnTF { \__enumext_level_int } = { 1 }
3044     {
3045       \keys_set:nn { enumext / level-1 } {#1}
3046       \__enumext_parse_series:n {#1}
3047     }
3048     {
3049       \exp_args:Ne \keys_set:nn
3050         { enumext / level-\int_use:N \__enumext_level_int } {#1}
3051     }
3052     \__enumext_store_active_keys:n {#1}
3053   }
3054 }

```

(End of definition for `__enumext_parse_keys:n`)

`__enumext_start_store_level:`

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

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

```

3055 \cs_new_protected:Nn \__enumext_start_store_level:
3056 {
3057   \bool_lazy_all:nT
3058   {
3059     { \bool_if_p:N \__enumext_store_active_bool }
3060     { \bool_not_p:n { \__enumext_keyans_env_bool } }
3061     { \bool_not_p:n { \g__enumext_starred_bool } }
3062   }
3063   {
3064     \int_compare:nNnT { \__enumext_level_int } > { 1 }
3065     {
3066       \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
3067       \__enumext_store_level_open:
3068     }
3069   }
3070   \bool_lazy_all:nT
3071   {
3072     { \bool_if_p:N \__enumext_store_active_bool }
3073     { \bool_not_p:n { \__enumext_keyans_env_bool } }
3074     { \bool_if_p:N \g__enumext_starred_bool }
3075   }
3076   {
3077     \int_compare:nNnT { \__enumext_level_int } > { 0 }
3078     {
3079       \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }

```

```

3080         \__enumext_store_level_open:
3081     }
3082 }
3083 }
3084 \cs_new_protected:Nn \__enumext_stop_store_level:
3085 {
3086     \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3087     {
3088         \__enumext_store_level_close:
3089     }
3090 }

```

(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.

```

3091 \cs_new_protected:Nn \__enumext_before_list:
3092 {
3093     \__enumext_vspace_above:
3094     \__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.

```

3095     \__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.

```

3096     \dim_compare:nNnT
3097     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3098     {
3099         \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3100         {
3101             \linewidth
3102             - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3103             - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3104         }

```

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

```

3105     \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3106     \int_gincr:N \g__enumext_minipage_stat_int
3107     \__enumext_mini_addvspace:
3108     \nointerlineskip\noindent
3109     \begin{\__enumext_mini_env*}
3110     { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3111 }
3112 \__enumext_multicols_start:
3113 }

```

(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.

```

3114 \cs_new_protected:Nn \__enumext_multicols_start:
3115 {
3116     \int_compare:nNnT
3117     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3118     {
3119         \dim_compare:nNnT
3120         { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }

```

```

3121         {
3122             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3123             {
3124                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3125                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3126                   ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3127                   - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3128             }
3129         }
3130         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3131         \skip_zero:N \multicolsep
3132         \int_compare:nNt { \l__enumext_level_int } > { 1 }
3133         {
3134             \dim_zero:N \columnseprule
3135         }

```

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.

```

3136         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3137         {
3138             \__enumext_multi_addvspace:
3139         }
3140         \raggedcolumns
3141         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3142     }
3143 }

```

(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.

```

3144 \cs_new_protected:Nn \__enumext_multicols_stop:
3145 {
3146     \int_compare:nNt
3147     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3148     {
3149         \end{multicols}
3150         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3151         {
3152             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3153         }
3154     }
3155 }

```

(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.

```

3156 \cs_new_protected:Nn \__enumext_after_list:
3157 {
3158     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3159     {
3160         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3161         {
3162             \msg_warning:nn { enumext } { missing-miniright }
3163             \miniright
3164         }
3165         \int_gzero:N \g__enumext_minipage_stat_int
3166         \end{__enumext_mini_env*}
3167         \par\addvspace { \l__enumext_minipage_after_skip }
3168     }
3169     { \__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`.

```

3170     \__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.

```

3171   \__enumext_after_stop_list:
3172   \__enumext_vspace_below:
3173   \bool_set_false:N \l__enumext_standar_bool
3174   \__enumext_resume_save_counter:
3175 }

```

(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`.

```

3176 \__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.

```

3177 \NewDocumentEnvironment{keyans}{0}{ }
3178 {
3179   \__enumext_keyans_safe_exec:
3180   \__enumext_keyans_parse_keys:n {#1}
3181   \__enumext_before_list_v:
3182   \__enumext_start_list:nn
3183   { \tl_use:N \l__enumext_label_v_tl }
3184   {
3185     \__enumext_list_arg_two_v:
3186     \__enumext_before_keys_exec_v:
3187   }
3188   \__enumext_after_args_exec_v:
3189 }
3190 {
3191   \__enumext_check_starred_cmd:n { item }
3192   \__enumext_stop_list:
3193   \__enumext_after_list_v:
3194 }

```

(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.

```

3195 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3196 {
3197   \bool_if:NF \l__enumext_store_active_bool
3198   {
3199     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3200   }
3201   \int_incr:N \l__enumext_keyans_level_int
3202   \bool_set_true:N \l__enumext_keyans_env_bool
3203   \__enumext_keyans_save_start_line:
3204   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3205   \bool_set_false:N \l__enumext_store_active_bool
3206   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3207   {
3208     \msg_error:nn { enumext } { keyans-nested }
3209   }
3210   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3211   {
3212     \msg_error:nn { enumext } { keyans-wrong-level }
3213   }
3214 }

```

(End of definition for `__enumext_keyans_safe_exec:`.)

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

```

3215 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3216 {
3217   \keys_set:nn { enumext / keyans } {#1}
3218 }

```

(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.

```

3219 \cs_new_protected:Nn \__enumext_before_list_v:
3220 {
3221   \__enumext_vspace_above_v:
3222   \__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.

```

3223   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3224   {
3225     \dim_set:Nn \l__enumext_minipage_left_v_dim
3226     {
3227       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3228     }

```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_keyans_mini_addvspace:` is called and the `__enumext_mini_env*` environment on *left side* will be initialized followed by the *vertical spacing* `\l__enumext_minipage_left_skip`. Here we use the plain TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3229     \bool_set_true:N \l__enumext_minipage_active_v_bool
3230     \int_gincr:N \g__enumext_minipage_stat_int
3231     \__enumext_keyans_mini_addvspace:
3232     \nointerlineskip\noindent
3233     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3234   }

```

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

```

3235   \__enumext_keyans_multicols_start:
3236 }

```

(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.

```

3237 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3238 {
3239   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3240   {

```

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

```

3241     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3242     {
3243       \dim_set:Nn \l__enumext_columns_sep_v_dim
3244       {
3245         (
3246           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3247         ) / \l__enumext_columns_v_int
3248         - \l__enumext_listoffset_v_dim
3249       }
3250     }
3251     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim

```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```

3252     \skip_zero:N \multicolsep
3253     \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.

```

3254     \bool_if:NF \l__enumext_minipage_active_v_bool
3255     {
3256         \__enumext_keyans_multi_addvspace:
3257     }
3258     \raggedcolumns
3259     \begin{multicols}{\l__enumext_columns_v_int }
3260 }
3261 }
```

(End of definition for `__enumext_keyans_multicols_start:`)

`__enumext_keyans_multicols_stop:`

The function `__enumext_keyans_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

3262 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3263 {
3264     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3265     {
3266         \end{multicols}
3267         \bool_if:NF \l__enumext_minipage_active_v_bool
3268         {
3269             \par\addvspace{ \l__enumext_multicols_below_v_skip }
3270         }
3271     }
3272 }
```

(End of definition for `__enumext_keyans_multicols_stop:`)

`__enumext_after_list_v:`

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

```

3273 \cs_new_protected:Nn \__enumext_after_list_v:
3274 {
3275     \bool_if:NTF \l__enumext_minipage_active_v_bool
3276     {
3277         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3278         {
3279             \msg_warning:nn { enumext } { missing-miniright }
3280             \miniright
3281         }
3282         \int_gzero:N \g__enumext_minipage_stat_int
3283         \end{__enumext_mini-env*}
3284         \par\addvspace{ \l__enumext_minipage_after_skip }
3285     }
3286     { \__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.

```

3287     \bool_set_false:N \l__enumext_keyans_env_bool
3288     \__enumext_after_stop_list_v:
3289     \__enumext_vspace_below_v:
3290 }
```

(End of definition for `__enumext_after_list_v:`)

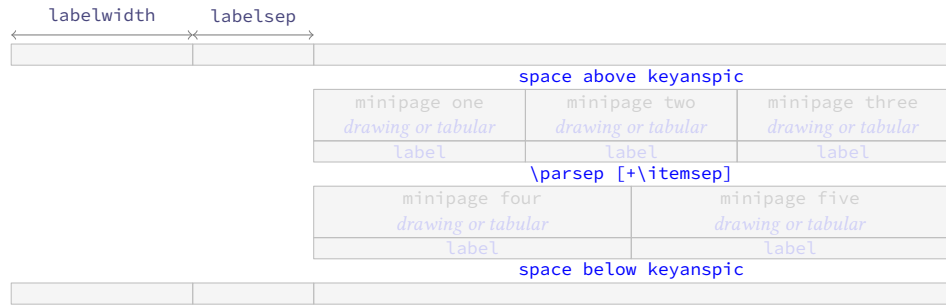
11.35 The environment `keyanspic` and `\anspic`

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.

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

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

11.35.1 The command `\anspic`

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

```
3291 \NewDocumentCommand \anspic { s o +m }
3292 {
```

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

```
3293   \bool_if:NF \l__enumext_store_active_bool
3294   {
3295     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3296   }
3297   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3298   {
3299     \msg_error:nn { enumext } { keyanspic-wrong-level }
3300   }
3301   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3302   {
3303     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3304   }
```

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.

```
3305   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3306   {
3307     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3308   }
3309 }
```

(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.

```
3310 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3311 {
3312   \stepcounter { enumXvi }
3313   #3 \l
3314   \bool_if:nT { #1 }
3315   {
3316     \__enumext_keyans_addto_prop:n { #2 }
3317     \__enumext_keyans_store_ref:
3318     \__enumext_keyans_addto_seq:n { #2 }
3319     \int_gincr:N \g__enumext_check_starred_cmd_int
3320     \bool_lazy_or:nnT
3321     { \bool_if_p:N \l__enumext_show_answer_bool }
3322     { \bool_if_p:N \l__enumext_show_position_bool }
3323     {
3324       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3325       \__enumext_keyans_show_left:n { #2 }
3326       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3327     }
3328   }
3329   \tl_use:N \l__enumext_label_font_style_v_tl
3330   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3331 }
```

(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.

```

3332 \NewDocumentEnvironment{keyanspic}{ o }
3333 {
3334   \__enumext_keyans_pic_safe_exec:
3335   \__enumext_start_list:nn
3336   { }
3337   {
3338     \__enumext_keyans_pic_arg_two:
3339   }

```

We apply the “adjusted” vertical spacing above the environment

```

3340   \vspace { \__enumext_keyans_pic_above_skip }
3341 }

```

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.

```

3342 {
3343   \tl_if_novalue:nTF { #1 }
3344   {
3345     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3346   }
3347   { \__enumext_keyans_pic_do:n { #1 } }
3348   \__enumext_stop_list:
3349   \__enumext_check_starred_cmd:n { anspic }
3350   \setcounter { enumXvi } { 0 }
3351   \vspace { \__enumext_topsep_v_skip }
3352   %\bool_set_false:N \l__enumext_store_active_bool
3353 }

```

(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.

```

3354 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3355 {
3356   \int_incr:N \l__enumext_keyans_pic_level_int
3357   \int_compare:nNt { \l__enumext_keyans_pic_level_int } > { 1 }
3358   {
3359     \msg_error:nn { enumext } { keyanspic-nested }
3360   }
3361   \__enumext_keyans_save_start_line:
3362 }

```

(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`.

```

3363 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3364 {
3365   \dim_compare:nNt { #1 } < { 0pt }
3366   { \skip_set:Nn #1 { -#1 } }
3367 }

```

(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.

```

3368 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3369 {

```

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.

```

3370   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3371   \__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.

```

3372 \skip_add:Nn \parsep { \itemsep }
3373 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3374 \dim_zero:N \labelwidth
3375 \dim_zero:N \listparindent
3376 \dim_zero:N \labelsep
3377 \skip_zero:N \partopsep
3378 \skip_zero:N \itemsep

```

We set the value of `\l__enumxt_keyans_pic_above_skip` which we will use to apply our “adjust” space above `keyanspic`, finally we call `__enumxt_item_std:w` followed by `\scan_stop:` to prevent the error message returned by \TeX when not using the `\item` command.

```

3379 \__enumxt_keyans_pic_skip_abs:N \parsep
3380 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
3381 {
3382   \box_dp:N \strutbox
3383   + \l__enumxt_topsep_v_skip
3384   - \parsep
3385 }
3386 \__enumxt_item_std:w \scan_stop:
3387 }

```

(End of definition for `__enumxt_keyans_pic_arg_two:.`)

```

\__enumxt_keyans_pic_do:n
\__enumxt_keyans_pic_do:e

```

The optional argument is split by comma and is handled directly by the function `__enumxt_keyans_pic_do:n` and passed to the function `__enumxt_keyans_pic_row:n`.

```

3388 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
3389 {
3390   \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
3391 }
3392 \cs_generate_variant:Nn \__enumxt_keyans_pic_do:n { e }

```

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

```
\__enumxt_keyans_pic_row:n
```

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

```

3393 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
3394 {
3395   \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
3396   \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
3397   \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
3398   \int_step_inline:nnn
3399     { \l__enumxt_keyans_pic_above_int + 1 }
3400     { \l__enumxt_keyans_pic_below_int }
3401     {
3402       \__enumxt_minipage:w [ b ]{ \l__enumxt_keyans_pic_width_dim }
3403       \centering
3404       \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
3405       \__enumxt_endminipage:
3406     }
3407   \par
3408 }

```

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

11.36 The environment `enumxt*`

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.

```

3409 \cs_new_protected:Nn \_enumext_starred_columns_set_vii:
3410 {
3411   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3412   {
3413     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3414     {
3415       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3416       / \l__enumext_columns_vii_int
3417     }
3418   }
3419   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3420   \dim_set:Nn \l__enumext_item_width_vii_dim
3421   {
3422     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3423     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3424     - \l__enumext_labelsep_vii_dim
3425   }
3426   \dim_zero_new:N \itemwidth
3427 }

```

(End of definition for _enumext_starred_columns_set_vii:.)

_enumext_starred_joined_item_vii:n

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

```

3428 \cs_new_protected:Npn \_enumext_starred_joined_item_vii:n #1
3429 {
3430   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3431   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3432   {
3433     \msg_warning:nnee { enumext } { item-joined }
3434     { \int_use:N \l__enumext_joined_item_vii_int }
3435     { \int_use:N \l__enumext_columns_vii_int }
3436     \int_set:Nn \l__enumext_joined_item_vii_int
3437     {
3438       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3439     }
3440   }
3441   \int_compare:nNnT
3442   { \l__enumext_joined_item_vii_int }
3443   >
3444   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3445   {
3446     \msg_warning:nnee { enumext } { item-joined-columns }
3447     { \int_use:N \l__enumext_joined_item_vii_int }
3448     {
3449       \int_eval:n
3450       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3451     }
3452     \int_set:Nn \l__enumext_joined_item_vii_int
3453     {
3454       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3455     }
3456   }

```

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

```

3457   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3458   {
3459     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3460     \int_decr:N \l__enumext_joined_item_aux_vii_int
3461     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3462     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3463     \dim_set:Nn \l__enumext_joined_width_vii_dim
3464     {
3465       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3466       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3467         + \l__enumext_columns_sep_vii_dim
3468         ) * \l__enumext_joined_item_aux_vii_int

```

```

3469     }
3470     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3471   }
3472   {
3473     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3474     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3475   }
3476 }

```

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

`__enumext_start_mini_vii:` The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` environment on the “*right side*” is executed “*after*” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

3477 \cs_new_protected:Nn \__enumext_start_mini_vii:
3478 {
3479   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3480   {
3481     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3482     {
3483       \linewidth
3484       - \l__enumext_minipage_right_vii_dim
3485       - \l__enumext_minipage_hsep_vii_dim
3486     }
3487     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3488     \dim_gset_eq:NN
3489       \g__enumext_minipage_right_vii_dim
3490       \l__enumext_minipage_right_vii_dim
3491     \__enumext_mini_addvspace_vii:
3492     \nointerlineskip\noindent
3493     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3494   }
3495 }

```

(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*”.

```

3496 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3497 {
3498   \bool_if:NT \l__enumext_minipage_active_vii_bool
3499   {
3500     \end{__enumext_mini_env*}
3501     \hfill
3502     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3503   }
3504 }

```

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*”.

```

3505 \__enumext_after_env:n {enumext*}
3506 {
3507   \bool_if:NT \g__enumext_minipage_active_vii_bool
3508   {
3509     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3510     \par\addvspace { \g__enumext_minipage_right_skip }
3511     \bool_if:NF \g__enumext_minipage_center_vii_bool
3512     {
3513       \centering
3514     }
3515     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3516     \end{__enumext_mini_env*}
3517     \par\addvspace{ \g__enumext_minipage_after_skip }
3518   }
3519   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3520   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3521   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3522   \dim_gzero:N \g__enumext_minipage_right_vii_dim

```



```

3523     \bool_gset_false:N \g__enumext_starred_bool
3524 }

```

(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.

```

3525 \NewDocumentEnvironment{enumext*}{ o }
3526 {
3527     \__enumext_safe_exec_vii:
3528     \__enumext_parse_keys_vii:n {#1}
3529     \__enumext_before_list_vii:
3530     \__enumext_start_store_level_vii:
3531     \__enumext_start_list:nn { }
3532     {
3533         \__enumext_list_arg_two_vii:
3534         \__enumext_before_keys_exec_vii:
3535     }
3536     \__enumext_starred_columns_set_vii:
3537     \item[] \scan_stop:
3538     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3539     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3540 }
3541 {
3542     \__enumext_stop_item_tmp_vii:
3543     \__enumext_remove_extra_parsep_vii:
3544     \__enumext_stop_list:
3545     \__enumext_stop_store_level_vii:
3546     \__enumext_after_list_vii:
3547 }

```

(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.

```

3548 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3549 {
3550     \__enumext_internal_mini_page:
3551     \__enumext_is_not_nested:
3552     \int_incr:N \l__enumext_level_h_int
3553     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3554     {
3555         \msg_error:nn { enumext } { nested }
3556     }
3557     \bool_set_true:N \l__enumext_starred_bool
3558     \__enumext_is_on_first_level:
3559 }

```

(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*.

```

3560 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3561 {
3562     \tl_if_novalue:nF {#1}
3563     {
3564         \str_clear:N \l__enumext_series_str
3565         \keys_set:nn { enumext / enumext* } {#1}
3566         \__enumext_parse_series:n {#1}
3567         \__enumext_store_active_keys_vii:n {#1}
3568     }
3569 }

```

(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`.

```

3570 \cs_new_protected:Nn \__enumext_before_list_vii:
3571 {
3572     \__enumext_vspace_above_vii:
3573     \__enumext_check_ans_active:
3574     \__enumext_before_args_exec_vii:
3575     \__enumext_start_mini_vii:
3576 }

```

(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.

```

3577 \cs_new_protected:Nn \__enumext_after_list_vii:
3578 {
3579     \__enumext_stop_mini_vii:
3580     \__enumext_after_stop_list_vii:
3581     \__enumext_check_ans_key_hook:
3582     \__enumext_vspace_below_vii:
3583     \bool_set_false:N \l__enumext_starred_bool
3584     \__enumext_resume_save_counter:
3585 }

```

(End of definition for `__enumext_after_list_vii:`.)

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

```

3586 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3587 {
3588     \bool_if:NT \l__enumext_store_active_bool
3589     {
3590         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3591         {
3592             \__enumext_store_level_open_vii:
3593         }
3594     }
3595 }
3596 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3597 {
3598     \bool_if:NT \l__enumext_store_active_bool
3599     {
3600         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
3601         {
3602             \__enumext_store_level_close_vii:
3603         }
3604     }
3605 }

```

(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`.

```

3606 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3607 {
3608     \__enumext_stop_item_tmp_vii:
3609     \int_incr:N \l__enumext_item_column_pos_vii_int
3610     \int_gincr:N \g__enumext_item_count_all_vii_int
3611     \__enumext_item_peek_args_vii:
3612 }

```

(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).

```
3613 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3614 {
3615   \peek_meaning:NTF (
3616     { \__enumext_joined_item_vii:w }
3617     { \__enumext_joined_item_vii:w (1) }
3618   }
```

(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`.

```
3619 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3620 {
3621   \__enumext_starred_joined_item_vii:n {#1}
3622   \peek_meaning_remove:NTF *
3623     { \__enumext_starred_item_vii:w }
3624     { \__enumext_standar_item_vii:w }
3625 }
```

(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]`.

```
3626 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3627 {
3628   \bool_set_false:N \l__enumext_item_starred_vii_bool
3629   \peek_meaning:NTF [
3630     {
3631       \bool_set_eq:NN
3632         \l__enumext_wrap_label_vii_bool
3633         \l__enumext_wrap_label_opt_vii_bool
3634       \__enumext_start_item_vii:w
3635     }
3636     {
3637       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3638       \legacy_if_set_true:n { @noitemarg }
3639       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3640     }
3641   }
```

(End of definition for `__enumext_standar_item_vii:w`)

`__enumext_starred_item_vii:w`

The function `__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[<symbol>]` and `\item*[<symbol>][<offset>]`.

`__enumext_starred_item_vii_aux_i:w`

`__enumext_starred_item_vii_aux_ii:w`

`__enumext_starred_item_vii_aux_iii:w`

```
3642 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3643 {
3644   \bool_set_true:N \l__enumext_item_starred_vii_bool
3645   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3646   \peek_meaning:NTF [
3647     { \__enumext_starred_item_vii_aux_i:w }
3648     { \__enumext_starred_item_vii_aux_ii:w }
3649   }
3650   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3651   {
3652     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3653     \__enumext_starred_item_vii_aux_ii:w
```

```

3654 }
3655 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3656 {
3657   \peek_meaning:NTF [
3658     { \__enumext_starred_item_vii_aux_iii:w }
3659     {
3660       \dim_set_eq:NN
3661         \l__enumext_item_symbol_sep_vii_dim
3662         \l__enumext_labelsep_vii_dim
3663       \legacy_if_set_true:n { @noitemarg }
3664       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3665     }
3666   }
3667 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3668 {
3669   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3670   \legacy_if_set_true:n { @noitemarg }
3671   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3672 }

```

(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.

```

3673 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3674 {
3675   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3676   \legacy_if:nT { @noitemarg }
3677   {
3678     \legacy_if_set_false:n { @noitemarg }
3679     \legacy_if:nT { @nmbrlist }
3680     {
3681       \bool_if:NT \l__enumext_hyperref_bool
3682       {
3683         \legacy_if_set_true:n { @hyper@item }
3684       }
3685       \refstepcounter{enumXvii}
3686       \bool_if:NT \l__enumext_check_answers_bool
3687       {
3688         \int_gincr:N \g__enumext_item_number_int
3689       }
3690     }
3691   }

```

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

```

3692 \group_begin:
3693   \lrbox{ \l__enumext_item_text_vii_box }
3694   \bool_if:NF \l__enumext_footnotes_key_bool
3695   {
3696     \__enumext_renew_footnote:
3697   }
3698   \bool_if:NT \l__enumext_item_starred_vii_bool
3699   {
3700     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3701     {
3702       \tl_gset_eq:NN
3703         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3704     }

```

```

3705         \mode_leave_vertical:
3706         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3707         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3708         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3709         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3710     }
3711 \group_begin:
3712     \tl_use:N \l__enumext_label_font_style_vii_tl
3713     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3714     {
3715         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3716         { \l__enumext_wrapper_label_vii:n {#1} }
3717     }
3718     {
3719         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3720     }
3721 \group_end:
3722 \skip_horizontal:N \l__enumext_labelsep_vii_dim
3723 \tl_use:N \l__enumext_after_list_args_vii_tl
3724 \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3725     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3726     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3727     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3728 }

```

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

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

```

3729 \cs_new_protected_nopar:Nn \l__enumext_stop_item_vii:
3730 {
3731     \l__enumext_endminipage:
3732     \endlrbox
3733     \group_end:
3734     \box_set_wd:Nn \l__enumext_item_text_vii_box
3735     {
3736         \l__enumext_joined_width_vii_dim
3737         + \l__enumext_labelwidth_vii_dim
3738         + \l__enumext_labelsep_vii_dim
3739     }
3740     \int_set:Nn \hbadness { 10000 }
3741     \box_use:N \l__enumext_item_text_vii_box
3742     \bool_if:NF \l__enumext_footnotes_key_bool
3743     {
3744         \l__enumext_print_footnote:
3745     }
3746     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3747     {
3748         \par\noindent
3749         \int_zero:N \l__enumext_item_column_pos_vii_int
3750     }
3751     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3752 }

```

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

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

```

3753 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
3754 {
3755     \int_compare:nNnT
3756     {
3757         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3758     }
3759     =
3760     { \c_zero_int }
3761     {
3762         \par
3763         \vspace{ -\l__enumext_itemsep_vii_skip }
3764         \int_zero:N \g__enumext_item_count_all_vii_int

```

```

3765     }
3766 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`)

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`.

```

3767 \__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.

```

3768 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3769 {
3770   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3771   {
3772     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3773     {
3774       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3775       / \l__enumext_columns_viii_int
3776     }
3777   }
3778   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3779   \dim_set:Nn \l__enumext_item_width_viii_dim
3780   {
3781     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3782     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3783     - \l__enumext_labelsep_viii_dim
3784   }
3785   \dim_zero_new:N \itemwidth
3786 }

```

(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`.

```

3787 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3788 {
3789   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3790   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3791   {
3792     \msg_warning:nnee { enumext } { item-joined }
3793     { \int_use:N \l__enumext_joined_item_viii_int }
3794     { \int_use:N \l__enumext_columns_viii_int }
3795     \int_set:Nn \l__enumext_joined_item_viii_int
3796     {
3797       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3798     }
3799   }
3800   \int_compare:nNnT
3801   { \l__enumext_joined_item_viii_int }
3802   >
3803   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3804   {
3805     \msg_warning:nnee { enumext } { item-joined-columns }
3806     { \int_use:N \l__enumext_joined_item_viii_int }
3807     {
3808       \int_eval:n
3809       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3810     }
3811     \int_set:Nn \l__enumext_joined_item_viii_int
3812     {
3813       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3814     }
3815   }
3816   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3817   {
3818     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3819     \int_decr:N \l__enumext_joined_item_aux_viii_int

```

```

3820 \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3821 \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3822 \dim_set:Nn \l__enumext_joined_width_viii_dim
3823 {
3824   \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3825   + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3826       + \l__enumext_columns_sep_viii_dim
3827       )*\l__enumext_joined_item_aux_viii_int
3828 }
3829 \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3830 }
3831 {
3832   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3833   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3834 }
3835 }

```

(End of definition for `__enumext_starred_joined_item_viii:n`)

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

```

3836 \cs_new_protected:Nn \__enumext_start_mini_viii:
3837 {
3838   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3839   {
3840     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3841     {
3842       \linewidth
3843       - \l__enumext_minipage_right_viii_dim
3844       - \l__enumext_minipage_hsep_viii_dim
3845     }
3846     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3847     \dim_gset_eq:NN
3848       \g__enumext_minipage_right_viii_dim
3849       \l__enumext_minipage_right_viii_dim
3850     \__enumext_mini_addvspace_viii:
3851     \nointerlineskip\noindent
3852     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3853   }
3854 }
3855 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3856 {
3857   \bool_if:NT \l__enumext_minipage_active_viii_bool
3858   {
3859     \end{__enumext_mini_env*}
3860     \hfill
3861     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3862   }
3863 }
3864 \__enumext_after_env:nn {keyans*}
3865 {
3866   \bool_if:NT \g__enumext_minipage_active_viii_bool
3867   {
3868     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3869     \par\addvspace { \g__enumext_minipage_right_skip }
3870     \bool_if:NF \g__enumext_minipage_center_viii_bool
3871     {
3872       \centering
3873     }
3874     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3875     \end{__enumext_mini_env*}
3876     \par\addvspace{ \g__enumext_minipage_after_skip }
3877   }
3878   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3879   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3880   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3881   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3882 }

```

(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.

```

3883 \NewDocumentEnvironment{keyans*}{ o }
3884 {
3885   \__enumext_safe_exec_viii:
3886   \__enumext_parse_keys_viii:n {#1}
3887   \__enumext_before_list_viii:
3888   \__enumext_start_list:nn { }
3889   {
3890     \__enumext_list_arg_two_viii:
3891     \__enumext_before_keys_exec_viii:
3892   }
3893   \__enumext_starred_columns_set_viii:
3894   \item[] \scan_stop:
3895   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3896   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3897 }
3898 {
3899   \__enumext_stop_item_tmp_viii:
3900   \__enumext_remove_extra_parsep_viii:
3901   \__enumext_check_starred_cmd:n { item }
3902   \__enumext_stop_list:
3903   \__enumext_after_list_viii:
3904 }

```

(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.

```

3905 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3906 {
3907   \int_incr:N \__enumext_keyans_level_h_int
3908   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
3909   {
3910     \msg_error:nn { enumext } { nested }
3911   }
3912   \__enumext_keyans_save_start_line:
3913   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3914   \bool_set_false:N \__enumext_store_active_bool
3915   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3916   {
3917     \msg_error:nn { enumext } { keyans-wrong-level }
3918   }
3919 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

3920 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3921 {
3922   \tl_if_novalue:nF {#1}
3923   {
3924     \keys_set:nn { enumext / keyans* } {#1}
3925   }
3926 }

```

(End of definition for `__enumext_parse_keys_viii:n`)

`__enumext_before_list_viii:` The function `__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{<code>}` defined by the **before*** key if it is active, the call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```

3927 \cs_new_protected:Nn \__enumext_before_list_viii:
3928 {
3929   \__enumext_vspace_above_viii:
3930   \__enumext_before_args_exec_viii:
3931   \__enumext_start_mini_viii:
3932 }

```

(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.

```

3933 \cs_new_protected:Nn \__enumext_after_list_viii:
3934 {
3935     \__enumext_stop_mini_viii:
3936     \__enumext_after_stop_list_viii:
3937     \__enumext_vspace_below_viii:
3938 }

```

(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`.

```

3939 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3940 {
3941     \__enumext_stop_item_tmp_viii:
3942     \int_incr:N \l__enumext_item_column_pos_viii_int
3943     \int_gincr:N \g__enumext_item_count_all_viii_int
3944     \__enumext_item_peek_args_viii:
3945 }

```

(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)`.

```

3946 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3947 {
3948     \peek_meaning:NTF (
3949         { \__enumext_joined_item_viii:w }
3950         { \__enumext_joined_item_viii:w (1) }
3951     }

```

(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`.

```

3952 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3953 {
3954     \__enumext_starred_joined_item_viii:n {#1}
3955     \peek_meaning_remove:NTF *
3956         { \__enumext_starred_item_viii:w }
3957         { \__enumext_standar_item_viii:w }
3958 }

```

(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]`.

```

3959 \cs_new_protected:Npn \__enumext_standar_item_viii:w
3960 {
3961   \bool_set_false:N \l__enumext_item_starred_viii_bool
3962   \peek_meaning:NTF [
3963     {
3964       \bool_set_eq:NN
3965       \l__enumext_wrap_label_viii_bool
3966       \l__enumext_wrap_label_opt_viii_bool
3967       \__enumext_start_item_viii:w
3968     }
3969     {
3970       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3971       \legacy_if_set_true:n { @noitemarg }
3972       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3973     }
3974   }

```

(End of definition for __enumext_standar_item_viii:w.)

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w

```

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

```

3975 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3976 {
3977   \bool_set_true:N \l__enumext_item_starred_viii_bool
3978   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3979   \peek_meaning:NTF [
3980     { \__enumext_starred_item_viii_aux_i:w }
3981     { \__enumext_starred_item_viii_aux_ii:w }
3982   }

```

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.

```

3983 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3984 {
3985   \tl_clear:N \l__enumext_store_keyans_label_tl
3986   \tl_if_no_value:nF { #1 }
3987   {
3988     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3989     {
3990       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3991       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3992     }
3993     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3994   }
3995   \__enumext_starred_item_viii_aux_ii:w
3996 }
3997 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3998 {
3999   \legacy_if_set_true:n { @noitemarg }
4000   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4001 }

```

(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 `[\langle content \rangle]` for `\item*[\langle content \rangle]` 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.

```

4002 \cs_new_protected:Nn \__enumext_starred_item_exec:
4003 {
4004   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
4005   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
4006   \__enumext_keyans_store_ref:
4007   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
4008   \__enumext_keyans_addto_seq_link:
4009   \int_gincr:N \g__enumext_check_starred_cmd_int
4010   \bool_if:NT \l__enumext_show_answer_bool
4011   {

```

```

4012     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4013   }
4014   \bool_if:NT \l__enumext_show_position_bool
4015   {
4016     \tl_set:Nx \l__enumext_mark_answer_sym_tl
4017     {
4018       \group_begin:
4019       \exp_not:N \normalfont
4020       \exp_not:N \footnotesize [ \int_eval:n
4021         {
4022           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4023         }
4024       ]
4025       \group_end:
4026     }
4027     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4028   }
4029 }

```

(End of definition for `__enumext_starred_item_exec:`.)

Real definition of `\item` in `keyans*`

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

```

4030 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4031 {
4032   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4033   \legacy_if:nT { @noitemarg }
4034   {
4035     \legacy_if_set_false:n { @noitemarg }
4036     \legacy_if:nT { @nmbrrlist }
4037     {
4038       \bool_if:NT \l__enumext_hyperref_bool
4039       {
4040         \legacy_if_set_true:n { @hyper@item }
4041       }
4042       \refstepcounter{enumXviii}
4043     }
4044   }

```

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

```

4045   \group_begin:
4046   \lrbox{ \l__enumext_item_text_viii_box }
4047   \bool_if:NF \l__enumext_footnotes_key_bool
4048   {
4049     \__enumext_renew_footnote:
4050   }
4051   \bool_if:NT \l__enumext_item_starred_viii_bool
4052   {
4053     \__enumext_starred_item_exec:
4054   }
4055   \group_begin:
4056   \tl_use:N \l__enumext_label_font_style_viii_tl
4057   \bool_if:NTF \l__enumext_wrap_label_viii_bool
4058   {
4059     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4060     { \__enumext_wrapper_label_viii:n {#1} }
4061   }
4062   {
4063     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
4064   }
4065   \group_end:
4066   \skip_horizontal:N \l__enumext_labelsep_viii_dim
4067   \tl_use:N \l__enumext_after_list_args_viii_tl
4068   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4069   \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4070   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4071   \bool_if:NT \l__enumext_item_starred_viii_bool
4072   {
4073     \tl_use:N \l__enumext_fake_item_indent_viii_tl
4074     \__enumext_keyans_show_item_opt:
4075     \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_

```

```

4076         }
4077         {
4078             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4079         }
4080     }

```

(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.

```

4081 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4082 {
4083     \__enumext_endminipage:
4084     \endlrbox
4085     \group_end:
4086     \box_set_wd:Nn \l__enumext_item_text_viii_box
4087     {
4088         \l__enumext_joined_width_viii_dim
4089         + \l__enumext_labelwidth_viii_dim
4090         + \l__enumext_labelsep_viii_dim
4091     }
4092     \int_set:Nn \hbadness { 10000 }
4093     \box_use:N \l__enumext_item_text_viii_box
4094     \bool_if:NF \l__enumext_footnotes_key_bool
4095     {
4096         \__enumext_print_footnote:
4097     }
4098     \int_compare:nNnTF
4099     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4100     {
4101         \par\noindent
4102         \int_zero:N \l__enumext_item_column_pos_viii_int
4103     }
4104     { \hspace{ \l__enumext_columns_sep_viii_dim } }
4105 }

```

(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.

```

4106 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4107 {
4108     \int_compare:nNnT
4109     {
4110         \int_mod:nn
4111         { \g__enumext_item_count_all_viii_int }
4112         { \l__enumext_columns_viii_int }
4113     }
4114     =
4115     { \c_zero_int }
4116     {
4117         \par
4118         \vspace{ -\l__enumext_itemsep_viii_skip }
4119         \int_gzero:N \g__enumext_item_count_all_viii_int
4120     }
4121 }

```

(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.

```

4122 \NewDocumentCommand \getkeyans { m }
4123 {
4124     \exp_args:Ne \__enumext_getkeyans_aux:n
4125     { \tl_to_str:e { \text_expand:n {#1} } }
4126 }

```

(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.

```

4127 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4128 {
4129   \str_if_in:nnTF {#1} { : }
4130   {
4131     \use:e
4132     {
4133       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4134       { {##1} {##2} }
4135     }
4136     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4137   }
4138   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4139 }

```

(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⟩*.

```

4140 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4141 {
4142   \prop_if_exist:cF { g__enumext_#1_prop }
4143   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4144   \group_begin:
4145   \prop_item:cn { g__enumext_#1_prop }{#2}
4146   \group_end:
4147 }

```

(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 `__enumext_print_keyans_X_tl`.

The variable `__enumext_print_keyans_starred_tl` will have the default *⟨keys⟩* for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *⟨sequence⟩* and will be set by `\setenumext[⟨print,*⟩]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[⟨print,level⟩]`

```

4148 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4149 \keys_define:nn { enumext / print }
4150 {
4151   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4152                 { \__enumext_filter_save_key:n {#1} }
4153                 \__enumext_print_keyans_starred_tl, % starred cmd
4154   print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4155   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4156                  { \__enumext_filter_save_key:n {#1} }
4157                  \__enumext_print_keyans_i_tl,
4158   print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4159   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4160                  { \__enumext_filter_save_key:n {#1} }
4161                  \__enumext_print_keyans_ii_tl,
4162   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4163   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4164                  { \__enumext_filter_save_key:n {#1} }
4165                  \__enumext_print_keyans_iii_tl,
4166   print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4167   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4168                  { \__enumext_filter_save_key:n {#1} }
4169                  \__enumext_print_keyans_iv_tl,
4170   print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4171   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4172                  { \__enumext_filter_save_key:n {#1} }
4173                  \__enumext_print_keyans_vii_tl, % starred nested
4174   print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4175 }

```

- The reason for storing $\langle keys \rangle$ 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 $\langle sequence \rangle$ for $\langle anskey \rangle$, $\langle item^* \rangle$ and $\langle anspic^* \rangle$. Within a group we will run our “precompiled keys” and then call the internal function `__enumext_printkeyans:nnn`.

```

4176 \NewDocumentCommand \printkeyans { s O{} m }
4177 {
4178   \group_begin:
4179     \tl_use:N \__enumext_print_keyans_i_tl
4180     \tl_use:N \__enumext_print_keyans_ii_tl
4181     \tl_use:N \__enumext_print_keyans_iii_tl
4182     \tl_use:N \__enumext_print_keyans_iv_tl
4183     \tl_use:N \__enumext_print_keyans_vii_tl
4184     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4185   \group_end:
4186 }

```

(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 $\langle sequence \rangle$, if it does not exist it will return an error message, then it will check if not empty.

```

4187 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4188 {
4189   \seq_if_exist:cTF { g__enumext_#3_seq }
4190   {
4191     \seq_if_empty:cF { g__enumext_#3_seq }
4192     {
4193       %%\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 $\langle sequence \rangle$, then execute the variable `__enumext_print_keyans_starred_tl` that contains the default $\langle keys \rangle$ for the environment `enumext^*`, it will open the environment `enumext^*` passing the optional argument to the first level and then will map the $\langle sequence \rangle$

```

4194     \bool_if:nTF {#1}
4195     {
4196       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext^*} }
4197       {
4198         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext^* }
4199       }
4200       {
4201         \tl_use:N \__enumext_print_keyans_starred_tl
4202         \begin{enumext^*}[#2]
4203           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4204           \end{enumext^*}
4205         }
4206       }

```

Otherwise it will open the environment `enumext` passing the optional argument to the first level and then map the $\langle sequence \rangle$.

```

4207       {
4208         \begin{enumext}[#2]
4209         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4210         \end{enumext}
4211       }
4212     }
4213   {
4214     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4215   }
4216 }
4217 }

```

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

11.40 The command \setenumext

First we define a “meta families” of $\langle keys \rangle$ to access from \setenumext.

```

4218 \keys_define:nn { enumext / meta-families }
4219 {
4220   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4221   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4222   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4223   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4224   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4225   enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4226   keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4227   print*    .code:n = { \keys_set:nn { enumext / print    } { print* = {#1} } } ,
4228   print-1   .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } } ,
4229   print-2   .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } } ,
4230   print-3   .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } } ,
4231   print-4   .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } } ,
4232   print-*   .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } } ,
4233   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4234 }

```

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

```

4235 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4236 {
4237   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4238   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4239 }

```

\setenumext Now we define the user command \setenumext.

```

4240 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4241 {
4242   \tl_if_novalue:nTF {#1}
4243   {
4244     \seq_map_inline:Nn \c__enumext_all_families_seq
4245   }
4246   {
4247     \seq_clear:N \l__enumext_setkey_tmpa_seq
4248     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4249     \int_set:Nn \l__enumext_setkey_tmpa_int
4250     {
4251       \seq_count:N \l__enumext_setkey_tmpb_seq
4252     }
4253     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4254     {
4255       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4256       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4257       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4258       {
4259         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4260       }
4261     }
4262     {
4263       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4264     }
4265     \seq_if_empty:NnTF \l__enumext_setkey_tmpa_seq
4266     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4267     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4268   }
4269   {
4270     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4271   }
4272 }

```

(End of definition for \setenumext. This function is documented on page 6.)

__enumext_set_parse:n
 __enumext_set_error:n

Internal functions used by the \setenumext command.

```

4273 \cs_new_protected:Npn \__enumext_set_parse:n #1
4274 {
4275   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4276   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4277   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }

```

```

4278     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4279     {
4280         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4281         { \tl_trim_spaces:n {#1} }
4282     }
4283     { \__enumext_set_error:nn {#1} { } }
4284 }
4285 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4286 { \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.

```

4287 \msg_new:nnn { enumext } { package-load }
4288 {
4289     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4290 }
4291 \msg_new:nnn { enumext } { package-not-load }
4292 {
4293     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4294 }
4295 \msg_new:nnn { enumext } { package-load-foot }
4296 {
4297     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4298 }

```

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

```

4299 \msg_new:nnn { enumext } { counters }
4300 {
4301     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4302     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4303 }

```

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

```

4304 \msg_new:nnn { enumext } { store-prop }
4305 {
4306     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4307 }
4308 \msg_new:nnn { enumext } { store-seq }
4309 {
4310     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4311 }
4312 \msg_new:nnn { enumext } { store-int }
4313 {
4314     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4315 }
4316 \msg_new:nnn { enumext } { prop-seq-int-hook }
4317 {
4318     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4319     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4320     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4321 }
4322 \msg_new:nnn { enumext } { item-answer-hook }
4323 {
4324     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4325     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4326     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4327 }

```

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

```

4328 \msg_new:nnn { enumext } { invalid-key }
4329 {
4330     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4331 }
4332 \msg_new:nnn { enumext } { unknown-key-family }
4333 {
4334     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4335 }

```

Messages used in length calculation.

```

4336 \msg_new:nnn { enumext } { width-negative }

```

```

4337 {
4338     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4339     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4340 }
4341 \msg_new:nnn { enumext } { width-zero }
4342 {
4343     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4344     The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4345 }

```

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

```

4346 \msg_new:nnn { enumext } { list-lengths }
4347 {
4348     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
4349     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4350     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4351     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4352     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4353     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4354     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4355     \__enumext_show_length:nnn { skip } { topsep } {#1}
4356     \__enumext_show_length:nnn { skip } { parsep } {#1}
4357     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4358     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4359     *****
4360 }

```

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

```

4361 \msg_new:nnn { enumext } { list-lengths-not-nested }
4362 {
4363     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
4364     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4365     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4366     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4367     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4368     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4369     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4370     \__enumext_show_length:nnn { skip } { topsep } {#1}
4371     \__enumext_show_length:nnn { skip } { parsep } {#1}
4372     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4373     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4374     *****
4375 }

```

Messages used by `ref` key.

```

4376 \msg_new:nnn { enumext } { key-ref-empty }
4377 {
4378     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4379 }

```

Messages used by `save-ans` key.

```

4380 \msg_new:nnn { enumext } { save-ans-empty }
4381 {
4382     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4383 }
4384 \msg_new:nnn { enumext } { save-ans-log }
4385 {
4386     * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save
4387     ans=#2 ~ \msg_line_context:.
4388 }
4389 \msg_new:nnn { enumext } { save-ans-log-hook }
4390 {
4391     * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save
4392     ans=#2 ~ \msg_line_context:.
4393 }
4394 \msg_new:nnn { enumext } { save-ans-hook }
4395 {
4396     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4397 }

```

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

```

4396 \msg_new:nnn { enumext } { need-save-ans }
4397 {

```

```

4398     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4399 }
4400 \msg_new:nnn { enumext } { items-same-answer }
4401 {
4402     *****\
4403     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str
4404     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\
4405     *****
4406 }
4407 \msg_new:nnn { enumext } { item-greater-answer }
4408 {
4409     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4410     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
4411     Items ~ > ~ Answers.
4412 }
4413 \msg_new:nnn { enumext } { item-less-answer }
4414 {
4415     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4416     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
4417     Items ~ < ~ Answers.
4418 }

```

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

```

4419 \msg_new:nnn { enumext } { missing-starred }
4420 {
4421     Missing ~ '\c_backslash_str #1*' ~ #2.
4422 }
4423 \msg_new:nnn { enumext } { many-starred }
4424 {
4425     Many ~ '\c_backslash_str #1*' ~ #2.
4426 }

```

Messages used by `\printkeyans*` command.

```

4427 \msg_new:nnn { enumext } { print-starred }
4428 {
4429     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4430     #2 ~ environment ~ \msg_line_context:.
4431 }

```

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

```

4432 \msg_new:nnn { enumext } { list-too-deep }
4433 {
4434     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
4435     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4436 }

```

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

```

4437 \msg_new:nnn { enumext } { anskey-empty-arg }
4438 {
4439     Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4440 }
4441 \msg_new:nnn { enumext } { anskey-wrong-place }
4442 {
4443     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
4444     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4445 }
4446 \msg_new:nnn { enumext } { anskey-nested }
4447 {
4448     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4449 }
4450 \msg_new:nnn { enumext } { anskey-nested-env }
4451 {
4452     The ~ environment ~ anskey* ~ can't ~ be ~ nested ~ \msg_line_context:.
4453 }
4454 \msg_new:nnn { enumext } { anspic-wrong-place }
4455 {
4456     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
4457     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4458 }
4459 \msg_new:nnn { enumext } { command-wrong-place }
4460 {
4461     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
4462     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.

```

4463 }

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

```
4464 \msg_new:nnn { enumext } { keyans-nested }
4465 {
4466   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4467 }
4468 \msg_new:nnn { enumext } { keyans-wrong-level }
4469 {
4470   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4471   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4472 }
4473 \msg_new:nnn { enumext } { wrong-place }
4474 {
4475   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
4476   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4477 }
4478 \msg_new:nnn { enumext } { keyanspic-nested }
4479 {
4480   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
4481 }
4482 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4483 {
4484   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4485   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4486 }
```

Messages used by `\getkeyans` command.

```
4487 \msg_new:nnn { enumext } { undefined-storage-anskey }
4488 {
4489   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4490 }
```

Messages used by `\miniright` command.

```
4491 \msg_new:nnn { enumext } { missing-miniright }
4492 {
4493   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
4494   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4495 }
4496 \msg_new:nnn { enumext } { wrong-miniright-place }
4497 {
4498   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4499   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4500 }
4501 \msg_new:nnn { enumext } { wrong-miniright-use }
4502 {
4503   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4504   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4505 }
```

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

```
4506 \msg_new:nnn { enumext } { nested }
4507 {
4508   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4509 }
4510 \msg_new:nnn { enumext } { item-joined }
4511 {
4512   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
4513 }
4514 \msg_new:nnn { enumext } { item-joined-columns }
4515 {
4516   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
4517 }
```

11.42 Finish package

Finish package implementation.

```
4518 \file_input_stop:
4519 </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>	2459
<code>*</code>	211
<code>\+</code>	203
<code>\-</code>	203
<code>\\</code>	219, 3313, 4301, 4318, 4319, 4324, 4325, 4338, 4343, 4348, 4363, 4402, 4403, 4404, 4409, 4410, 4415, 4416, 4434, 4443, 4456, 4461, 4470, 4475, 4484, 4493, 4498, 4503
A	
<code>above</code>	<u>1380</u>
<code>above*</code>	<u>1380</u>
<code>\addvspace</code>	1034, 1062, 1178, 1257, 1320, 1326, 1354, 1371, 3152, 3167, 3269, 3284, 3510, 3517, 3869, 3876
<code>after</code>	<u>873</u>
<code>align</code>	<u>490</u>
<code>\Alph</code>	35, <u>40</u>
<code>\Alph</code>	442, 551, 596, 664, 4170
<code>\alph</code>	35, <u>40</u>
<code>\alph</code>	443, 549, 4162
<code>\anskey</code>	12, 71, <u>2212</u> , 2517, 2522
<code>anskey*</code>	13, <u>2428</u>
<code>\anspic</code>	15, 93, 94, <u>3291</u>
<code>\anspic*</code>	66
<code>\arabic</code>	29, 35
<code>\arabic</code>	441, 548, 595, 4154, 4158, 4174
B	
<code>\baselineskip</code>	47
<code>\baselineskip</code>	2157, 2165
<code>before</code>	<u>873</u>
<code>before*</code>	<u>873</u>
<code>below</code>	<u>1380</u>
<code>below*</code>	<u>1380</u>
bool commands:	
<code>\bool_gset_false:N</code>	317, 318, 319, 2529, 2531, 3519, 3523, 3878
<code>\bool_gset_true:N</code>	231, 240, 976, 1872, 1878, 3502, 3520, 3861, 3879
<code>\bool_if:NTF</code>	382, 394, 411, 1402, 1416, 1429, 1440, 1451, 1462, 1473, 1484, 1537, 1554, 1559, 1567, 1594, 1632, 1637, 1644, 1648, 1670, 1675, 1683, 1690, 1721, 1729, 1822, 1946, 2030, 2040, 2119, 2143, 2150, 2178, 2216, 2226, 2254, 2280, 2397, 2408, 2412, 2484, 2558, 2573, 2648, 2659, 2663, 2776, 2806, 2880, 2896, 2958, 2968, 2998, 3003, 3086, 3136, 3150, 3158, 3197, 3254, 3267, 3275, 3293, 3498, 3507, 3511, 3588, 3598, 3681, 3686, 3694, 3698, 3713, 3742, 3857, 3866, 3870, 4010, 4014, 4038, 4047, 4051, 4057, 4071, 4094
<code>\bool_if:nTF</code>	1355, 1372, 2817, 2852, 2916, 3314, 4194
<code>\bool_if_p:N</code>	249, 263, 1701, 1702, 1710, 1711, 1835, 1857, 1869, 1870, 1875, 1876, 2265, 2306, 2307, 2331, 2340, 2341, 2353, 2369, 2470, 2635, 2636, 2673, 2674, 3059, 3072, 3074, 3321, 3322
<code>\bool_lazy_all:nTF</code>	247, 261, 1833, 1855, 2329, 2338, 2351, 2367, 3057, 3070
<code>\bool_lazy_and:nnTF</code>	227, 236, 1700, 1709, 1868, 1874, 2264, 2271, 2305, 2469, 2475, 2634
<code>\bool_lazy_or:nnTF</code>	1762, 1769, 2672, 3320
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 60, 70, 91, 96, 97, 102, 103, 106, 125, 128, 131, 132, 141, 142, 143, 151, 152, 166, 177, 179
<code>\bool_not_p:n</code>	228, 237, 2266, 2272, 2356, 2371, 2471, 2476, 3060, 3061, 3073
<code>\bool_set_eq:NN</code>	2784, 2832, 3631, 3964
<code>\bool_set_false:N</code>	391, 1807, 1808, 3173, 3205, 3287, 3352, 3370, 3583, 3628, 3914, 3961
<code>\bool_set_true:N</code>	254, 268, 373, 377, 483, 801, 1386, 1391, 1657, 1779, 1780, 2062, 2070, 2780, 2810, 2828, 2840, 3035, 3066, 3079, 3105, 3202, 3229, 3487, 3557, 3637, 3644, 3645, 3846, 3970, 3977, 3978
box commands:	
<code>\box_dp:N</code>	1074, 1078, 1082, 1093, 1097, 1108, 1117, 1123, 1133, 1146, 1152, 1158, 1189, 1190, 1191, 1194, 1204, 1208, 1217, 1224, 1229, 1237, 1266, 1267, 1270, 1277, 1290, 1298, 1304, 1312, 3382
<code>\box_new:N</code>	67, 172
<code>\box_set_wd:Nn</code>	3734, 4086
<code>\box_use:N</code>	3741, 4093
<code>\box_wd:N</code>	449
C	
<code>\c</code>	211, 212, 701, 703, 715, 717
<code>\cB</code>	212
<code>\cE</code>	212
<code>\centering</code>	1357, 1374, 3403, 3513, 3872
<code>check-ans</code>	<u>1799</u>
Document class:	
<code>article</code>	41
clist commands:	
<code>\clist_const:Nn</code>	184
<code>\clist_map_function:nN</code>	3390
<code>\clist_map_inline:Nn</code>	489, 743, 806, 872, 887, 968, 1396
<code>\clist_map_inline:nn</code>	45, 56, 75, 81, 93, 105, 130, 160, 183, 514, 531, 811, 982, 1502, 1746, 1813, 2009, 2027, 2059, 2326, 2567, 2734, 2945, 2948, 2975, 2985, 2988, 3008, 4276
<code>\columnbreak</code>	72
<code>\columnbreak</code>	2268
<code>columns</code>	<u>952</u>
<code>columns-sep</code>	<u>952</u>
<code>\columnsep</code>	89, 92
<code>\columnsep</code>	3130, 3251
<code>\columnseprule</code>	89, 92
<code>\columnseprule</code>	3134, 3253
Commands provide by enumext :	
<code>\anskey</code>	27, 62, 67, 69, 71, 73–75, 77, 79, 88, 100, 110, 112, 116
<code>\anspic*</code>	27, 28, 66, 69, 77, 78, 94–96, 110, 112
<code>\anspic</code>	69, 93–96, 116
<code>\getkeyans</code>	69, 110, 117
<code>\item*</code>	27, 28, 66, 69, 77, 78, 82, 83, 101, 108, 110, 112
<code>\itemwidth</code>	97, 104
<code>\item</code>	81, 83, 97, 100–102, 104, 107
<code>\miniright</code>	26, 45, 52, 53, 89, 90, 92, 93, 117
<code>\printkeyans*</code>	111
<code>\printkeyans</code>	27, 69, 111, 112
<code>\setenumext</code>	27, 112–114

Counters defined by `enumext`:

<code>enumXiii</code>	25, 34
<code>enumXii</code>	25, 34
<code>enumXiv</code>	25, 34
<code>enumXi</code>	25, 34
<code>enumXviii</code>	25, 34
<code>enumXvii</code>	25, 34, 102
<code>enumXvi</code>	25, 34
<code>enumXv</code>	25, 34

cs commands:

<code>\cs_generate_variant:Nn</code>	451, 467, 707, 723, 2111, 2116, 2196, 2448, 2935, 3392, 4148
<code>\cs_if_exist:NTF</code>	421
<code>\cs_new:Nn</code>	197
<code>\cs_new:Npn</code>	215, 1503, 1512, 1521, 2074, 2083, 2091
<code>\cs_new_eq:NN</code>	344, 345, 346, 350, 351, 396, 397, 400, 401
<code>\cs_new_protected:Nn</code>	207, 221, 245, 276, 303, 309, 315, 321, 327, 335, 353, 368, 572, 635, 687, 888, 892, 896, 900, 904, 908, 912, 916, 920, 924, 928, 932, 936, 940, 944, 948, 983, 995, 1019, 1036, 1047, 1064, 1139, 1163, 1180, 1242, 1259, 1281, 1316, 1322, 1397, 1411, 1425, 1436, 1447, 1458, 1469, 1480, 1565, 1668, 1681, 1698, 1719, 1747, 1752, 1777, 1818, 1828, 1866, 1881, 1888, 1897, 1902, 1907, 1912, 1921, 1926, 1931, 1936, 2117, 2141, 2148, 2176, 2183, 2224, 2317, 2449, 2467, 2509, 2527, 2556, 2571, 2599, 2632, 2668, 2680, 2688, 2739, 2743, 2762, 2813, 2848, 2864, 2874, 2890, 3028, 3055, 3084, 3091, 3114, 3144, 3156, 3195, 3219, 3237, 3262, 3273, 3310, 3354, 3368, 3388, 3393, 3409, 3477, 3496, 3548, 3570, 3577, 3586, 3596, 3613, 3753, 3768, 3836, 3855, 3905, 3927, 3933, 3946, 4002, 4106
<code>\cs_new_protected:Npn</code>	189, 193, 404, 419, 436, 446, 452, 552, 597, 669, 694, 708, 1344, 1363, 1533, 1552, 1622, 1655, 1757, 1957, 2028, 2038, 2060, 2068, 2103, 2112, 2239, 2251, 2394, 2406, 2428, 2535, 2609, 2653, 2772, 2790, 2824, 2836, 2904, 2938, 2978, 3038, 3215, 3363, 3428, 3560, 3619, 3626, 3642, 3650, 3655, 3667, 3787, 3920, 3952, 3959, 3975, 3983, 3997, 4127, 4140, 4187, 4273, 4285
<code>\cs_new_protected_nopar:Nn</code>	3606, 3729, 3939, 4081
<code>\cs_new_protected_nopar:Npn</code>	3673, 4030
<code>\cs_set:Nn</code>	2399
<code>\cs_set:Npn</code>	2327, 2365, 4133
<code>\cs_set_eq:NN</code>	3538, 3539, 3675, 3895, 3896, 4032
<code>\cs_set_protected:Nn</code>	812, 828, 840, 852
<code>\cs_set_protected:Npn</code>	41, 50, 68, 76, 88, 94, 122, 156, 164, 468, 490, 519, 532, 579, 724, 744, 788, 807, 864, 873, 952, 969, 1380, 1491, 1738, 1799, 1974, 2010, 2046, 2319, 2560, 2723, 2936, 2976
<code>\cs_to_str:N</code>	438, 461
<code>\cs_undefine:N</code>	1951, 1952

D

<code>\d</code>	203
<code>\DeclareDocumentEnvironment</code>	357

dim commands:

<code>\dim_abs:n</code>	2909, 2914
<code>\dim_add:Nn</code>	3373
<code>\dim_compare:nNnTF</code>	814, 830, 842, 854, 1346, 1365, 2906, 2911, 2917, 2923, 2925, 2927, 3096, 3119, 3223, 3241, 3365, 3411, 3479, 3770, 3838
<code>\dim_compare:nTF</code>	2290, 2497
<code>\dim_gset_eq:NN</code>	3488, 3847
<code>\dim_gzero:N</code>	2533, 3522, 3881

<code>\dim_new:N</code>	64, 71, 72, 73, 90, 137, 173, 174, 180
<code>\dim_set:Nn</code>	449, 802, 2804, 2909, 2914, 2916, 2919, 2920, 2924, 2926, 2929, 2930, 2932, 3099, 3122, 3225, 3243, 3395, 3413, 3420, 3463, 3481, 3669, 3772, 3779, 3822, 3840
<code>\dim_set_eq:NN</code>	539, 586, 657, 661, 2799, 2947, 2987, 3130, 3251, 3470, 3473, 3474, 3660, 3829, 3832, 3833
<code>\dim_use:N</code>	815, 823, 1347, 1353, 2186, 2189, 2194, 2869, 2871, 3097, 3102, 3103, 3110, 3120, 3124, 3125, 3127
<code>\dim_zero:N</code>	3134, 3253, 3374, 3375, 3376
<code>\dim_zero_new:N</code>	3426, 3785
<code>\c_zero_dim</code>	817, 831, 843, 855, 1347, 1365, 2292, 2499, 2906, 2911, 2917, 2924, 3097, 3120, 3223, 3241, 3411, 3479, 3770, 3838

E

<code>\end</code>	1350, 1368, 2145, 2180, 3149, 3166, 3266, 3283, 3500, 3516, 3859, 3875, 4196, 4204, 4210
<code>\endlist</code>	32
<code>\endlist</code>	345
<code>\endlrbox</code>	3732, 4084
<code>\endminipage</code>	32
<code>\endminipage</code>	351
<code>enumext</code>	5, 3009

enumext internal commands:

<code>\l__enumext__check_start_line_env_tl</code>	31
<code>\l__enumext__ref_the_count_tl</code>	37
<code>\l__enumext__resume_name_tl</code>	58
<code>__enumext_add_pre_parsep:</code>	46, 993, 995, 995
<code>__enumext_after_args_exec:</code>	44, 888, 900, 3021
<code>__enumext_after_args_exec_v:</code>	44, 45, 904, 916, 3188
<code>__enumext_after_args_exec_vii:</code>	920, 944
<code>__enumext_after_args_exec_viii:</code>	948
<code>__enumext_after_env:nn</code>	65, 91, 104, 193, 193, 2451, 3176, 3505, 3767, 3864
<code>__enumext_after_hyperref:</code>	33, 366, 368, 368
<code>__enumext_after_list:</code>	90, 100, 107, 3026, 3156, 3156
<code>\l__enumext_after_list_args_v_tl</code>	918
<code>\l__enumext_after_list_args_vii_tl</code>	946, 3723
<code>\l__enumext_after_list_args_viii_tl</code>	950, 4067
<code>__enumext_after_list_v:</code>	93, 3193, 3273, 3273
<code>__enumext_after_list_vii:</code>	3546, 3577, 3577
<code>__enumext_after_list_viii:</code>	3903, 3933, 3933
<code>__enumext_after_star_env:nn</code>	98
<code>__enumext_after_stop_list:</code>	44, 45, 888, 896, 3171
<code>__enumext_after_stop_list_v:</code>	44, 904, 912, 3288
<code>\l__enumext_after_stop_list_v_tl</code>	914
<code>__enumext_after_stop_list_vii:</code>	920, 936, 3580
<code>\l__enumext_after_stop_list_vii_tl</code>	938
<code>__enumext_after_stop_list_viii:</code>	940, 3936
<code>\l__enumext_after_stop_list_viii_tl</code>	942
<code>\l__enumext_align_label_vii_str</code>	3715, 3719
<code>\l__enumext_align_label_viii_str</code>	4059, 4063
<code>\l__enumext_align_label_X_str</code>	164
<code>\c__enumext_all_envs_clist</code>	184, 489, 743, 806, 872, 887, 968, 1396
<code>\c__enumext_all_families_seq</code>	113, 4235, 4244, 4266
<code>\l__enumext_anskey_level_int</code>	28, 2245, 2246
<code>__enumext_anskey_safe_inner:n</code>	71, 2219, 2224, 2239

__enumext_anskey_safe_outer: . 71, 2214, 2224, 2224
 __enumext_anskey_wrapper:n 1978, 2404
 __enumext_at_begin_document:n . . 32, 189, 189, 342, 348
 __enumext_before_args_exec: 43, 888, 888, 3094
 __enumext_before_args_exec_v: . . 44, 904, 904, 3222
 __enumext_before_args_exec_vii: . . 920, 920, 3574
 __enumext_before_args_exec_viii: 924, 3930
 __enumext_before_keys_exec: 43, 888, 892, 3019
 __enumext_before_keys_exec_v: . . 44, 904, 908, 3186
 __enumext_before_keys_exec_vii 920
 __enumext_before_keys_exec_vii: 44, 928, 3534
 __enumext_before_keys_exec_viii: . . 44, 932, 3891
 __enumext_before_list: . . . 89, 3013, 3091, 3091
 __enumext_before_list_v: . 92, 3181, 3219, 3219
 __enumext_before_list_vii: . . . 100, 3529, 3570, 3570
 __enumext_before_list_viii: . . 106, 3887, 3927, 3927
 \l__enumext_before_no_starred_key_v_tl 910
 \l__enumext_before_no_starred_key_vii_tl 930
 \l__enumext_before_no_starred_key_viii_tl 934
 \l__enumext_before_starred_key_v_tl . . . 906
 \l__enumext_before_starred_key_vii_tl . 922
 \l__enumext_before_starred_key_viii_tl 926
 __enumext_calc_hspace:NNNNNNN 85, 2904, 2904, 2935, 2940, 2980
 __enumext_check_ans_active: 63, 89, 1818, 1818, 3095, 3573
 \g__enumext_check_ans_item_tl 79
 \g__enumext_check_ans_key_bool 64, 65, 141, 317, 1872, 1878, 1946
 \l__enumext_check_ans_key_bool 64, 81, 82, 141, 1803, 1808, 1869, 1875
 __enumext_check_ans_key_hook: 64, 1866, 1866, 3170, 3581
 __enumext_check_ans_level: 63, 1818, 1824, 1828
 __enumext_check_ans_log: 64, 65, 1912, 1912, 1950
 __enumext_check_ans_log_msg_greater: 1912, 1918, 1931
 __enumext_check_ans_log_msg_less: 1912, 1916, 1921
 __enumext_check_ans_log_msg_same_ok: 1912, 1917, 1926
 __enumext_check_ans_msg_greater: 1888, 1894, 1907
 __enumext_check_ans_msg_less: 1888, 1892, 1897
 __enumext_check_ans_msg_same_ok: 1888, 1893, 1902
 __enumext_check_ans_show: . . 64, 65, 1888, 1888, 1948
 \g__enumext_check_ans_show_bool 90
 \l__enumext_check_answers_bool 62, 63, 71, 141, 1780, 1807, 1822, 2119, 2143, 2150, 2178, 2216, 2648, 2776, 2806, 3686
 __enumext_check_starred_cmd:n 31, 66, 79, 1957, 1957, 3191, 3349, 3901
 \g__enumext_check_starred_cmd_int 141, 1960, 1966, 1971, 2846, 3319, 4009
 \l__enumext_check_start_line_env_tl 141, 282, 289, 296, 1963, 1969, 1972
 \l__enumext_columns_sep_v_dim 3241, 3243, 3251
 \l__enumext_columns_sep_vii_dim . . 3411, 3413, 3422, 3467, 3751
 \l__enumext_columns_sep_viii_dim . 3770, 3772, 3781, 3826, 4104
 \l__enumext_columns_v_int 1185, 3239, 3247, 3259, 3264
 \l__enumext_columns_vii_int . . 3416, 3419, 3423, 3431, 3435, 3438, 3444, 3450, 3454, 3746, 3757
 \l__enumext_columns_viii_int . 3775, 3778, 3782, 3790, 3794, 3797, 3803, 3809, 3813, 4099, 4112
 \l__enumext_counter_i_tl 41, 428
 \l__enumext_counter_ii_tl 41, 429
 \l__enumext_counter_iii_tl 41, 430
 \l__enumext_counter_iv_tl 41, 431
 \c__enumext_counter_style_tl 29, 46, 209
 \g__enumext_counter_styles_tl . 26, 35, 64, 439, 457
 \l__enumext_counter_v_tl 41, 432, 677
 \l__enumext_counter_vi_tl 41, 433
 \l__enumext_counter_vii_tl 41, 434, 607
 \l__enumext_counter_viii_tl 41, 435, 624
 \l__enumext_current_widest_dim 26, 64, 463, 540, 587, 658, 662
 __enumext_default_item:n . . . 2772, 2772, 2821
 __enumext_define_counters:Nn 25, 419, 419, 428, 429, 430, 431, 432, 433, 434, 435
 __enumext_endminipage: . 32, 348, 351, 363, 3405, 3731, 4083
 \g__enumext_envir_name_tl 30, 146, 255, 269, 325, 1750, 1755, 1765, 1900, 1905, 1910, 1924, 1929, 1934
 __enumext_execute_after_env: 31, 32, 61, 64, 65, 1936, 1936, 3176, 3767
 __enumext_fake_item: 812, 812, 2967
 \l__enumext_fake_item_indent_v_dim 831, 836
 \l__enumext_fake_item_indent_v_tl 833, 2829, 2833, 2841
 \l__enumext_fake_item_indent_vii_dim 843, 848
 \l__enumext_fake_item_indent_vii_tl 845, 3727
 \l__enumext_fake_item_indent_viii_dim . 855, 860, 4075
 \l__enumext_fake_item_indent_viii_tl . . 857, 4073, 4078
 \l__enumext_fake_item_indent_X_tl 94
 __enumext_fake_item_vii: . . . 812, 840, 2997
 __enumext_fake_item_viii: . . . 812, 852, 3002
 __enumext_filter_save_key:n . . 68, 2035, 2043, 2066, 2072, 2074, 2074, 4152, 4156, 4160, 4164, 4168, 4172
 __enumext_filter_save_key_key:n . . 68, 2074, 2079, 2083
 __enumext_filter_save_key_pair:nn 68, 2074, 2080, 2091
 __enumext_filter_series:n 56, 1503, 1503, 1545, 1557, 1562
 __enumext_filter_series_key:n 56, 1503, 1508, 1512
 __enumext_filter_series_pair:nn . . 56, 1503, 1509, 1521
 \g__enumext_footnote_arg_seq . 161, 2745, 2758, 2768
 \g__enumext_footnote_int . 161, 2752, 2755, 2757,

2759
 \g__enumext_footnote_int_seq . 161, 2746, 2759, 2764, 2767
 __enumext_footnotes_key_bool 33
 \l__enumext_footnotes_key_bool 28, 33, 102, 151, 377, 382, 391, 3694, 3742, 4047, 4094
 __enumext_footnotetext:nn . . . 2739, 2739, 2769
 __enumext_getkeyans:nn . . 111, 4136, 4140, 4140
 __enumext_getkeyans_aux:n 111, 4124, 4127, 4127
 \l__enumext_hyperref_bool 28, 33, 151, 373, 394, 411, 2307, 2636, 3681, 4038
 __enumext_hypertarget:nn 33, 368, 396, 400, 416
 __enumext_if_is_int:n 201
 __enumext_if_is_int:nTF 201, 696, 710
 __enumext_internal_mini_page: 33, 75, 353, 353, 3030, 3550
 __enumext_is_not_nested: 30, 87, 221, 221, 3031, 3551
 __enumext_is_on_first_level: . 30, 88, 221, 245, 3036, 3558
 \g__enumext_item_anskey_int 71, 79, 141, 312, 339, 340, 1885, 2218, 2650
 __enumext_item_answer_diff: 64, 65, 1881, 1881, 1943
 \g__enumext_item_answer_diff_int . 64, 65, 150, 313, 1883, 1890, 1914
 \l__enumext_item_column_pos_vii_int 100, 3438, 3444, 3450, 3454, 3461, 3609, 3746, 3749
 \l__enumext_item_column_pos_viii_int . . 107, 3797, 3803, 3809, 3813, 3820, 3942, 4099, 4102
 __enumext_item_column_pos_X_int 164
 \g__enumext_item_count_all_vii_int 100, 3462, 3610, 3757, 3764
 \g__enumext_item_count_all_viii_int 107, 3821, 3943, 4111, 4119
 \g__enumext_item_count_all_X_int 164
 \g__enumext_item_number_int . . 63, 141, 311, 338, 340, 1839, 1843, 1846, 1849, 1861, 1885, 2778, 2808, 3688
 __enumext_item_peek_args_vii: 100, 101, 3611, 3613, 3613
 __enumext_item_peek_args_viii: . . 107, 3944, 3946, 3946
 __enumext_item_starred: . . 84, 2864, 2864, 2882
 \l__enumext_item_starred_vii_bool 3628, 3644, 3698
 \l__enumext_item_starred_viii_bool 3961, 3977, 4051, 4071
 \l__enumext_item_starred_X_bool 164
 __enumext_item_std:w 32, 81–83, 96, 342, 346, 2781, 2787, 2811, 2829, 2833, 2841, 3386
 \g__enumext_item_symbol_aux_vii_tl 3652, 3700, 3703, 3707, 3709
 \g__enumext_item_symbol_aux_X_tl 164
 \l__enumext_item_symbol_sep_vii_dim . . 3661, 3669, 3706, 3708
 \g__enumext_item_symbol_tl 25, 82, 57, 2796, 2870, 2887
 \l__enumext_item_symbol_vii_tl 3703
 \l__enumext_item_text_vii_box 3693, 3734, 3741
 \l__enumext_item_text_viii_box 4046, 4086, 4093
 \l__enumext_item_text_X_box 164
 \l__enumext_item_width_vii_dim . . 3420, 3465, 3473, 3474
 \l__enumext_item_width_viii_dim . . 3779, 3824, 3832, 3833
 \l__enumext_item_width_X_dim 164
 \l__enumext_itemindent_X_dim 68
 \l__enumext_itemsep_vii_skip 3763
 \l__enumext_itemsep_viii_skip 4118
 \l__enumext_joined_item_aux_vii_int . . 3459, 3460, 3461, 3462, 3468
 \l__enumext_joined_item_aux_viii_int . 3818, 3819, 3820, 3821, 3827
 \l__enumext_joined_item_aux_X_int 164
 __enumext_joined_item_vii:w . . 101, 3616, 3617, 3619, 3619
 \l__enumext_joined_item_vii_int . . 3430, 3431, 3434, 3436, 3442, 3447, 3452, 3457, 3459, 3465
 __enumext_joined_item_viii:w . 107, 3949, 3950, 3952, 3952
 \l__enumext_joined_item_viii_int . 3789, 3790, 3793, 3795, 3801, 3806, 3811, 3816, 3818, 3824
 \l__enumext_joined_item_X_int 164
 \l__enumext_joined_width_vii_dim . 3463, 3470, 3473, 3724, 3736
 \l__enumext_joined_width_viii_dim 3822, 3829, 3832, 4068, 4088
 \l__enumext_joined_width_X_dim 164
 __enumext_keyans_addto_prop:n 77, 2535, 2535, 2843, 3316
 __enumext_keyans_addto_seq:n . 78, 2609, 2609, 2845, 3318
 __enumext_keyans_addto_seq_link: 2609, 2630, 2632, 4008
 __enumext_keyans_anspic_code:nnn . 94, 3307, 3310, 3310
 __enumext_keyans_default_item:n . . 83, 2824, 2824, 2860
 \l__enumext_keyans_env_bool 34, 3060, 3073, 3202, 3287
 __enumext_keyans_fake_item: . . 812, 828, 2957
 \l__enumext_keyans_item_opt_tl 108, 106, 2657, 2670, 2676, 3993
 \l__enumext_keyans_level_h_int . . 28, 617, 644, 2587, 3907, 3908
 \l__enumext_keyans_level_int . . 28, 1338, 2230, 2582, 3201, 3206, 3301
 __enumext_keyans_make_label: 36, 84, 2890, 2890, 2955
 __enumext_keyans_mini_addvspace: 51, 92, 1242, 1242, 3231
 __enumext_keyans_mini_right_cmd:n 53, 1340, 1363, 1363
 __enumext_keyans_mini_set_vskip: . 50, 1180, 1180, 1244
 __enumext_keyans_multi_addvspace: 93, 1036, 1047, 3256
 __enumext_keyans_multi_set_vskip: 47, 1036, 1036, 1049
 __enumext_keyans_multicols_start: 92, 3235, 3237, 3237
 __enumext_keyans_multicols_stop: . 93, 1367, 3262, 3262, 3286
 __enumext_keyans_parse_keys:n 3180, 3215, 3215
 \l__enumext_keyans_pic_above_int . 136, 3396, 3397, 3399
 \l__enumext_keyans_pic_above_skip . . 96, 136, 3340, 3380

__enumext_keyans_pic_arg_two: 95, 3338, 3368, 3368
 \l__enumext_keyans_pic_below_int . 136, 3396, 3397, 3400
 \l__enumext_keyans_pic_body_seq .. 94-96, 136, 3305, 3345, 3404
 __enumext_keyans_pic_do:n 96, 3345, 3347, 3388, 3388, 3392
 \l__enumext_keyans_pic_level_int .. 28, 1330, 2234, 2538, 2577, 2612, 2690, 3356, 3357
 __enumext_keyans_pic_row:n 96, 3390, 3393, 3393
 __enumext_keyans_pic_safe_exec: .. 95, 3334, 3354, 3354
 __enumext_keyans_pic_skip_abs:N .. 95, 3363, 3363, 3379
 \l__enumext_keyans_pic_width_dim . 136, 3395, 3402
 __enumext_keyans_redefine_item: .. 83, 2848, 2848, 2954
 __enumext_keyans_ref: 39, 669, 687, 2956
 __enumext_keyans_ref:n 39, 666, 669, 669
 __enumext_keyans_safe_exec: . 3179, 3195, 3195
 __enumext_keyans_save_start_line: . 31, 276, 276, 3203, 3361, 3912
 __enumext_keyans_show_ans: .. 2653, 2661, 2680
 __enumext_keyans_show_item_opt: . 2653, 2668, 2841, 3330, 4074
 __enumext_keyans_show_left:n . 83, 2653, 2653, 2839, 3325
 __enumext_keyans_show_pos: .. 2653, 2665, 2688
 __enumext_keyans_starred_item:n .. 83, 2836, 2836, 2856
 __enumext_keyans_store_ref: .. 77, 2556, 2556, 2844, 3317, 4006
 __enumext_keyans_store_ref_aux_i: 78, 2556, 2568, 2571
 __enumext_keyans_store_ref_aux_ii: 78, 2556, 2597, 2599
 \l__enumext_keyans_tmpa_tl . 27, 106, 2838, 2842
 __enumext_keyans_wrapper_opt:n .. 1981, 2676
 \l__enumext_label_copy_i_tl .. 2361, 2575, 2580, 2585, 2590
 \l__enumext_label_copy_v_tl 2585
 \l__enumext_label_copy_vi_tl 2580
 \l__enumext_label_copy_vii_tl 2336, 2347, 2378, 2575
 \l__enumext_label_copy_viii_tl 2590
 \l__enumext_label_copy_X_tl 153
 \l__enumext_label_fill_left_v_tl 2894
 \l__enumext_label_fill_left_X_tl 94
 \l__enumext_label_fill_right_v_tl 2901
 \l__enumext_label_fill_right_X_tl 94
 \l__enumext_label_font_style_v_tl 2895, 3329
 \l__enumext_label_font_style_vii_tl ... 3712
 \l__enumext_label_font_style_viii_tl .. 4056
 \l__enumext_label_i_tl 532
 \l__enumext_label_ii_tl 532
 \l__enumext_label_iii_tl 532
 \l__enumext_label_iv_tl 532
 __enumext_label_style:Nnn 25, 35, 452, 452, 467, 537, 584, 655, 659
 \l__enumext_label_v_tl .. 77, 78, 652, 2543, 2617, 2682, 2717, 2838, 2842, 3183, 3324, 3326
 \l__enumext_label_vi_tl . 77, 78, 652, 2540, 2614, 3324, 3326, 3330
 \l__enumext_label_vii_tl . 579, 3639, 3664, 3671
 \l__enumext_label_viii_tl 579, 3972, 4000, 4004
 \l__enumext_label_width_by_box .. 64, 448, 449
 __enumext_label_width_by_box:Nn 35, 446, 446, 451, 463, 720
 \l__enumext_labelsep_i_dim ... 2685, 2720, 4012, 4027
 \l__enumext_labelsep_v_dim 3246
 \l__enumext_labelsep_vii_dim . 3415, 3424, 3466, 3662, 3722, 3738
 \l__enumext_labelsep_viii_dim 3774, 3783, 3825, 4066, 4075, 4090
 \l__enumext_labelwidth_i_dim . 2685, 2720, 4012, 4027
 \l__enumext_labelwidth_v_dim 3246
 \l__enumext_labelwidth_vii_dim ... 3415, 3423, 3466, 3715, 3719, 3737
 \l__enumext_labelwidth_viii_dim .. 3774, 3782, 3825, 4059, 4063, 4089
 \l__enumext_leftmargin_tmp_v_bool . 95, 3370
 \l__enumext_leftmargin_tmp_X_bool 68
 \l__enumext_leftmargin_tmp_X_dim 68
 \l__enumext_leftmargin_X_dim 68
 __enumext_level: 197, 197, 561, 564, 565, 574, 576, 815, 819, 823, 890, 894, 898, 902, 985, 987, 989, 991, 1024, 1026, 1028, 1030, 1034, 1067, 1070, 1089, 1098, 1104, 1109, 1113, 1124, 1128, 1129, 1134, 1170, 1174, 1347, 1353, 1400, 1402, 1404, 1407, 1414, 1416, 1418, 1421, 2030, 2032, 2034, 2062, 2063, 2065, 2121, 2129, 2133, 2137, 2399, 2402, 2403, 2780, 2781, 2785, 2786, 2787, 2794, 2796, 2800, 2801, 2804, 2810, 2811, 2866, 2869, 2871, 2878, 2879, 2880, 2883, 2886, 3016, 3018, 3066, 3079, 3086, 3097, 3099, 3102, 3103, 3105, 3110, 3117, 3120, 3122, 3124, 3125, 3126, 3127, 3130, 3136, 3141, 3147, 3150, 3152, 3158
 \l__enumext_level_h_int .. 28, 229, 251, 264, 600, 637, 1836, 1852, 2355, 2372, 3552, 3553
 \l__enumext_level_int . 87, 28, 199, 238, 250, 265, 355, 997, 1141, 1334, 1830, 1858, 1938, 2332, 2342, 2348, 2354, 2362, 2370, 2377, 2970, 3032, 3033, 3043, 3050, 3064, 3077, 3132, 3210, 3297, 3590, 3600, 3915
 __enumext_list_arg_two_i: 2936
 __enumext_list_arg_two_ii: 2936
 __enumext_list_arg_two_iii: 2936
 __enumext_list_arg_two_iv: 2936
 __enumext_list_arg_two_v: . 83, 2936, 3185, 3371
 __enumext_list_arg_two_vii: 2976, 3533
 __enumext_list_arg_two_viii: 2976, 3890
 \l__enumext_listoffset_v_dim 3248
 \l__enumext_listparindent_vii_dim 3725
 \l__enumext_listparindent_viii_dim ... 4069
 __enumext_log_answer_vars: . 32, 327, 335, 1945
 __enumext_log_global_vars: . 32, 327, 327, 1944
 __enumext_make_label: 36, 81, 82, 84, 2874, 2874, 2965
 \l__enumext_mark_answer_sym_tl . 70, 131, 1987, 2191, 2414, 2692, 2705, 4016
 \l__enumext_mark_position_str 131, 1991, 1992, 2015, 2016, 2189
 \l__enumext_mark_ref_sym_tl .. 131, 2001, 2312, 2644
 __enumext_mini_addvspace: .. 50, 89, 1163, 1163, 3107

```

\__enumext_mini_addvspace_vii: 52, 1316, 1316,
    3491
\__enumext_mini_addvspace_viii: 52, 1316, 1322,
    3850
__enumext_mini_env* ..... 353
\__enumext_mini_right_cmd:n 53, 1342, 1344, 1344
\__enumext_mini_set_vskip: . 48, 1064, 1064, 1165
\__enumext_mini_set_vskip_vii: 51, 1259, 1259,
    1318
\__enumext_mini_set_vskip_viii: 51, 1259, 1281,
    1324
\__enumext_minipage:w 32, 348, 350, 359, 3402, 3724,
    4068
\l__enumext_minipage_active_v_bool . . 92, 93,
    3229, 3254, 3267, 3275
\g__enumext_minipage_active_vii_bool . . . 98,
    3502, 3507, 3519
\l__enumext_minipage_active_vii_bool . 3487,
    3498
\g__enumext_minipage_active_viii_bool 3861,
    3866, 3878
\l__enumext_minipage_active_viii_bool 3846,
    3857
\g__enumext_minipage_active_X_bool . . . 164
\l__enumext_minipage_active_X_bool . . . . 82
\g__enumext_minipage_after_skip 82, 1263, 1275,
    3517, 3876
\l__enumext_minipage_after_skip 48, 49, 90, 93,
    82, 1080, 1095, 1115, 1131, 1146, 1152, 1158, 1172,
    1182, 1191, 1194, 1206, 1224, 1235, 1251, 1283, 1296,
    1310, 3167, 3284
\g__enumext_minipage_center_vii_bool . 3511,
    3520
\g__enumext_minipage_center_viii_bool 3870,
    3879
\g__enumext_minipage_center_X_bool . . . 164
\l__enumext_minipage_hsep_v_dim . . . 92, 3227
\l__enumext_minipage_hsep_vii_dim . . . . 3485
\l__enumext_minipage_hsep_viii_dim . . . 3844
\l__enumext_minipage_left_skip 48, 92, 82, 1072,
    1087, 1106, 1121, 1168, 1178, 1183, 1189, 1198, 1215,
    1227, 1247, 1257, 1261, 1266, 1270, 1284, 1288, 1302,
    1320, 1326
\l__enumext_minipage_left_v_dim 92, 3225, 3233
\l__enumext_minipage_left_vii_dim 3481, 3493
\l__enumext_minipage_left_viii_dim 3840, 3852
\l__enumext_minipage_left_X_dim . . . . . 82
\g__enumext_minipage_right_skip 82, 1262, 1267,
    1271, 3510, 3869
\l__enumext_minipage_right_skip . 48, 82, 1076,
    1091, 1111, 1126, 1184, 1190, 1202, 1220, 1231, 1285,
    1292, 1306, 1354, 1371
\l__enumext_minipage_right_v_dim . . 92, 1365,
    1370, 3223, 3227
\g__enumext_minipage_right_vii_dim 98, 3489,
    3509, 3522
\l__enumext_minipage_right_vii_dim 98, 3479,
    3484, 3490
\g__enumext_minipage_right_viii_dim . . 3848,
    3868, 3881
\l__enumext_minipage_right_viii_dim . . 3838,
    3843, 3849
\g__enumext_minipage_right_X_dim . . . . . 164
\g__enumext_minipage_right_X_skip . . . . 164
\g__enumext_minipage_stat_int . 89, 92, 82, 1359,
    1376, 3106, 3160, 3165, 3230, 3277, 3282
\g__enumext_miniright_code_vii_tl . 98, 3515,
    3521
\g__enumext_miniright_code_viii_tl 3874, 3880
\g__enumext_miniright_code_X_tl . . . . . 164
\__enumext_multi_addvspace: . 47, 90, 1019, 1019,
    3138
\__enumext_multi_set_vskip: . 46, 983, 983, 1021
\l__enumext_multicols_above_ii_skip . . . 1002
\l__enumext_multicols_above_iii_skip . . 1008
\l__enumext_multicols_above_iv_skip . . . 1014
\l__enumext_multicols_above_v_skip 1038, 1052,
    1062
\l__enumext_multicols_above_X_skip . . . . 76
\l__enumext_multicols_below_v_skip 1042, 1056,
    3269
\l__enumext_multicols_below_X_skip . . . . 76
\__enumext_multicols_start: 89, 3112, 3114, 3114
\__enumext_multicols_stop: 90, 1349, 3144, 3144,
    3169
\__enumext_newlabel:nn 28, 34, 74, 404, 404, 2388,
    2603
\l__enumext_newlabel_arg_one_tl 28, 34, 74, 78,
    153, 2311, 2381, 2389, 2592, 2604, 2642
\l__enumext_newlabel_arg_two_tl 28, 34, 73, 153,
    2335, 2345, 2359, 2375, 2390, 2579, 2584, 2589, 2605
\__enumext_parse_keys:n . . . 57, 3012, 3038, 3038
\__enumext_parse_keys_vii:n 57, 3528, 3560, 3560
\__enumext_parse_keys_viii:n . 3886, 3920, 3920
\__enumext_parse_save_key:n 68, 2055, 2060, 2060
\__enumext_parse_save_key_vii:n 68, 2050, 2060,
    2068
\__enumext_parse_serie:n . . . . . 99
\__enumext_parse_series:n . . 57, 88, 1533, 1533,
    3046, 3566
\__enumext_parse_store_keys:n . . . . . 88
\l__enumext_parsep_i_skip 1000, 1002, 1144, 1192
\l__enumext_parsep_ii_skip . . . 1006, 1008, 1150
\l__enumext_parsep_iii_skip . . 1012, 1014, 1156
\l__enumext_parsep_vii_skip . . . . . 3726
\l__enumext_parsep_viii_skip . . . . . 4070
\l__enumext_partopsep_v_skip . 1054, 1058, 1218,
    1222, 1229, 1233, 1249, 1253
\l__enumext_partopsep_viii_skip . . . . . 1294
\__enumext_phantomsection: 33, 368, 397, 401, 417
\__enumext_print_footnote: . . . 2739, 2762, 3744,
    4096
\__enumext_print_keyans_box:NN 70, 2183, 2183,
    2196, 2401, 2684, 2719, 4012, 4027
\l__enumext_print_keyans_i_tl . . . . 4157, 4179
\l__enumext_print_keyans_ii_tl . . . 4161, 4180
\l__enumext_print_keyans_iii_tl . . 4165, 4181
\l__enumext_print_keyans_iv_tl . . . 4169, 4182
\l__enumext_print_keyans_starred_tl 111, 112,
    121, 4153, 4201
\l__enumext_print_keyans_vii_tl 111, 4173, 4183
\l__enumext_print_keyans_X_tl . . . . . 121
\__enumext_printkeyans:nnn 112, 4184, 4187, 4187
\__enumext_redefine_item: . 82, 2813, 2813, 2964
\l__enumext_ref_key_arg_tl 37, 46, 212, 554, 555,
    568, 599, 602, 613, 619, 630, 671, 672, 683
\l__enumext_ref_the_count_tl . 37, 46, 561, 564,
    567, 607, 609, 612, 624, 626, 629, 677, 679, 682
\__enumext_regex_counter_style: . . 29, 37, 207,

```


207, 562, 608, 625, 678
 __enumext_register_counter_style:Nn .. 436,
 436, 441, 442, 443, 444, 445
 __enumext_remove_extra_parsep_vii: .. 3543,
 3753, 3753
 __enumext_remove_extra_parsep_viii: . 3900,
 4106, 4106
 __enumext_renew_footnote: ... 2739, 2743, 3696,
 4049
 \l__enumext_renew_the_count_v_tl 680, 689, 691
 \l__enumext_renew_the_count_vii_tl 610, 639,
 641
 \l__enumext_renew_the_count_viii_tl 627, 646,
 648
 \l__enumext_renew_the_count_X_tl 46
 __enumext_reset_global_bool: .. 303, 306, 315
 __enumext_reset_global_int: ... 303, 305, 309
 __enumext_reset_global_tl: 303, 307, 321
 __enumext_reset_global_vars: . 31, 65, 303, 303,
 1954
 \l__enumext_resume_active_bool 57, 59, 57, 1537,
 1657
 __enumext_resume_counter: .. 58, 59, 1655, 1661,
 1668
 __enumext_resume_counter:n . 57, 59, 1626, 1631,
 1655, 1655, 1725, 1733
 __enumext_resume_counter_save_ans: .. 59, 60,
 1655, 1666, 1698
 __enumext_resume_counter_series: 59, 60, 1655,
 1664, 1681
 \g__enumext_resume_int ... 57, 1578, 1672, 1673
 __enumext_resume_last:n .. 57, 1533, 1539, 1552
 \l__enumext_resume_name_tl 57, 1574, 1582, 1585,
 1601, 1609, 1612, 1658, 1659, 1687, 1694
 __enumext_resume_save_counter: 57, 1565, 1565,
 3174, 3584
 __enumext_resume_series:n . 58, 1497, 1622, 1622
 __enumext_resume_starred: . 60, 1498, 1719, 1719
 \g__enumext_resume_vii_int . 100, 57, 1605, 1677,
 1678
 __enumext_safe_exec: 33, 75, 87, 3011, 3028, 3028
 __enumext_safe_exec_vii: 33, 75, 3527, 3548, 3548
 __enumext_safe_exec_viii: ... 3885, 3905, 3905
 __enumext_scontents_anskey:n. 75, 1782, 2428,
 2428, 2448
 __enumext_scontents_anskey_clean_
 vars: 2449, 2527
 __enumext_scontents_anskey_exec: 2446, 2449,
 2449
 __enumext_scontents_anskey_keys: . 76, 2449,
 2461, 2467
 __enumext_scontents_anskey_store: 2449, 2463,
 2509
 \l__enumext_series_name_tl 59
 \l__enumext_series_str . 58, 88, 1495, 1535, 1543,
 1544, 1546, 1548, 1569, 1572, 1576, 1596, 1599, 1603,
 3042, 3564
 __enumext_set_error:nn 4273, 4283, 4285
 __enumext_set_parse:n 4256, 4273, 4273
 \l__enumext_setkey_tmpa_int ... 116, 4249, 4253
 \l__enumext_setkey_tmpa_seq .. 116, 4247, 4257,
 4263, 4265, 4267, 4280
 \l__enumext_setkey_tmpa_tl ... 116, 4255, 4259
 \l__enumext_setkey_tmpb_seq .. 116, 4248, 4251,
 4255, 4256
 \l__enumext_setkey_tmpb_tl 116, 4275, 4277, 4278
 \l__enumext_show_answer_bool . 131, 1995, 2019,
 2408, 2659, 2673, 3321, 4010
 __enumext_show_length:nnn .. 43, 215, 215, 4349,
 4350, 4351, 4352, 4353, 4354, 4355, 4356, 4357, 4358,
 4364, 4365, 4366, 4367, 4368, 4369, 4370, 4371, 4372,
 4373
 \l__enumext_show_position_bool 131, 1998, 2022,
 2412, 2663, 2674, 3322, 4014
 \g__enumext_standar_bool 30, 87, 34, 228, 231, 249,
 318, 1567, 1632, 1644, 1670, 1683, 1721, 1857, 1870
 \l__enumext_standar_bool . 87, 91, 34, 2340, 2353,
 2369, 3035, 3173
 \l__enumext_standar_first_bool 30, 87, 34, 254,
 1554, 1701, 1763, 1770
 __enumext_standar_item_vii:w . 101, 3624, 3626,
 3626
 __enumext_standar_item_viii:w 107, 3957, 3959,
 3959
 __enumext_standar_ref: 37, 552, 572, 2966
 __enumext_standar_ref:n 37, 544, 552, 552
 \g__enumext_standar_series_tl . 57, 1556, 1557,
 1723, 1726
 \g__enumext_starred_bool 30, 99, 100, 34, 237, 240,
 263, 319, 1594, 1637, 1648, 1675, 1690, 1729, 1835,
 1876, 2331, 2341, 2371, 2573, 3061, 3074, 3523
 \l__enumext_starred_bool 99, 100, 34, 2266, 2272,
 2356, 2397, 2471, 2476, 3557, 3583
 __enumext_starred_columns_set_vii: .. 3409,
 3409, 3536
 __enumext_starred_columns_set_viii: . 3768,
 3768, 3893
 \l__enumext_starred_first_bool ... 30, 34, 268,
 1559, 1710, 1763, 1770
 __enumext_starred_item:nn ... 2790, 2790, 2819
 __enumext_starred_item_exec: . 108, 4002, 4002,
 4053
 __enumext_starred_item_vii:w . 101, 3623, 3642,
 3642
 __enumext_starred_item_vii_aux_i:w .. 3642,
 3647, 3650
 __enumext_starred_item_vii_aux_ii:w . 3642,
 3648, 3653, 3655
 __enumext_starred_item_vii_aux_iii:w 3642,
 3658, 3667
 __enumext_starred_item_viii:w 107, 108, 3956,
 3975, 3975
 __enumext_starred_item_viii_aux_i:w .. 108,
 3975, 3980, 3983
 __enumext_starred_item_viii_aux_ii:w . 108,
 3975, 3981, 3995, 3997
 __enumext_starred_joined_item_vii:n 97, 101,
 3428, 3428, 3621
 __enumext_starred_joined_item_viii:n . 104,
 107, 3787, 3787, 3954
 __enumext_starred_ref: 38, 597, 635, 2994
 __enumext_starred_ref:n 38, 591, 597, 597
 \g__enumext_starred_series_tl . 57, 1561, 1562,
 1731, 1734
 __enumext_start_from:NNn 40, 694, 694, 707, 729
 \l__enumext_start_i_int 1673, 1685, 1704
 __enumext_start_item_tmp_vii: 99, 3539, 3606,
 3606

`__enumext_start_item_tmp_viii:` .. 106, 3896, 3939, 3939
`__enumext_start_item_vii:w` 101, 102, 3634, 3639, 3664, 3671, 3673, 3673
`__enumext_start_item_viii:w` .. 107, 3967, 3972, 4000, 4030, 4030
`\g__enumext_start_line_tl` 30, 141, 256, 270, 324, 1900, 1905, 1910, 1924, 1929, 1934
`__enumext_start_list:nn` 32, 85, 95, 342, 344, 3015, 3182, 3335, 3531, 3888
`__enumext_start_mini_vii:` 100, 3477, 3477, 3575
`__enumext_start_mini_viii:` ... 106, 3836, 3836, 3931
`__enumext_start_save_ans_msg:` 61, 1747, 1747, 1772
`__enumext_start_store_level:` . 88, 3014, 3055, 3055
`__enumext_start_store_level_vii:` 100, 3530, 3586, 3586
`\l__enumext_start_vii_int` ... 1678, 1692, 1713
`\l__enumext_start_X_int` 94, 724
`__enumext_stop_item_tmp_vii:` 99, 100, 102, 3538, 3542, 3608, 3675
`__enumext_stop_item_tmp_viii:` 106, 107, 3895, 3899, 3941, 4032
`__enumext_stop_item_vii:` 102, 103, 3675, 3729, 3729
`__enumext_stop_item_viii:` 110, 4032, 4081, 4081
`__enumext_stop_list:` .. 32, 342, 345, 3024, 3192, 3348, 3544, 3902
`__enumext_stop_mini_vii:` . 98, 100, 3496, 3496, 3579
`__enumext_stop_mini_viii:` 107, 3836, 3855, 3935
`__enumext_stop_save_ans_msg:` . 61, 1747, 1752, 1942
`__enumext_stop_store_level:` .. 88, 3025, 3055, 3084
`__enumext_stop_store_level_vii:` . 100, 3545, 3586, 3596
`\l__enumext_store_active_bool` 27, 62, 88, 99, 106, 1702, 1711, 1779, 2226, 3059, 3072, 3197, 3205, 3293, 3352, 3588, 3598, 3914
`__enumext_store_active_keys:n` 67, 2028, 2028, 3052
`__enumext_store_active_keys_vii:n` .. 67, 99, 2028, 2038, 3567
`__enumext_store_addto_prop:n` 69, 77, 2103, 2103, 2111, 2253, 2554, 4005
`__enumext_store_addto_seq:n` 69, 79, 2112, 2112, 2116, 2123, 2137, 2145, 2154, 2172, 2180, 2315, 2647
`\l__enumext_store_anskey_arg_tl` .. 27, 72, 106, 2263, 2268, 2270, 2275, 2282, 2285, 2295, 2300, 2303, 2309, 2315, 2453, 2456, 2459, 2512, 2517, 2522
`__enumext_store_anskey_code:nn` . 71, 72, 2220, 2251, 2251
`\l__enumext_store_anskey_opt_tl` 76, 106, 2454, 2473, 2479, 2486, 2492, 2502, 2514, 2520
`__enumext_store_anskey_safe_outer:` 71
`__enumext_store_anskey_show_left:n` 75, 2258, 2406, 2406
`__enumext_store_anskey_show_wrap:n` 74, 2394, 2394, 2410, 2425
`\g__enumext_store_columns_break_bool` . 2432, 2470, 2529
`\l__enumext_store_columns_break_bool` . 2199, 2265
`\l__enumext_store_columns_join_int` ... 106
`__enumext_store_internal_ref:` .. 72, 73, 2256, 2317, 2317
`\g__enumext_store_item_join_int` .. 2435, 2477, 2481, 2530
`\l__enumext_store_item_join_int` .. 2202, 2273, 2277
`\g__enumext_store_item_star_bool` . 2437, 2484, 2531
`\l__enumext_store_item_star_bool` . 2204, 2280
`\g__enumext_store_item_symbol_sep_dim` 2442, 2499, 2504, 2533
`\l__enumext_store_item_symbol_sep_dim` 2209, 2292, 2297
`\g__enumext_store_item_symbol_tl` . 2440, 2490, 2494, 2532
`\l__enumext_store_item_symbol_tl` . 2207, 2283, 2287
`\l__enumext_store_keyans_item_opt_sep_tl` 1984, 2548, 2550, 2621, 2625, 3988, 3990
`\l__enumext_store_keyans_item_opt_tl` .. 106
`\l__enumext_store_keyans_label_tl` 27, 77-79, 108, 106, 2537, 2540, 2543, 2550, 2552, 2554, 2611, 2614, 2617, 2623, 2628, 2638, 2647, 3985, 3990, 3991, 4004, 4005, 4007
`__enumext_store_level_close:` . 69, 2117, 2141, 3088
`__enumext_store_level_close_vii:` 2148, 2176, 3602
`__enumext_store_level_open:` .. 69, 2117, 2117, 3067, 3080
`__enumext_store_level_open_vii:` . 2148, 2148, 3592
`\g__enumext_store_name_tl` 27, 90, 106, 323, 330, 331, 332, 333, 1755, 1781, 1899, 1904, 1909, 1923, 1928, 1933, 1940, 2456, 2457
`\l__enumext_store_name_tl` 27, 61, 63, 106, 1588, 1591, 1615, 1618, 1706, 1715, 1750, 1759, 1760, 1781, 1782, 1783, 1785, 1786, 1788, 1790, 1791, 1793, 1795, 1796, 1820, 2105, 2107, 2114, 2383, 2384, 2420, 2594, 2595, 2698, 2711, 4022
`\l__enumext_store_ref_key_bool` 72, 2004, 2254, 2306, 2558, 2635
`\l__enumext_store_save_key_vii_bool` .. 2040, 2070
`\l__enumext_store_save_key_vii_tl` 2042, 2043, 2071, 2072, 2152, 2162, 2168, 2172
`\l__enumext_store_save_key_X_bool` 67
`\l__enumext_store_save_key_X_tl` 67, 121
`\l__enumext_store_upper_level_X_bool` .. 121
`\l__enumext_store_write_aux_file_tl` 28, 74, 78, 153, 2386, 2392, 2601, 2607
`__enumext_storing_exec:` 61, 62, 1757, 1773, 1777
`__enumext_storing_set:n` .. 61, 1742, 1757, 1757
`\l__enumext_the_counter_v_tl` 679
`\l__enumext_the_counter_vii_tl` 609
`\l__enumext_the_counter_viii_tl` 626
`\l__enumext_the_counter_X_tl` 46
`__enumext_tmp:n` 41, 45, 50, 56, 68, 75, 76, 81, 88, 93, 94, 105, 122, 130, 156, 160, 164, 183, 807, 811, 1491, 1502, 1738, 1746, 1799, 1817, 1974, 2009, 2010, 2027, 2046, 2059, 2319, 2326, 2327, 2348, 2362, 2365, 2377, 2560, 2567, 2936, 2975, 2976, 3008

<code>__enumext_tmp:nn</code>	468, 489, 490, 518, 519, 531, 724, 743, 788, 806, 864, 872, 873, 887, 952, 968, 969, 982, 1380, 1396, 2723, 2738
<code>__enumext_tmp:nnn</code>	532, 548, 549, 550, 551, 579, 595, 596
<code>__enumext_tmp:nnnnn</code>	744, 769, 772, 775, 777, 779, 782, 785
<code>__enumext_tmp:w</code>	4133, 4136
<code>\l__enumext_tmpa_vii_int</code>	3419, 3422
<code>\l__enumext_tmpa_viii_int</code>	3778, 3781
<code>\l__enumext_tmpa_X_int</code>	164
<code>\l__enumext_topsep_v_skip</code>	1040, 1044, 1187, 1200, 1208, 1213, 1233, 1237, 3351, 3383
<code>\l__enumext_topsep_vii_skip</code>	1264, 1273, 1277
<code>\l__enumext_topsep_viii_skip</code>	1286, 1308, 1312
<code>\l__enumext_vspace_a_star_v_bool</code>	1429
<code>\l__enumext_vspace_a_star_vii_bool</code>	1451
<code>\l__enumext_vspace_a_star_viii_bool</code>	1462
<code>\l__enumext_vspace_a_star_X_bool</code>	94
<code>__enumext_vspace_above:</code>	54, 1397, 1397, 3093
<code>__enumext_vspace_above_v:</code>	55, 1425, 1425, 3221
<code>\l__enumext_vspace_above_v_skip</code>	1427, 1431, 1433
<code>__enumext_vspace_above_vii:</code>	55, 1447, 1447, 3572
<code>\l__enumext_vspace_above_vii_skip</code>	1449, 1453, 1455
<code>__enumext_vspace_above_viii:</code>	55, 1447, 1458, 3929
<code>\l__enumext_vspace_above_viii_skip</code>	1460, 1464, 1466
<code>\l__enumext_vspace_b_star_v_bool</code>	1440
<code>\l__enumext_vspace_b_star_vii_bool</code>	1473
<code>\l__enumext_vspace_b_star_viii_bool</code>	1484
<code>\l__enumext_vspace_b_star_X_bool</code>	94
<code>__enumext_vspace_below:</code>	54, 1411, 1411, 3172
<code>__enumext_vspace_below_v:</code>	55, 1436, 1436, 3289
<code>\l__enumext_vspace_below_v_skip</code>	1438, 1442, 1444
<code>__enumext_vspace_below_vii:</code>	55, 1469, 1469, 3582
<code>\l__enumext_vspace_below_vii_skip</code>	1471, 1475, 1477
<code>__enumext_vspace_below_viii:</code>	55, 1469, 1480, 3937
<code>\l__enumext_vspace_below_viii_skip</code>	1482, 1486, 1488
<code>__enumext_widest_from:nNNn</code>	40, 708, 708, 723, 735
<code>\g__enumext_widest_label_tl</code>	26, 35, 64, 456, 460, 464
<code>\l__enumext_wrap_label_opt_v_bool</code>	2832
<code>\l__enumext_wrap_label_opt_vii_bool</code>	101, 3633
<code>\l__enumext_wrap_label_opt_viii_bool</code>	107, 3966
<code>\l__enumext_wrap_label_opt_X_bool</code>	94
<code>\l__enumext_wrap_label_v_bool</code>	2828, 2832, 2840, 2896
<code>\l__enumext_wrap_label_vii_bool</code>	101, 3632, 3637, 3645, 3713
<code>\l__enumext_wrap_label_viii_bool</code>	107, 3965, 3970, 3978, 4057
<code>\l__enumext_wrap_label_X_bool</code>	94
<code>__enumext_wrapper_label_v:n</code>	2898, 3330
<code>__enumext_wrapper_label_vii:n</code>	3716
<code>__enumext_wrapper_label_viii:n</code>	4060
<code>__enumext_zero_parsep:</code>	49, 1084, 1139, 1139
<code>enumext*</code>	5, 3525
<code>enumXi</code>	428
<code>enumXii</code>	428
<code>enumXiii</code>	428
<code>enumXiv</code>	428
<code>enumXv</code>	428
<code>enumXvi</code>	428
<code>enumXvii</code>	428
<code>enumXviii</code>	428
Environments provide by enumext :	
<code>anskey*</code>	75
<code>enumext*</code>	24, 25, 27–30, 33, 34, 37, 38, 40–45, 51, 52, 55–58, 60–63, 65–76, 78, 80, 81, 87, 88, 99, 100, 102, 104, 105, 107, 109, 111, 112, 115, 117
<code>enumext</code>	24, 25, 27, 29, 30, 33–41, 43–51, 53, 54, 56–58, 60–63, 65–76, 78, 80–82, 84, 85, 87, 88, 91, 95, 96, 98, 100, 111, 112, 115, 116
<code>keyans*</code>	24, 25, 27–31, 34, 37–45, 51, 52, 55, 62, 66, 69, 77, 81, 87, 106, 115, 117
<code>keyanspic</code>	24, 25, 27, 28, 31, 34, 36, 39, 53, 62, 66, 69, 77–79, 93–96, 117
<code>keyans</code>	24, 25, 27, 28, 30, 31, 34–36, 39–41, 43–45, 47, 50, 51, 53–55, 62, 66, 69, 77–79, 83–85, 91–95, 98, 107, 115, 117
Environments:	
<code>list</code>	29, 32, 85, 87
<code>lrbox</code>	96, 102, 103, 109, 110
<code>minipage</code>	29, 32, 33, 45, 47, 48, 75, 93, 95, 96, 102, 103, 110
<code>multicols</code>	46–48, 53, 89, 90, 92, 93
<code>scontents</code>	75
exp commands:	
<code>\exp_after:wN</code>	4136
<code>\exp_args:Ne</code>	2516, 2521, 3049, 4124
<code>\exp_not:N</code>	54, 459, 567, 612, 629, 682, 821, 835, 836, 847, 848, 859, 860, 2311, 2417, 2418, 2640, 2695, 2696, 2708, 2709, 4019, 4020, 4133
<code>\exp_not:n</code>	258, 272, 284, 291, 298, 567, 568, 612, 613, 629, 630, 682, 683, 822, 1519, 1531, 2089, 2101, 2277, 2287, 2297, 2311, 2312, 2389, 2481, 2494, 2504, 2604, 2642, 2644
F	
<code>\fbox</code>	1979
file commands:	
<code>\file_input_stop:</code>	4518
<code>first</code>	873
<code>font</code>	468
<code>\footnote</code>	81
<code>\footnote</code>	81, 2747
<code>\footnotemark</code>	2757
<code>\footnotesize</code>	2418, 2696, 2709, 4020
<code>\footnotetext</code>	2741
G	
<code>\getkeyans</code>	15, 110, 4122
group commands:	
<code>\group_begin:</code>	2215, 2416, 2511, 2694, 2707, 3692, 3711, 4018, 4045, 4055, 4144, 4178
<code>\group_end:</code>	2222, 2423, 2525, 2701, 2714, 3721, 3733, 4025, 4065, 4085, 4146, 4185
H	
<code>\hbadness</code>	3740, 4092

hbox commands:

<code>\hbox_set:Nn</code>	448
---------------------------	-----

`\hfill` 498, 502, 507, 508, 1351, 1369, 2311, 2640, 3501, 3860

hook commands:

<code>\hook_gput_code:nnn</code>	9, 191, 195, 366
<code>\hook_gset_rule:nnnn</code>	367

`\hspace` 3751, 4104

`\hyperlink` 73, 76, 77, 79

`\hyperlink` 2311, 2640

`\hypertarget` 33

`\hypertarget` 396

I

<code>\IfHyperBoolean</code>	374
<code>\IfPackageLoadedTF</code>	11, 19, 370, 384
<code>\ignorespaces</code>	824
<code>\inputlineno</code>	258, 272, 284, 291, 298

int commands:

<code>\int_add:Nn</code>	3461, 3820
<code>\int_case:nn</code>	997, 1141, 1830, 1852, 1890, 1914
<code>\int_compare:nNnTF</code>	355, 600, 617, 637, 644, 1066, 1185, 1330, 1334, 1338, 1938, 1959, 1965, 2230, 2234, 2246, 2538, 2577, 2582, 2587, 2612, 2690, 3033, 3043, 3064, 3077, 3116, 3132, 3146, 3160, 3206, 3210, 3239, 3264, 3277, 3297, 3301, 3357, 3431, 3441, 3457, 3553, 3590, 3600, 3746, 3755, 3790, 3800, 3816, 3908, 3915, 4098, 4108, 4253
<code>\int_compare_p:nNn</code>	229, 238, 250, 251, 264, 265, 1836, 1858, 2273, 2332, 2342, 2354, 2355, 2370, 2372, 2477
<code>\int_decr:N</code>	3460, 3819
<code>\int_eval:n</code>	340, 2107, 2384, 2418, 2595, 2696, 2709, 2951, 2993, 3449, 3808, 4020
<code>\int_from_alph:n</code>	702, 716
<code>\int_from_roman:n</code>	704, 718
<code>\int_gadd:Nn</code>	3462, 3821
<code>\int_gdecr:N</code>	1839, 1843, 1846, 1849, 1861
<code>\int_gincr:N</code>	1672, 1677, 2218, 2650, 2778, 2808, 2846, 3106, 3230, 3319, 3610, 3688, 3943, 4009
<code>\int_gset:Nn</code>	1883, 2755
<code>\int_gset_eq:NN</code>	1571, 1578, 1584, 1590, 1598, 1605, 1611, 1617, 2752
<code>\int_gzero:N</code>	311, 312, 313, 1359, 1376, 1971, 2530, 3165, 3282, 3764, 4119
<code>\int_if_exist:N</code>	1546, 1582, 1588, 1609, 1615, 1793
<code>\int_incr:N</code>	2245, 3032, 3201, 3356, 3552, 3609, 3907, 3942
<code>\int_mod:nn</code>	3757, 4110
<code>\int_new:N</code>	28, 29, 30, 31, 32, 33, 57, 58, 82, 98, 111, 118, 138, 139, 147, 148, 149, 150, 161, 167, 168, 169, 170, 171, 1548, 1796
<code>\int_set:Nn</code>	698, 702, 704, 1685, 1692, 1704, 1713, 3396, 3397, 3419, 3430, 3436, 3452, 3740, 3778, 3789, 3795, 3811, 4092, 4249
<code>\int_set_eq:NN</code>	1673, 1678, 3459, 3818
<code>\int_sign:n</code>	1885
<code>\int_step_function:nnN</code>	2348, 2362, 2377
<code>\int_step_inline:nnn</code>	3398
<code>\int_to_roman:n</code>	199, 2328, 2366
<code>\int_use:N</code>	333, 338, 339, 1067, 1687, 1694, 1706, 1715, 2951, 2970, 2993, 3050, 3117, 3126, 3141, 3147, 3434, 3435, 3447, 3793, 3794, 3806
<code>\int_zero:N</code>	3749, 4102
<code>\c_one_int</code>	3419, 3438, 3444, 3450, 3454, 3457, 3778, 3797, 3803, 3809, 3813, 3816
<code>\c_zero_int</code>	2332, 2342, 2354, 2355, 2370, 2372, 3590, 3600, 3760, 4115

<code>\item</code>	32, 44, 45, 70, 81, 93, 94, 96, 99, 106
<code>\item</code>	81, 83, 100, 102, 107, 109, 346, 2125, 2131, 2156, 2164, 2270, 2614, 2617, 2815, 2850, 3537, 3539, 3894, 3896, 4007
<code>\item*</code>	5, 14, 66, 2848
<code>item-pos*</code>	2723
<code>item-sym*</code>	2723
<code>\itemindent</code>	26, 86
<code>\itemindent</code>	85
<code>itemindent</code>	788
<code>\itemsep</code>	95, 96
<code>\itemsep</code>	3372, 3378
<code>\itemwidth</code>	3426, 3470, 3474, 3785, 3829, 3833

K

keyans	13, 3177
keyans*	13, 3883
keyanspic	14, 3332

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

<code>break-col</code>	71, 72, 76
<code>item-join</code>	71, 72, 76
<code>item-pos*</code>	71, 72, 76
<code>item-star</code>	71, 72, 76
<code>item-sym*</code>	71, 72, 76

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

<code>above*</code>	26, 54, 55
<code>above</code>	26, 54, 55, 89, 92, 100, 106
<code>after</code>	43-45, 91, 93, 100, 107
<code>align</code>	26, 36, 84, 102
<code>before*</code>	43, 44, 89, 100, 106
<code>before</code>	43, 44, 92
<code>below*</code>	26, 54, 55
<code>below</code>	26, 54, 55, 91, 93, 100, 107
<code>check-ans</code>	27-30, 61, 62, 64-66, 68, 79, 81, 82, 89-91, 104, 115
<code>columns-sep</code>	45, 89, 92
<code>columns</code>	26, 45, 48, 54, 89, 92
<code>first</code>	43-45, 102
<code>font</code>	35, 84, 102
<code>item-pos*</code>	80
<code>item-sym*</code>	25, 80, 82
<code>item*-sep</code>	82
<code>itemindent</code>	26, 41, 42, 83, 102
<code>itemsep</code>	41, 87
<code>labelsep</code>	35, 82, 86, 102
<code>labelwidth</code>	35, 37, 39, 40, 86
<code>label</code>	25, 26, 35, 36, 40, 96
<code>lisparindent</code>	87
<code>list-indent</code>	26, 41, 42, 95
<code>list-offset</code>	41, 42
<code>listparindent</code>	41, 102
<code>mark-ans</code>	27, 66, 68, 75
<code>mark-pos</code>	66
<code>mark-ref</code>	27, 66, 68, 73, 76, 77
<code>mini-env</code>	26, 33, 45, 53, 54, 68, 75, 81, 89, 92, 98, 100, 105, 106
<code>mini-right*</code>	26, 45, 68, 98
<code>mini-right</code>	26, 45, 52, 68, 98
<code>mini-sep</code>	26, 45, 68, 89, 92
<code>minirigth*</code>	29
<code>minirigth</code>	29
<code>no-store</code>	28, 61-63, 68, 71
<code>noitemsep</code>	41, 49
<code>nosep</code>	41, 49
<code>parindent</code>	87

parsep	41, 87, 102
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, 78, 83, 88, 91, 94, 100, 107, 108, 110, 111, 115
save-key	27, 56, 67, 88
save-pos	68
save-ref	28, 34, 66, 68, 72, 73, 76, 77, 79, 83, 108
save-sep	66, 68, 108
series	25, 56–60, 68, 88, 91
show-ans	27, 66, 68, 70, 72, 75, 83, 108
show-length	30, 43, 115
show-pos	27, 66, 70, 72, 75, 79, 83, 108
start	26, 29, 40, 56
store-key	67
topsep	41
widest	26, 29, 40
wrap-ans	66, 68, 70, 74
wrap-label*	35, 81, 84, 101, 102, 107
wrap-label	35, 84, 101, 102, 107
wrap-opt	66, 68
keys commands:	
\keys_define:nn	470, 492, 521, 534, 581, 652, 726, 746, 790, 809, 866, 875, 954, 971, 1382, 1493, 1740, 1801, 1976, 2012, 2048, 2053, 2197, 2430, 2725, 4149, 4218
\l_keys_key_str	4334
\keys_precompile:nnN	112, 4148, 4151, 4155, 4159, 4163, 4167, 4171
\keys_set:nn	484, 977, 1387, 1392, 1634, 1639, 1726, 1734, 2261, 2520, 3045, 3049, 3217, 3565, 3924, 4220, 4221, 4222, 4223, 4224, 4225, 4226, 4227, 4228, 4229, 4230, 4231, 4232, 4270
keyval commands:	
\keyval_parse:NNn	1507, 2078
L	
label	532, 579, 652
Labels provide by enumext:	
\Alph*	35
\Roman*	35
\alph*	35
\arabic*	29, 35
\roman*	35
\labelsep	96
\labelsep	3373, 3376
labelsep	468
\labelwidth	35, 96
\labelwidth	3373, 3374
labelwidth	468
\leftmargin	26, 86
\leftmargin	85, 3373
legacy commands:	
\legacy_if:nTF	3676, 3679, 4033, 4036
\legacy_if_gset_false:n	360
\legacy_if_set_false:n	3678, 4035
\legacy_if_set_true:n	3638, 3663, 3670, 3683, 3971, 3999, 4040
\linewidth	89, 92
\linewidth	3101, 3227, 3395, 3422, 3483, 3781, 3842
\list	32
\list	344
list-indent	788

list-offset	788
\listparindent	3375
listparindent	788
\lrbox	3693, 4046
M	
\makebox	96
\makebox	2187, 2189, 2870, 3707, 3715, 3719, 4059, 4063
\makelabel	81, 84, 96
\makelabel	84, 2876, 2892
\makesavenoteenv	390
mark-ans	1974
mark-pos	1974, 2010
mark-ref	1974
mini-env	952
mini-sep	952
\minipage	32
\minipage	350
\miniright	10, 52, 1328, 3163, 3280
\miniright*	10
mode commands:	
\mode_if_vertical:TF	1022, 1050, 1166, 1245
\mode_leave_vertical:	821, 835, 847, 859, 2156, 2164, 2185, 2868, 3705
msg commands:	
\msg_error:nn	2243, 2248, 3208, 3212, 3299, 3359, 3555, 3910, 3917, 4233
\msg_error:nnn	557, 604, 621, 674, 1332, 1336, 1361, 1378, 1646, 1650, 1765, 4138, 4143, 4215, 4286
\msg_error:nnnn	2228, 2232, 2236, 3199, 3295, 3303, 4198
\msg_fatal:nn	3034
\msg_fatal:nnn	422
\msg_info:nnn	13, 16, 21, 24, 372, 386
\msg_line_context:	4306, 4310, 4314, 4338, 4343, 4348, 4363, 4378, 4382, 4386, 4390, 4394, 4398, 4404, 4410, 4416, 4430, 4434, 4439, 4443, 4448, 4452, 4456, 4461, 4466, 4470, 4475, 4480, 4484, 4489, 4493, 4498, 4503, 4508, 4512, 4516
\msg_log:nnn	1785, 1790, 1795
\msg_log:nnnnn	337, 1923, 1928, 1933
\msg_log:nnnnnn	329
\msg_new:nnn	4287, 4291, 4295, 4299, 4304, 4308, 4312, 4316, 4322, 4328, 4332, 4336, 4341, 4346, 4361, 4376, 4380, 4384, 4388, 4392, 4396, 4400, 4407, 4413, 4419, 4423, 4427, 4432, 4437, 4441, 4446, 4450, 4454, 4459, 4464, 4468, 4473, 4478, 4482, 4487, 4491, 4496, 4501, 4506, 4510, 4514
\msg_term:nnnn	1749, 1754, 2960, 2970, 2999, 3004
\msg_term:nnnnn	1904
\msg_warning:nn	3162, 3279
\msg_warning:nnnn	1962, 1968, 2908, 2913, 3433, 3446, 3792, 3805
\msg_warning:nnnnn	1899, 1909
\multicolsep	89, 92
\multicolsep	3131, 3252
N	
\NeedsTeXFormat	3
\newcounter	425
\NewDocumentCommand	1328, 2212, 3291, 4122, 4176, 4240
\NewDocumentEnvironment	3009, 3177, 3332, 3525, 3883
\newenvsc	2445
\newlabel	34
\newlabel	408

no-store [1799](#)
\noindent [99, 106](#)
\noindent . [3108, 3232, 3492, 3538, 3748, 3851, 3895, 4101](#)
\nointerlineskip [3108, 3232, 3492, 3851](#)
noitemsep [744](#)
\nopagebreak [1033, 1061, 1177, 1256, 1319, 1325](#)
\normalfont [2417, 2695, 2708, 4019](#)
nosep [744](#)

P

Packages:
enumext [24, 36, 61, 85, 94, 114](#)
enumitem [34, 35](#)
expl3 [96](#)
footnotehyper [33](#)
hyperref [28, 29, 33, 34, 73, 76, 77, 79, 102, 114](#)
lua-visual-debug [48](#)
multicol [24, 114](#)
scontents [24](#)
shortlst [96](#)
\par .. [1033, 1061, 1177, 1256, 1319, 1325, 1354, 1371, 2396, 3152, 3167, 3269, 3284, 3407, 3510, 3517, 3748, 3762, 3869, 3876, 4101, 4117](#)
\parindent [3725, 4069](#)
\parsep [46, 49, 95, 96](#)
\parsep [2157, 2165, 2990, 3372, 3379, 3384](#)
parsep [744](#)
\parskip [3726, 4070](#)
\partopsep [96](#)
\partopsep [2991, 3377](#)
partopsep [744](#)
peek commands:
 \peek_meaning:NNTF [3615, 3629, 3646, 3657, 3948, 3962, 3979](#)
 \peek_meaning_remove:NNTF [3622, 3955](#)
 \peek_remove_spaces:n [2854](#)
\phantomsection [33](#)
\phantomsection [397](#)
prg commands:
 \prg_do_nothing: [401](#)
 \prg_new_protected_conditional:Npnn ... [201](#)
 \prg_replicate:nn [218](#)
 \prg_return_false: [205](#)
 \prg_return_true: [204](#)
\printkeyans [15, 111, 4176](#)
prop commands:
 \prop_count:N [331, 2107, 2384, 2420, 2595, 2698, 2711, 4022](#)
 \prop_gput_if_not_in:Nnn [2105](#)
 \prop_if_exist:NNTF [1783, 4142](#)
 \prop_item:Nn [4145](#)
 \prop_new:N [1786](#)
\ProvidesExplPackage [4](#)

R

\raggedcolumns [3140, 3258](#)
\ref [73, 77](#)
ref [532, 579, 652](#)
\refstepcounter [3685, 4042](#)
regex commands:
 \regex_match:nnTF [203, 701, 703, 715, 717](#)
 \regex_replace_all:nnN [2459](#)
 \regex_replace_once:nnN [211](#)
\renewcommand [567, 612, 629, 682](#)
\RenewDocumentCommand ... [2747, 2815, 2850, 2876, 2892](#)

\RequirePackage [17, 25](#)
resume [1491](#)
resume* [1491](#)
rightmargin [788](#)
\Roman [35, 40](#)
\Roman [444](#)
\roman [35, 40](#)
\roman [445, 550, 4166](#)

S

\s [2459](#)
save-ans [1738](#)
save-key [2046](#)
save-ref [1974](#)
save-sep [1974](#)
scan commands:
 \scan_stop: [96, 3386, 3537, 3894, 4133, 4136](#)
scontents internal commands:
 __scontents_rescan_tokens:n [2517, 2522](#)
seq commands:
 \seq_clear:N [4247](#)
 \seq_const_from_clist:Nn [4235](#)
 \seq_count:N [332, 3345, 4251](#)
 \seq_gclear:N [2745, 2746](#)
 \seq_gpop_right:NNTF [2455](#)
 \seq_gput_right:Nn [2114, 2758, 2759](#)
 \seq_if_empty:NNTF [2764, 4191, 4265](#)
 \seq_if_exist:NNTF [1788, 4189](#)
 \seq_if_in:NnTF [4196](#)
 \seq_item:Nn [2457, 3404](#)
 \seq_map_function:NN [4256](#)
 \seq_map_inline:Nn .. [4203, 4209, 4244, 4266, 4267](#)
 \seq_map_pairwise_function:NNN [2766](#)
 \seq_new:N [119, 120, 136, 162, 163, 1791](#)
 \seq_pop_left:NN [4255](#)
 \seq_put_right:Nn [3305, 4263, 4280](#)
 \seq_set_from_clist:Nn [4248](#)
 \seq_set_map_e:NNn [4257](#)
 \seq_show:N [4193](#)
series [1491](#)
\setcounter [712, 716, 718, 2951, 2993, 3350](#)
\setenumext [6, 113, 4240](#)
\setlength [2158, 2166](#)
show-ans [1974, 2010](#)
show-length [864](#)
show-pos [2010](#)
skip commands:
 \skip_add:Nn [1002, 1008, 1014, 1024, 1028, 1052, 1056, 1146, 1152, 1158, 1168, 1172, 1194, 1247, 1251, 3372](#)
 \skip_eval:n [2157, 2165](#)
 \skip_gset:Nn [1267, 1271, 1275](#)
 \skip_gzero_new:N [1262, 1263](#)
 \skip_horizontal:N [836, 848, 860, 3708, 3722, 4066](#)
 \skip_horizontal:n ... [822, 2186, 2194, 2869, 2871, 3706, 4075](#)
 \skip_if_eq:nnTF [1000, 1006, 1012, 1069, 1103, 1144, 1150, 1156, 1187, 1192, 1213, 1264, 1286, 1399, 1413, 1427, 1438, 1449, 1460, 1471, 1482](#)
 \skip_new:N [78, 79, 83, 84, 85, 86, 87, 140, 181](#)
 \skip_set:Nn . [985, 989, 1038, 1042, 1072, 1076, 1080, 1087, 1091, 1095, 1106, 1111, 1115, 1121, 1126, 1131, 1189, 1190, 1191, 1198, 1202, 1206, 1215, 1220, 1224, 1227, 1231, 1235, 1266, 1270, 1288, 1292, 1296, 1302, 1306, 1310, 3366, 3380](#)

<code>\skip_set_eq:NN</code>	2949, 2989, 2990, 3725, 3726, 4069, 4070
<code>\skip_use:N</code>	.. 987, 991, 1026, 1030, 1034, 1054, 1058, 1070, 1089, 1098, 1104, 1109, 1113, 1124, 1128, 1129, 1134, 1170, 1174, 1200, 1400, 1404, 1407, 1414, 1418, 1421, 3152
<code>\skip_zero:N</code> 2991, 3131, 3252, 3377, 3378
<code>\skip_zero_new:N</code>	1182, 1183, 1184, 1261, 1283, 1284, 1285
<code>\c_zero_skip</code>	1000, 1006, 1012, 1070, 1104, 1144, 1150, 1156, 1187, 1192, 1213, 1264, 1286, 1400, 1414, 1427, 1438, 1449, 1460, 1471, 1482
<code>\small</code> 4154, 4158, 4162, 4166, 4170, 4174
<code>\star</code> 2729
<code>start</code> <u>724</u>
<code>\stepcounter</code> 2751, 3312
str commands:	
<code>\c_backslash_str</code>	4306, 4310, 4314, 4318, 4319, 4320, 4324, 4325, 4421, 4425, 4429, 4443, 4444, 4448, 4456, 4457, 4461, 4462, 4493, 4494, 4498, 4503, 4504
<code>\c_colon_str</code> 2383, 2594, 4133
<code>\c_left_brace_str</code>	.. 4386, 4390, 4403, 4409, 4415
<code>\c_right_brace_str</code>	.. 4386, 4390, 4403, 4409, 4415
<code>\str_case:nn</code> 223, 278
<code>\str_case:nnTF</code> 1514, 1523, 2085, 2093
<code>\str_clear:N</code> 3042, 3564
<code>\str_count:n</code> 218
<code>\str_if_empty:NTF</code> 1535, 1576, 1603
<code>\str_if_eq:nnTF</code> 2952, 2995
<code>\str_if_in:nnTF</code> 4129
<code>\str_new:N</code> 135, 176
<code>\str_set:Nn</code>	... 524, 525, 526, 1991, 1992, 2015, 2016
<code>\string</code> 390
<code>\strutbox</code>	.. 1074, 1078, 1082, 1093, 1097, 1108, 1117, 1123, 1133, 1146, 1152, 1158, 1189, 1190, 1191, 1194, 1204, 1208, 1217, 1224, 1229, 1237, 1266, 1267, 1270, 1277, 1290, 1298, 1304, 1312, 3382

T

T _E X and L ^A T _E X 2 _ε commands:	
<code>\@auxout</code> 406
<code>\@currentenv</code> 223, 278
<code>\protected@write</code> 406
text commands:	
<code>\text_expand:n</code> 4125
<code>\textasteriskcentered</code> 1988, 2002
<code>\thepage</code> 412
tl commands:	
<code>\c_space_tl</code> 2676, 4348, 4363, 4386, 4390
<code>\tl_clear:N</code>	.. 497, 503, 1972, 2032, 2042, 2063, 2071, 2263, 2453, 2454, 2537, 2611, 3985
<code>\tl_clear_new:N</code> 454
<code>\tl_const:Nn</code> 46, 438
<code>\tl_gclear:N</code>	.. 323, 324, 325, 1556, 1561, 2532, 2887, 3521, 3709, 3880
<code>\tl_gclear_new:N</code> 1543
<code>\tl_gput_right:Nn</code> 439
<code>\tl_greplace_all:Nnn</code> 460
<code>\tl_gset:Nn</code>	255, 256, 269, 270, 1544, 1557, 1562, 1781, 3652
<code>\tl_gset_eq:NN</code> 456, 2796, 3702
<code>\tl_if_blank:nTF</code> 3700
<code>\tl_if_empty:NTF</code>	.. 555, 574, 602, 619, 639, 646, 672, 689, 1569, 1574, 1596, 1601, 1659, 1723, 1731, 1760,

1820, 1940, 2121, 2152, 2283, 2490, 2512, 2514, 2548, 2621, 2670, 2866, 3988, 4278	
<code>\tl_if_empty:nTF</code> 1624, 2241
<code>\tl_if_exist:NTF</code> 1629
<code>\tl_if_novalue:nTF</code>	.. 2259, 2545, 2619, 2655, 2749, 2774, 2792, 2797, 2826, 3040, 3343, 3562, 3922, 3986, 4242
<code>\tl_map_inline:Nn</code> 209, 457
<code>\tl_new:N</code>	.. 43, 48, 49, 52, 53, 59, 61, 62, 63, 65, 66, 99, 100, 101, 107, 108, 109, 110, 112, 113, 114, 115, 116, 117, 121, 124, 126, 127, 133, 134, 144, 145, 146, 153, 154, 155, 158, 175, 178
<code>\tl_put_left::Ne</code> 2479
<code>\tl_put_left:Nn</code>	2129, 2162, 2268, 2473, 2486, 2492, 2502, 2682, 2717, 4004, 4007
<code>\tl_put_right:Nn</code>	455, 565, 610, 627, 680, 2133, 2168, 2270, 2275, 2282, 2285, 2295, 2300, 2303, 2309, 2335, 2345, 2359, 2375, 2381, 2386, 2540, 2543, 2550, 2552, 2579, 2584, 2589, 2592, 2601, 2614, 2617, 2623, 2628, 2638, 3990, 3991
<code>\tl_remove_all:Nn</code> 4277
<code>\tl_remove_once:Nn</code> 2323, 2564
<code>\tl_replace_all:Nnn</code> 459
<code>\tl_reverse:N</code> 2322, 2324, 2563, 2565
<code>\tl_set:Nn</code>	.. 54, 282, 289, 296, 424, 498, 502, 507, 508, 554, 599, 671, 819, 833, 845, 857, 1658, 1759, 2033, 2043, 2064, 2072, 2414, 2657, 2692, 2705, 2794, 3993, 4016, 4275
<code>\tl_set_eq:NN</code>	465, 560, 563, 607, 609, 624, 626, 677, 679, 2321, 2562, 2575, 2838, 2842, 3324, 3326
<code>\tl_to_str:n</code> 1629, 1635, 1640, 4125
<code>\tl_trim_spaces:n</code> 455, 4263, 4275, 4281
<code>\tl_use:N</code>	.. 461, 464, 576, 641, 648, 691, 890, 894, 898, 902, 906, 910, 914, 918, 922, 926, 930, 934, 938, 942, 946, 950, 2191, 2328, 2336, 2347, 2361, 2366, 2378, 2781, 2787, 2811, 2829, 2833, 2841, 2878, 2879, 2886, 2894, 2895, 2901, 3016, 3183, 3329, 3515, 3712, 3723, 3727, 3874, 4056, 4067, 4073, 4078, 4179, 4180, 4181, 4182, 4183, 4201, 4259

token commands:

<code>\token_to_str:N</code> 408
<code>\topsep</code> 2158, 2166
<code>topsep</code> <u>744</u>
<code>\typeout</code> 376, 379, 389, 390

U

<code>\u</code> 212, 2459
use commands:	
<code>\use:N</code> 219, 2883, 3018
<code>\use:n</code> 1505, 2076, 4131
<code>\use_none:nn</code> 400
<code>\usecounter</code> 2950, 2992

V

<code>\value</code>	... 1572, 1578, 1585, 1591, 1599, 1605, 1612, 1618
<code>\vspace</code>	361, 1404, 1407, 1418, 1421, 1431, 1433, 1442, 1444, 1453, 1455, 1464, 1466, 1475, 1477, 1486, 1488, 2157, 2165, 3340, 3351, 3763, 4118

W

<code>widest</code> <u>724</u>
<code>wrap-ans</code> <u>1974</u>
<code>wrap-label</code> <u>468</u>
<code>wrap-label*</code> <u>468</u>
<code>wrap-opt</code> <u>1974</u>