

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-06*

©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 `multicol` and `scontents` packages 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>	11
1.3.3	Support for minipage	3	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>anskeyenv</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	20
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	120
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>	9			
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 \LaTeX 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 2024-06-06.

†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] and `scontents`[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. 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 / 133

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`[6], 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`[6] 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`[6] 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`[8] 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`[9] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

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

2 The environments provided

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

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

2.1 The environment `enumext`

The `enumext` is an environment that works in the same way as the standard `enumerate` environment provided by \LaTeX , `\item` and `\item[\langle custom \rangle]` commands work in the usual way. The environment can be nested with at most “four levels” and the options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Example with `columns=2`

1. This text is in the first level.

(a) This text is in the second level.

i. This text is in the third level.
- A. This text is in the fourth level.

X This text is in the first level.

★ 2. This text is in the first level.

2.2 The environment `enumext*`

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

Some considerations to take into account for this environment:

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

Example with `columns=2`

1. This text is in the first level.

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

★ 3. This text is in the first level.

2.3 The command `\item*`

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

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

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

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

2.3.1 Keys for `\item*`

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

default: $\$ \star \$$

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

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

default: *by levels*

Sets the *offset* between the box containing the current $\langle label \rangle$ defined by `labelwidth` key and the $\langle symbol \rangle$ set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

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

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

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

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

3 The command `\setenumext`

`\setenumext`

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

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

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

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

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

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

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

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

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

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

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

4 The keyval system

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

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

All $\langle keys \rangle$ related to vertical or horizontal spacing accept a “skip” or “dim” expression if passed between braces, i.e. you do not need to use `\dimeval` or `\dimexpr` to perform calculations.

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

4.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2 Keys for spaces

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

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

4.2.1 Vertical spaces

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

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

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

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

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

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

- The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T_EX will enter \langle vertical mode \rangle and apply this value to the “top” and “bottom” the environment or nested level.

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

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

`noitemsep` \langle value forbidden \rangle default: *not used*

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

`nosep` \langle value forbidden \rangle default: *not used*

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

- The following \langle keys \rangle should be used with “caution”, they are intended to be used at the “top” and “bottom” of the environment when the `columns` or `mini-env` keys do not provide adequate *vertical spaces*. The values passed can be *rubber* or *rigid* lengths, the way they are applied is the way you differ, using the star ‘*’ \langle keys \rangle applies `\vspace*` so that T_EX does *not discard* this space at page break.

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

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

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

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

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

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

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

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

4.2.2 Horizontal spaces

`itemindent` = { \langle rigid length \rangle } default: 0pt

Extra *horizontal indentation*, beyond `labelsep`, of the “first line” off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.

`rightmargin` = { \langle rigid length \rangle } default: 0pt

Set the *horizontal space* between the right margin of the environment and the right margin of the enclosing environment, the value it takes must be greater than or equal to 0pt. Internally sets the value of `\rightmargin` for the current level.

`listparindent` = {*<rigid length>*} default: 0pt
 Sets the *horizontal space* indentation, beyond `list-indent`, for second and subsequent paragraphs within a list item. Internally sets the value of `\listparindent` for the current level.

`list-offset` = {*<rigid length>*} default: 0pt
 Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.

`list-indent` = {*<rigid length>*} default: labelwidth + labelsep
 Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.

If `list-indent=0pt` is set in the environment `enumext` the *<label>* will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “*common paragraph*”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

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

4.3 Keys for add code

- The following *<keys>* should be used with “*caution*”, they are intended to inject *{<code>}* into different parts of the defined environments. We must keep in mind that the defined environments are based on the `list` base environment provided by \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`[8] 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 anskeyenv

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

The environment `anskeyenv` 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. If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “*label and ref*” system provided by \TeX will be used.

By design the environment cannot be nested but full supports *verbatim material* 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*” unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskeyenv` environment is implemented using the `scontents` package, for the correct operation `\begin{anskeyenv}` and `\end{anskeyenv}` must be in different lines, all `\langle keys \rangle` must be passed separated by commas and “without separation” of the start of the environment. Comments “%” or “any character” after `\begin{anskeyenv}` or `[\langle key = val \rangle]` on the same line are NOT supported, the package `scontents` will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{anskeyenv}` on the same line the package `scontents` will return a “warning” message.

The `anskeyenv` environment uses the same `\langle keys \rangle` as the `\anskey` command next to the keys `write-env`, `force-eol` and `overwrite` inherited from package `scontents`. The environment 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.

- 🔒 For security reasons the keys `store-env`, `print-env` and `write-out` they have been left disabled. It is recommended that you review the `scontents`[4] documentation to understand how the keys described here work.

Example

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

- | | |
|--|---|
| <ul style="list-style-type: none"> ★ 5. Text containing our instructions or questions. [5] First answer with verbatim 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.


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

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

The $\langle keys \rangle$ 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[<keyans>]{<key = val>}` or `\setenumext[<keyans*>]{<key = val>}`. If the optional argument is not passed or the $\langle keys \rangle$ are not set by `\setenumext`, the default values will be the same as the second level of the `enumext` environment with the difference in the $\langle label \rangle$ which will be set to `label=\Alph*`.

5.4.1 The `\item*` in `keyans` and `keyans*`

```
\item* \item*
\item* [<content>]
```

The `\item*` and `\item* [<content>]` command “store” the current $\langle label \rangle$ set by `label` key next to the $\langle content \rangle$ (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.

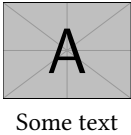
Example

```
\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



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.

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



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

`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}
```

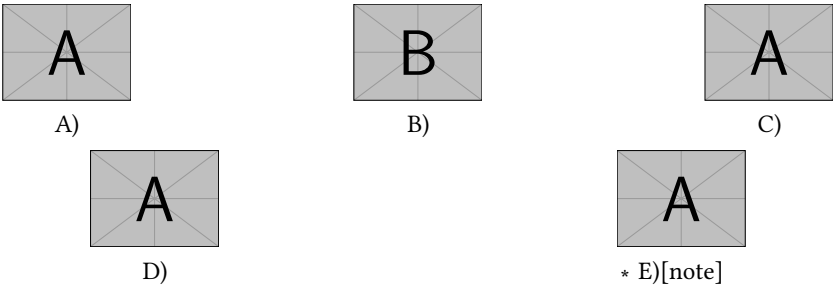
The `\anspic` command take three arguments, the *starred argument* `*` store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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.



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 \printkeyans[⟨keys⟩]{⟨store name⟩}
\printkeyans* [⟨keys⟩]{⟨store name⟩}
```

The command `\printkeyans` prints “all stored content” in sequence $\{\langle store\ name\rangle\}$ 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 $\{\langle store\ name\rangle\}$ does not exist the command will return an error.

The optional argument allows managing the $\langle keys\rangle$ in the “first level” of the environment in which the “stored content” of the *sequence* $\{\langle store\ name\rangle\}$ 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* $\{\langle store\ name\rangle\}$ 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* $\{\langle store\ name\rangle\}$ 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*{\langle store\ name\rangle}` and the *sequence* $\{\langle store\ name\rangle\}$ already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeyans*{\langle store\ name\rangle}` and the *sequence* $\{\langle store\ name\rangle\}$ contains any `enumext` environments, they will start with the $\langle keys\rangle$ 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{\langle store\ name\rangle}` and the *sequence* $\{\langle store\ name\rangle\}$ contains any environment `enumext*`, they will start with the $\langle keys\rangle$ 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 $\langle keys\rangle$ for the environment `enumext` “saved” in the *sequence* $\{\langle store\ name\rangle\}$ we will use `\setenumext[⟨print , level⟩]{⟨keys⟩}` and if we need to set the $\langle keys\rangle$ for the environment `enumext*` “saved” in the *sequence* $\{\langle store\ name\rangle\}$ 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) ~~ETEX~~e is cool?

[2] Very True!

3. Related to Linux

(a) You use linux?

[3] Yes

(b) Rate the following package and class

i. xsim

[4] very good

ii. exsheets

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

6 Full examples

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

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

and then you can use the excellent arara¹ tool to compile them.

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

D $3,60 \times 10^8$ km/h.
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.
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


*

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

- 2. A
- 3. B
- 4. A

*
*
*

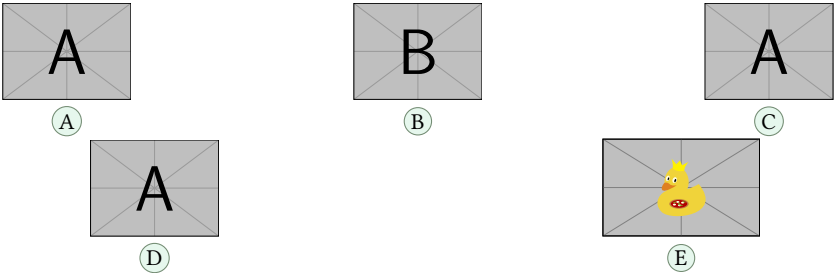
Example 3

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
- 3. Third type of questions
 - (1) $2\alpha + 2\delta = 90^\circ$
 - (2) $\angle EDF = 45^\circ$
 - (A) value
 - (B) value
 - (C) value
- 4. Question with image and label below:

- (D) I and III only
- (E) I, II, and III

- (D) value
- (E) value



- 5. Question with image on left side:
 - (A) value
 - (B) value
 - (C) value
 - (D) correct
 - (E) value




Test keys





- 1. B, $x = 5$
- 2. D
- 3. C, some note

- * 4. E, A duck
- * 5. D, other note
- *

*
*

Example 4

A “simple worksheet” using ducks :) .

-  Factor $x^2 - 2x + 1$
 -  Factor $3x + 3y + 3z$
 - The following questions need to be cuaqtified :)
 -  True False
 - (a) $\alpha > \delta$
 - (b) \LaTeX is cool?
 -  Related to Linux
 - (a) You use linux?
 - (b) Usually uses the package manager?
 - (c) Rate the following package and class
 - i. `xsim-exam`
 - ii. `xsim`
 - iii. `exsheets`
- The answer to 1 is $(x - 1)^2$ and the answer to 3.(a) is False.

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

* ii. very good

* iii. obsolete

*
- *


*

*

*

*

Example 5

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

1

Which choice best describes what happens in the passage?

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

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

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

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

2

Which choice best describes what happens in the passage?

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

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

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

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

3

Which choice best describes what happens in the passage?

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

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

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

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

4

Which choice best describes what happens in the passage?

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

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

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

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

1. A)

2. C)

3. B)

4. D)

7 The way of non-enumerated lists

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

Fake itemize environment

Here we set the `label` key using the default settings in \LaTeX for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

- First level item

– Second level item

* Third level item

· Fourth level item

• First level item
- * First level item

◇ Second level item

○ Third level item

★ Fourth level item

* First level item

Fake description environment

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

Something A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

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

Description indented by label

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate`[5] 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`[11] and `l3seq`[11] modules together with the `hyperref`[8] and `enumitem`[6] 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] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.

- [12] The L^AT_EX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.
- [13] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [14] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [15] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

9 Change history

v1.0 2024-06-06 – First public release.

10 Index of Documentation

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

C

Document class:

article2

book2

exam2

letter2

report2

\columnbreak4, 12

\columnsep10

Commands provide by enumext:

\anskey4, 10–13

\anspic*4, 10–12, 14, 15

\anspic14, 15

\getkeyans4, 12, 15

\item*4–7, 10–14

\itemwidth5

\item5–7, 9, 10, 12, 13

\miniright4, 10

\printkeyans4, 6, 11, 16

\setenumext4–7, 11, 14, 16

Counters defined by enumext:

enumXiii4

enumXii4

enumXiv4

enumXi4

enumXviii4

enumXvii4

enumXvi4

enumXv4

E

Environments provide by enumext:

anskeyenv13

enumext*4–14, 16

enumext4–14, 16, 19

keyans*4–14

keyanspic4, 6, 8, 10–15, 19

keyans4–15, 19

Environments:

enumerate1, 3, 5, 21

figure5

list3, 9, 21

minipage3–5, 10, 21

multicols3, 4, 10

table5

task5

F

\footnote5

I

\item3, 5

\itemsep8

K

Keys for command provide by enumext:

break-col12

item-join12

item-pos*12

item-star12

item-sym*12

Keys for environments provide by enumext:

above*8

above8

after9, 10

align7, 20

before*9

before9

below*8

below8

check-ans12

columns-sep4, 10

columns4, 8, 10

first9

font7

item-pos*5, 6

item-sym*5, 6

itemindent8

itemsep8, 15

labelsep3, 5–10, 12, 20

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

labelwidth5

label7, 9, 14, 19, 20

list-indent3, 9

list-offset3, 9, 20

listparindent9

mark-ans11, 12

mark-pos12

mark-ref11

mini-env4, 8, 10

mini-right*6, 10

mini-right6, 10

mini-sep4, 10

no-store11–13

noitemsep8

nosep8, 19

parsep8, 15

partopsep8

ref4, 7

resume*6, 9, 10

resume6, 9, 10

rightmargin8

save-ans4, 6, 9–16

save-key9, 11, 16

save-ref4, 7, 11–13, 15

save-sep11

series6, 9, 10

show-ans11

show-length7

show-pos11, 12, 15

start9

topsep8

widest7

wrap-ans11

wrap-label*7, 20

wrap-label7

wrap-opt11

L

\label4

Labels provide by enumext:

\Alph*7, 14

<code>\Roman*</code>	7	<code>l3prop</code>	1, 21
<code>\alph*</code>	7	<code>l3seq</code>	1, 21
<code>\arabic*</code>	7	<code>multicol</code>	1, 2, 4, 21
<code>\roman*</code>	7	<code>scontents</code>	1, 2, 13
<code>\labelsep</code>	3, 7	<code>task</code>	5, 6
<code>\labelwidth</code>	3, 7	<code>xsim</code>	2
<code>\linewidth</code>	10	<code>\parsep</code>	8
<code>\listparindent</code>	9	<code>\partopsep</code>	8

P

Packages:

<code>enumerate</code>	21
<code>enumext</code>	1–6, 15, 21
<code>enumitem</code>	3–5, 9, 20, 21
<code>footnotehyper</code>	4, 5
<code>hyperref</code>	4, 5, 11–13, 21
<code>l3keys</code>	6

R

<code>\raggedcolumns</code>	4
<code>\ref</code>	4
<code>\rightmargin</code>	8

T

<code>\topsep</code>	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   {2024-06-06}
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}[2024-05-23]
18   }
19   \IfPackageLoadedTF { scontents }
20   {
21     \msg_info:nnn { enumext } { package-load } { scontents }
22   }
23   {
24     \msg_info:nnn { enumext } { package-not-load } { scontents }
25     \RequirePackage{scontents}
26   }
27 }
```

11.4 Definition of variables

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

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

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

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

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

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

The boolean variables `\g__enumext_starred_bool` and `\g__enumext_standar_bool` will be set to “true” when the `enumext` and `enumext*` environments are not nested with each other.

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

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

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

Variables to store the “name of the counters” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment. The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments.

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

```
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { l__enumext_counter_#1_tl }
44 }
45 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

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

```
46 \tl_const:Nn \c__enumext_counter_style_tl
47 { { arabic } { roman } { Roman } { alph } { Alph } }
48 \tl_new:N \l__enumext_ref_key_arg_tl
49 \tl_new:N \l__enumext_ref_the_count_tl
50 \cs_set_protected:Npn \__enumext_tmp:n #1
51 {
52   \tl_new:c { l__enumext_renew_the_count_#1_tl }
53   \tl_new:c { l__enumext_the_counter_#1_tl }
54   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
55 }
56 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by `resume`, `resume*` and `series` keys (§11.22). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§11.29).

```
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_standar_series_tl
62 \tl_new:N \g__enumext_starred_series_tl
63 \tl_new:N \g__enumext_item_symbol_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:N` (§11.33.1).

```

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 (§11.15.1), 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 (§11.17).

```

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_env_tl
\l__enumext_store_anskey_opt_tl
\l__enumext_store_columns_join_int
\l__enumext_store_keyans_label_tl
\l__enumext_store_keyans_item_opt_tl
\l__enumext_keyans_item_opt_tl
\l__enumext_keyans_tmpa_tl
```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§11.23.1) activates all the mechanism related to `\anskey`, `anskeyenv`, `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_name_tl` save 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` (§11.26) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§11.31.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§11.36.1) 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_env_tl
111 \tl_new:N \l__enumext_store_anskey_opt_tl
112 \int_new:N \l__enumext_store_columns_join_int
113 \tl_new:N \l__enumext_store_keyans_label_tl
114 \tl_new:N \l__enumext_store_keyans_item_opt_tl
115 \tl_new:N \l__enumext_keyans_item_opt_tl
116 \tl_new:N \l__enumext_keyans_tmpa_tl
```

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

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

Internal variables used by the command `\setenumext` (§11.41).

```
117 \tl_new:N \l__enumext_setkey_tmpa_tl
118 \tl_new:N \l__enumext_setkey_tmpb_tl
119 \int_new:N \l__enumext_setkey_tmpa_int
120 \seq_new:N \l__enumext_setkey_tmpa_seq
121 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

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

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§11.40) and `save-key` key.

```
122 \tl_new:N \l__enumext_print_keyans_starred_tl
123 \cs_set_protected:Npn \l__enumext_tmp:n #1
124 {
125   \tl_new:c { \l__enumext_store_save_key_#1_tl }
126   \bool_new:c { \l__enumext_store_save_key_#1_bool }
127   \tl_new:c { \l__enumext_store_active_keys_#1_tl }
128   \tl_new:c { \l__enumext_print_keyans_#1_tl }
129   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
130 }
131 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__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` (§11.26), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§11.25).

```
132 \bool_new:N \l__enumext_show_answer_bool
133 \bool_new:N \l__enumext_show_position_bool
134 \tl_new:N \l__enumext_mark_ref_sym_tl
135 \tl_new:N \l__enumext_mark_answer_sym_tl
136 \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 (§11.36.2).

```
137 \seq_new:N \l__enumext_keyans_pic_body_seq
138 \dim_new:N \l__enumext_keyans_pic_width_dim
139 \int_new:N \l__enumext_keyans_pic_above_int
140 \int_new:N \l__enumext_keyans_pic_below_int
141 \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.

```

142 \bool_new:N \l__enumext_check_answers_bool
143 \bool_new:N \l__enumext_check_ans_key_bool
144 \bool_new:N \g__enumext_check_ans_key_bool
145 \tl_new:N \l__enumext_check_start_line_env_tl
146 \tl_new:N \g__enumext_start_line_tl
147 \tl_new:N \g__enumext_envir_name_tl
148 \int_new:N \g__enumext_check_starred_cmd_int
149 \int_new:N \g__enumext_item_anskey_int
150 \int_new:N \g__enumext_item_number_int
151 \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`.

```

152 \bool_new:N \l__enumext_hyperref_bool
153 \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.

```

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

```

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

```

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

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

(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.

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

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

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

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

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

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

```
202 \cs_new:Nn \__enumext_level:
203 {
204   \int_to_roman:n { \__enumext_level_int }
205 }
```

(End of definition for `__enumext_level:`.)

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

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

(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.

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

(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.

```

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

```

(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.

```

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

```

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.

```

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

```

```

276         {
277             on ~ line ~ \exp_not:V \inputlineno
278         }
279     }
280 }

```

(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.

```

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

```

(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.

```

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

```

(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.

```

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

```

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.

```

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

```

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

11.6 Copying list and minipage environments

The `list` environment provided by \TeX has the following plain form:

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

(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 (§11.34) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.37)

```

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

(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.

```

371 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
372 \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”.

```

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

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`.

```

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

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.

```

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

```

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

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

#1: `__enumext_newlabel_arg_one_tl`

#2: `__enumext_newlabel_arg_two_tl`

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

```

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

```

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

11.9 Definition of counters

`__enumext_define_counters:Nn` To create the necessary “counters” we must first make sure that they are not already defined by the user or a package such as `enumitem`, otherwise a error will be returned and the package loading will be aborted. The arguments taken by the function are:

#1: A token list `__enumext_counter_X_tl` for “store” the counter’s name.

#2: The counter’s name.

```

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

```

(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   433 \__enumext_define_counters:Nn \__enumext_counter_i_tl   { enumXi   }
enumXii  434 \__enumext_define_counters:Nn \__enumext_counter_ii_tl  { enumXii  }
enumXiii 435 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv  436 \__enumext_define_counters:Nn \__enumext_counter_iv_tl  { enumXiv  }
enumXvii 437 \__enumext_define_counters:Nn \__enumext_counter_v_tl   { enumXv   }
enumXviii438 \__enumext_define_counters:Nn \__enumext_counter_vi_tl  { enumXvi  }
          439 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
          440 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

11.10 Definition of labels

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

`__enumext_register_counter_style:Nn`

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

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

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

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

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

```

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

```

495 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
496 {
497     \keys_define:nn { enumext / #1 }
498     {
499         align .choice:,
500         align / left .code:n =
501             {
502                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
503                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
504             },
505         align / right .code:n =
506             {
507                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
508                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
509             },
510         align / center .code:n =
511             {
512                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
513                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
514             },
515         align .initial:n = left,
516         align .value_required:n = true,
517     }
518 }
519 \clist_map_inline:nn
520 {
521     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
522 }
523 { \__enumext_tmp:nn #1 }

524 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
525 {
526     \keys_define:nn { enumext / #1 }
527     {
528         align .choice:,
529         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
530         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
531         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
532         align .initial:n = left,
533         align .value_required:n = true,
534     }
535 }
536 \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 537 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 538 {
\l__enumext_label_iii_tl 539   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 540   {
541     label .code:n = {
542       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
543       { l__enumext_counter_#2_tl } {##1}
544       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
545       \l__enumext_current_widest_dim
546     },
547     label .initial:n = #3,
548     label .value_required:n = true,
549     ref .code:n = \__enumext_standar_ref:n {##1},
550     ref .value_required:n = true,
551   }
552 }
553 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
554 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
555 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
556 \__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.

```

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

```

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

```

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

```

(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 584 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 585 {

```



```

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

```

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

```

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.

```

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

```

```

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

```

(End of definition for `__enumext_starred_ref:n` and `__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.

```

\__enumext_label_v_tl 657 \keys_define:nn { enumext / keyans }
\__enumext_label_vi_tl 658 {
659   label .code:n = {
660     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
661     { \l__enumext_counter_v_tl } {#1}
662     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
663     \l__enumext_current_widest_dim
664     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
665     { \l__enumext_counter_vi_tl } {#1}
666     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
667     \l__enumext_current_widest_dim
668   },
669   label .initial:n = \Alph*,
670   label .value_required:n = true,
671   ref .code:n = \__enumext_keyans_ref:n {#1},
672   ref .value_required:n = true,
673 }

```

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

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

```

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

```

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

```

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

```

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

11.13 Setting start and widest keys

```
\__enumext_start_from:NNn
\__enumext_start_from:ccn
```

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

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

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

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

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

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

```

```

744         widest .value_required:n = true,
745         start .value_required:n = true,
746     }
747 }
748 \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
749 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
750 {
751     \keys_define:nn { enumext / #1 }
752     {
753         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
754         topsep .initial:n = {#3},
755         topsep .value_required:n = true,
756         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
757         partopsep .initial:n = {#4},
758         partopsep .value_required:n = true,
759         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
760         parsep .initial:n = {#5},
761         parsep .value_required:n = true,
762         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
763         itemsep .initial:n = {#6},
764         itemsep .value_required:n = true,
765         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
766         noitemsep .value_forbidden:n = true,
767         nosep .meta:n = {
768             itemsep = 0pt, parsep = 0pt,
769             topsep = 0pt, partopsep = 0pt,
770         },
771         nosep .value_forbidden:n = true,
772     }
773 }

```

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

```

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

```

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

```

```

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

```

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

```

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

```

```

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

```

(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.

```

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

before*

after

first

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

```

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

`__enumext_before_keys_exec:`

`__enumext_after_stop_list:`

`__enumext_after_args_exec:`

```

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

```

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 $\langle keys \rangle$.

```

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

```


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

```

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

```

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

```

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

```

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

11.17.2 Functions for before, after and first keys in keyans

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:

```

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

```

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

```

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

```

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

```

```

939   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
940 }

```

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

```

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

```

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`.

```

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

```

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

11.18 Setting keys for multicols and minipage

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

```

957 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
958 {
959   \keys_define:nn { enumext / #1 }
960   {
961     mini-env    .dim_set:c = { l__enumext_minipage_right_#2_dim },
962     mini-env    .value_required:n = true,
963     mini-sep    .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
964     mini-sep    .initial:n = 0.3333em,
965     mini-sep    .value_required:n = true,
966     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
967     columns-sep .value_required:n = true,
968     columns     .int_set:c = { l__enumext_columns_#2_int },
969     columns     .initial:n = 1,
970     columns     .value_required:n = true,
971   }
972 }
973 \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.

```

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

```

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

```

(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*.

```

1000 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1001 {
1002   \int_case:nn { \__enumext_level_int }
1003   {
1004     { 2 }{
1005       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip } {
1006         {
1007           \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
1008         }
1009       }
1010     { 3 }{
1011       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip } {
1012         {
1013           \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
1014         }
1015       }
1016     { 4 }{
1017       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip } {
1018         {
1019           \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
1020         }
1021       }

```

```

1022     }
1023 }

```

(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*.

```

1024 \cs_new_protected:Nn \__enumext_multi_addvspace:
1025 {
1026   \__enumext_multi_set_vskip:
1027   \mode_if_vertical:T
1028   {
1029     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
1030     {
1031       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1032     }
1033     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
1034     {
1035       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1036     }
1037   }
1038   \par\nopagebreak
1039   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
1040 }

```

(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:`

```

1041 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1042 {
1043   \skip_set:Nn \__enumext_multicols_above_v_skip
1044   {
1045     \__enumext_topsep_v_skip
1046   }
1047   \skip_set:Nn \__enumext_multicols_below_v_skip
1048   {
1049     \__enumext_topsep_v_skip
1050   }
1051 }
1052 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1053 {
1054   \__enumext_keyans_multi_set_vskip:
1055   \mode_if_vertical:T
1056   {
1057     \skip_add:Nn \__enumext_multicols_above_v_skip
1058     {
1059       \skip_use:N \__enumext_partopsep_v_skip
1060     }
1061     \skip_add:Nn \__enumext_multicols_below_v_skip
1062     {
1063       \skip_use:N \__enumext_partopsep_v_skip
1064     }
1065   }
1066   \par\nopagebreak
1067   \addvspace{ \__enumext_multicols_above_v_skip }
1068 }

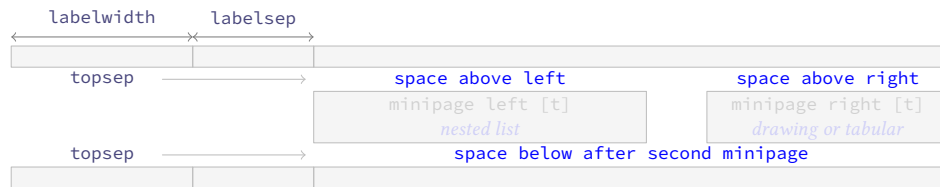
```

(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`[13] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

11.20.1 Adjustment of vertical spaces for minipage in enumext

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

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

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

```

1069 \cs_new_protected:Nn __enumext_mini_set_vskip:
1070 {
1071   \int_compare:nNnTF
1072     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1073     {

```

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.

```

1074   \skip_if_eq:nNnTF
1075     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1076   {
1077     \skip_set:Nn \l__enumext_minipage_left_skip
1078       {
1079         -0.150\box_dp:N \strutbox
1080       }
1081     \skip_set:Nn \l__enumext_minipage_right_skip
1082       {
1083         0.695\box_dp:N \strutbox
1084       }
1085     \skip_set:Nn \l__enumext_minipage_after_skip
1086       {
1087         \box_dp:N \strutbox
1088       }
1089     __enumext_zero_parsep:
1090   }
1091   {
1092     \skip_set:Nn \l__enumext_minipage_left_skip
1093       {
1094         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1095       }
1096     \skip_set:Nn \l__enumext_minipage_right_skip
1097       {
1098         0.695\box_dp:N \strutbox
1099       }
1100     \skip_set:Nn \l__enumext_minipage_after_skip
1101       {
1102         1.85\box_dp:N \strutbox
1103         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1104       }

```

```

1105     }
1106   }
1107   {

```

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.

```

1108     \skip_if_eq:nnTF
1109     { \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip } } { \c_zero_skip }
1110     {
1111       \skip_set:Nn \l__enumext_minipage_left_skip
1112       {
1113         0.5\box_dp:N \strutbox
1114         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1115       }
1116       \skip_set:Nn \l__enumext_minipage_right_skip
1117       {
1118         \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1119       }
1120       \skip_set:Nn \l__enumext_minipage_after_skip
1121       {
1122         1.6\box_dp:N \strutbox
1123       }
1124     }
1125     {
1126       \skip_set:Nn \l__enumext_minipage_left_skip
1127       {
1128         0.5875\box_dp:N \strutbox
1129         - \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1130       }
1131       \skip_set:Nn \l__enumext_minipage_right_skip
1132       {
1133         + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1134         + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1135       }
1136       \skip_set:Nn \l__enumext_minipage_after_skip
1137       {
1138         0.325\box_dp:N \strutbox
1139         + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1140       }
1141     }
1142   }
1143 }

```

(End of definition for `\l__enumext_mini_set_vskip:`)

`\l__enumext_zero_parsep:`

The function `\l__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.

```

1144 \cs_new_protected:Nn \l__enumext_zero_parsep:
1145 {
1146   \int_case:nn { \l__enumext_level_int }
1147   {
1148     { 2 } {
1149       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1150       {
1151         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1152       }
1153     }
1154     { 3 } {
1155       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1156       {
1157         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1158       }
1159     }
1160     { 4 } {
1161       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1162       {
1163         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1164       }
1165     }

```



```

1166     }
1167 }

```

(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*.

```

1168 \cs_new_protected:Nn \__enumext_mini_addvspace:
1169 {
1170   \__enumext_mini_set_vskip:
1171   \mode_if_vertical:T
1172   {
1173     \skip_add:Nn \l__enumext_minipage_left_skip
1174     {
1175       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1176     }
1177     \skip_add:Nn \l__enumext_minipage_after_skip
1178     {
1179       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1180     }
1181   }
1182   \par\nopagebreak
1183   \addvspace { \l__enumext_minipage_left_skip }
1184 }

```

(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`.

```

1185 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1186 {
1187   \skip_zero_new:N \l__enumext_minipage_after_skip
1188   \skip_zero_new:N \l__enumext_minipage_left_skip
1189   \skip_zero_new:N \l__enumext_minipage_right_skip
1190   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1191   {
1192     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1193     {
1194       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1195       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1196       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1197       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1198       {
1199         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1200       }
1201     }
1202     {
1203       \skip_set:Nn \l__enumext_minipage_left_skip
1204       {
1205         \skip_use:N \l__enumext_topsep_v_skip
1206       }
1207       \skip_set:Nn \l__enumext_minipage_right_skip
1208       {
1209         0.705\box_dp:N \strutbox
1210       }
1211       \skip_set:Nn \l__enumext_minipage_after_skip
1212       {
1213         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1214       }
1215     }
1216   }
1217   {
1218     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1219     {
1220       \skip_set:Nn \l__enumext_minipage_left_skip
1221       {

```

```

1222         0.5\box_dp:N \strutbox
1223         + \l__enumext_partopsep_v_skip
1224     }
1225     \skip_set:Nn \l__enumext_minipage_right_skip
1226     {
1227         \l__enumext_partopsep_v_skip
1228     }
1229     \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1230 }
1231 {
1232     \skip_set:Nn \l__enumext_minipage_left_skip
1233     {
1234         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1235     }
1236     \skip_set:Nn \l__enumext_minipage_right_skip
1237     {
1238         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1239     }
1240     \skip_set:Nn \l__enumext_minipage_after_skip
1241     {
1242         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1243     }
1244 }
1245 }
1246 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

The function `__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `__enumext_mini_env*` environment in `keyans`, taking into account whether \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`.

```

1247 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1248 {
1249     \__enumext_keyans_mini_set_vskip:
1250     \mode_if_vertical:T
1251     {
1252         \skip_add:Nn \l__enumext_minipage_left_skip
1253         {
1254             \l__enumext_partopsep_v_skip
1255         }
1256         \skip_add:Nn \l__enumext_minipage_after_skip
1257         {
1258             \l__enumext_partopsep_v_skip
1259         }
1260     }
1261     \par\nopagebreak
1262     \addvspace { \l__enumext_minipage_left_skip }
1263 }

```

(End of definition for `__enumext_keyans_mini_addvspace:`)

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

`__enumext_mini_set_vskip_vii:`

`__enumext_mini_set_vskip_viii:`

The functions `__enumext_mini_set_vskip_vii:` and `__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `__enumext_mini_env*` environment in `enumext*` and `keyans*`.

```

1264 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1265 {
1266     \skip_zero_new:N \l__enumext_minipage_left_skip
1267     \skip_gzero_new:N \g__enumext_minipage_right_skip
1268     \skip_gzero_new:N \g__enumext_minipage_after_skip
1269     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1270     {
1271         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1272         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1273     }
1274     {
1275         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1276         \skip_gset:Nn \g__enumext_minipage_right_skip

```

```

1277         {
1278             \l__enumext_topsep_vii_skip
1279         }
1280         \skip_gset:Nn \g__enumext_minipage_after_skip
1281         {
1282             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1283         }
1284     }
1285 }
1286 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1287 {
1288     \skip_zero_new:N \l__enumext_minipage_after_skip
1289     \skip_zero_new:N \l__enumext_minipage_left_skip
1290     \skip_zero_new:N \l__enumext_minipage_right_skip
1291     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1292     {
1293         \skip_set:Nn \l__enumext_minipage_left_skip
1294         {
1295             0.5\box_dp:N \strutbox
1296         }
1297         \skip_set:Nn \l__enumext_minipage_right_skip
1298         {
1299             \l__enumext_partopsep_viii_skip
1300         }
1301         \skip_set:Nn \l__enumext_minipage_after_skip
1302         {
1303             1.6\box_dp:N \strutbox
1304         }
1305     }
1306     {
1307         \skip_set:Nn \l__enumext_minipage_left_skip
1308         {
1309             0.5875\box_dp:N \strutbox
1310         }
1311         \skip_set:Nn \l__enumext_minipage_right_skip
1312         {
1313             \l__enumext_topsep_viii_skip
1314         }
1315         \skip_set:Nn \l__enumext_minipage_after_skip
1316         {
1317             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1318         }
1319     }
1320 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

Here we will NOT take into account whether \TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1321 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1322 {
1323     \__enumext_mini_set_vskip_vii:
1324     \par\nopagebreak
1325     \addvspace { \l__enumext_minipage_left_skip }
1326 }
1327 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1328 {
1329     \__enumext_mini_set_vskip_viii:
1330     \par\nopagebreak
1331     \addvspace { \l__enumext_minipage_left_skip }
1332 }

```

(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.

```

1333 \NewDocumentCommand \miniright { s }
1334 {
1335   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1336   {
1337     \msg_error:nnn { enumext } { wrong-miniright-place }
1338   }
1339   \int_compare:nNt { \l__enumext_level_int } = { 0 }
1340   {
1341     \msg_error:nnn { enumext } { wrong-miniright-place }
1342   }
1343   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1344   {
1345     \__enumext_keyans_mini_right_cmd:n {#1}
1346   }
1347   { \__enumext_mini_right_cmd:n {#1} }
1348 }

```

(End of definition for `\miniright`. This function is documented on page 9.)

`__enumext_mini_right_cmd:n` The function `__enumext_mini_right_cmd:n` takes as argument the *starred* ‘*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\l__enumext_minipage_right_X_dim`, if so we close the `multicols` environment with the `__enumext__mini_env*` environment on the “left side”, then we open the `__enumext_mini_env*` environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the `\centering` command when the starred argument ‘*’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1349 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1350 {
1351   \dim_compare:nNtF
1352   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1353   {
1354     \__enumext_multicols_stop:
1355     \end{__enumext_mini_env*}
1356     \hfill
1357     \begin{__enumext_mini_env*}
1358       { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1359       \par\addvspace { \l__enumext_minipage_right_skip }
1360       \bool_if:nF {#1}
1361       {
1362         \centering
1363       }
1364       \int_gzero:N \g__enumext_minipage_stat_int
1365     }
1366     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1367   }

```

(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.

```

1368 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1369 {
1370   \dim_compare:nNtF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1371   {
1372     \__enumext_keyans_multicols_stop:
1373     \end{__enumext_mini_env*}
1374     \hfill
1375     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1376       \par\addvspace { \l__enumext_minipage_right_skip }
1377       \bool_if:nF {#1}
1378       {
1379         \centering
1380       }

```

```

1381         \int_gzero:N \g__enumext_minipage_stat_int
1382     }
1383     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1384 }

```

(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* \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below  {
below* \keys_define:nn { enumext / #1 }
{
    above .skip_set:c = { l__enumext_vspace_above_#2_skip },
    above .value_required:n = true,
    above* .code:n      = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
                        \keys_set:nn { enumext / #1 } { above = {##1} },
    above* .value_required:n = true,
    below .skip_set:c = { l__enumext_vspace_below_#2_skip },
    below .value_required:n = true,
    below* .code:n      = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
                        \keys_set:nn { enumext / #1 } { below = {##1} },
    below* .value_required:n = true,
}
}
\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.

```

1402 \cs_new_protected:Nn \__enumext_vspace_above:
1403 {
1404     \skip_if_eq:nnF
1405     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1406     {
1407         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1408         {
1409             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1410         }
1411         {
1412             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1413         }
1414     }
1415 }

```

(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.

```

1416 \cs_new_protected:Nn \__enumext_vspace_below:
1417 {
1418     \skip_if_eq:nnF
1419     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1420     {
1421         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1422         {
1423             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1424         }
1425         {
1426             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1427         }
1428     }
1429 }

```

(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.

```

1430 \cs_new_protected:Nn \__enumext_vspace_above_v:
1431 {
1432   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1433   {
1434     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1435     {
1436       \vspace*{ \l__enumext_vspace_above_v_skip }
1437     }
1438     { \vspace { \l__enumext_vspace_above_v_skip } }
1439   }
1440 }

```

(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.

```

1441 \cs_new_protected:Nn \__enumext_vspace_below_v:
1442 {
1443   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1444   {
1445     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1446     {
1447       \vspace*{ \l__enumext_vspace_below_v_skip }
1448     }
1449     { \vspace { \l__enumext_vspace_below_v_skip } }
1450   }
1451 }

```

(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:`

```

1452 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1453 {
1454   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1455   {
1456     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1457     {
1458       \vspace*{ \l__enumext_vspace_above_vii_skip }
1459     }
1460     { \vspace { \l__enumext_vspace_above_vii_skip } }
1461   }
1462 }
1463 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1464 {
1465   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1466   {
1467     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1468     {
1469       \vspace*{ \l__enumext_vspace_above_viii_skip }
1470     }
1471     { \vspace { \l__enumext_vspace_above_viii_skip } }
1472   }
1473 }

```

(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:`

```

1474 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1475 {
1476   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1477   {
1478     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1479     {

```

```

1480         \vspace*{ \l__enumext_vspace_below_vii_skip }
1481     }
1482     { \vspace { \l__enumext_vspace_below_vii_skip } }
1483 }
1484 }
1485 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1486 {
1487     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1488     {
1489         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1490         {
1491             \vspace*{ \l__enumext_vspace_below_viii_skip }
1492         }
1493         { \vspace { \l__enumext_vspace_below_viii_skip } }
1494     }
1495 }

```

(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 `<keys>` passed to the optional argument of the “first level” of the environments `enumext` and `enumext*`, but, discarding some specific `<keys>`. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in [chat-Tex-SX](#)

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1496 \cs_set_protected:Npn \__enumext_tmp:n #1
1497 {
1498     \keys_define:nn { enumext / #1 }
1499     {
1500         series .str_set:N = \l__enumext_series_str,
1501         series .value_required:n = true,
1502         resume .code:n = \__enumext_resume_series:n {##1},
1503         resume* .code:n = \__enumext_resume_starred:,
1504         resume* .value_forbidden:n = true,
1505     }
1506 }
1507 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.22.1 Internal functions for series key

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

```

1508 \cs_new:Npn \__enumext_filter_series:n #1
1509 {
1510     \use:e
1511     {
1512         \keyval_parse:NNn
1513         \__enumext_filter_series_key:n
1514         \__enumext_filter_series_pair:nn {#1}
1515     }
1516 }

```

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

```

1517 \cs_new:Npn \__enumext_filter_series_key:n #1
1518 {
1519     \str_case:nnF {#1}
1520     {
1521         { resume } {}
1522         { resume* } {}
1523     }
1524     { , { \exp_not:n {#1} } }
1525 }

```

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

```

1526 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1527 {
1528     \str_case:nnF {#1}

```



```

1529     {
1530     { series } {}
1531     { resume } {}
1532     { start } {}
1533     { save-ans } {}
1534     { save-key } {}
1535     }
1536     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1537 }

```

(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 (§11.34) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.37).

```

1538 \cs_new_protected:Npn \__enumext_parse_series:n #1
1539 {
1540   \str_if_empty:NTF \l__enumext_series_str
1541   {
1542     \bool_if:NF \l__enumext_resume_active_bool
1543     {
1544       \__enumext_resume_last:n {#1}
1545     }
1546   }
1547   {
1548     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1549     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1550     { \__enumext_filter_series:n {#1} }
1551     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1552     {
1553       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1554     }
1555   }
1556 }

```

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.

```

1557 \cs_new_protected:Npn \__enumext_resume_last:n #1
1558 {
1559   \bool_if:NT \l__enumext_standar_first_bool
1560   {
1561     \tl_gclear:N \g__enumext_standar_series_tl
1562     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1563   }
1564   \bool_if:NT \l__enumext_starred_first_bool
1565   {
1566     \tl_gclear:N \g__enumext_starred_series_tl
1567     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1568   }
1569 }

```

(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 (§11.34) and the `enumext*` environment definition (§11.37).

```

1570 \cs_new_protected:Nn \__enumext_resume_save_counter:
1571 {
1572   \bool_if:NT \g__enumext_standar_bool
1573   {
1574     \tl_if_empty:NF \__enumext_series_str
1575     {
1576       \int_gset_eq:cN
1577       { g__enumext_series_ \__enumext_series_str_int } \value{enumXi}
1578     }
1579     \tl_if_empty:NTF \__enumext_resume_name_tl
1580     {
1581       \str_if_empty:NT \__enumext_series_str
1582       {
1583         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1584       }
1585     }
1586     {
1587       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1588       {
1589         \int_gset_eq:cN
1590         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXi}
1591       }
1592     }
1593     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1594     {
1595       \int_gset_eq:cN
1596       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXi}
1597     }
1598   }
1599   \bool_if:NT \g__enumext_starred_bool
1600   {
1601     \tl_if_empty:NF \__enumext_series_str
1602     {
1603       \int_gset_eq:cN
1604       { g__enumext_series_ \__enumext_series_str_int } \value{enumXvii}
1605     }
1606     \tl_if_empty:NTF \__enumext_resume_name_tl
1607     {
1608       \str_if_empty:NT \__enumext_series_str
1609       {
1610         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1611       }
1612     }
1613     {
1614       \int_if_exist:cT { g__enumext_series_ \__enumext_resume_name_tl_int }
1615       {
1616         \int_gset_eq:cN
1617         { g__enumext_series_ \__enumext_resume_name_tl_int } \value{enumXvii}
1618       }
1619     }
1620     \int_if_exist:cT { g__enumext_resume_ \__enumext_store_name_tl_int }
1621     {
1622       \int_gset_eq:cN
1623       { g__enumext_resume_ \__enumext_store_name_tl_int } \value{enumXvii}
1624     }
1625   }
1626 }

```

(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.

```

1627 \cs_new_protected:Npn \__enumext_resume_series:n #1
1628 {
1629   \tl_if_empty:NTF {#1}
1630   {
1631     \__enumext_resume_counter:n { }
1632   }
1633   {
1634     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1635     {
1636       \__enumext_resume_counter:n {#1}
1637       \bool_if:NT \g__enumext_standar_bool
1638       {
1639         \keys_set:nv { enumext / level-1 }
1640         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1641       }
1642       \bool_if:NT \g__enumext_starred_bool
1643       {
1644         \keys_set:nv { enumext / enumext* }
1645         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1646       }
1647     }
1648     {
1649       \bool_if:NT \g__enumext_standar_bool
1650       {
1651         \msg_error:nnn { enumext } { unknown-series } {#1}
1652       }
1653       \bool_if:NT \g__enumext_starred_bool
1654       {
1655         \msg_error:nnn { enumext } { unknown-series } {#1}
1656       }
1657     }
1658   }
1659 }

```

(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`.

```

1660 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1661 {
1662   \bool_set_true:N \l__enumext_resume_active_bool
1663   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1664   \tl_if_empty:NTF \l__enumext_resume_name_tl
1665   {
1666     \__enumext_resume_counter:
1667   }
1668   {
1669     \__enumext_resume_counter_series:
1670   }
1671   \__enumext_resume_counter_save_ans:
1672 }

```

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.

```

1673 \cs_new_protected:Nn \__enumext_resume_counter:
1674 {
1675   \bool_if:NT \g__enumext_standar_bool
1676   {
1677     \int_gincr:N \g__enumext_resume_int
1678     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1679   }
1680   \bool_if:NT \g__enumext_starred_bool
1681   {
1682     \int_gincr:N \g__enumext_resume_vii_int

```

```

1683     \int_set_eq:Nn \l__enumext_start_vii_int \g__enumext_resume_vii_int
1684   }
1685 }

```

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.

```

1686 \cs_new_protected:Nn \__enumext_resume_counter_series:
1687 {
1688   \bool_if:NT \g__enumext_standar_bool
1689   {
1690     \int_set:Nn \l__enumext_start_i_int
1691     {
1692       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1693     }
1694   }
1695   \bool_if:NT \g__enumext_starred_bool
1696   {
1697     \int_set:Nn \l__enumext_start_vii_int
1698     {
1699       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1700     }
1701   }
1702 }

```

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.

```

1703 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1704 {
1705   \bool_lazy_and:nnT
1706   { \bool_if_p:N \l__enumext_standar_first_bool }
1707   { \bool_if_p:N \l__enumext_store_active_bool }
1708   {
1709     \int_set:Nn \l__enumext_start_i_int
1710     {
1711       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1712     }
1713   }
1714   \bool_lazy_and:nnT
1715   { \bool_if_p:N \l__enumext_starred_first_bool }
1716   { \bool_if_p:N \l__enumext_store_active_bool }
1717   {
1718     \int_set:Nn \l__enumext_start_vii_int
1719     {
1720       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1721     }
1722   }
1723 }

```

(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.

```

1724 \cs_new_protected:Nn \__enumext_resume_starred:
1725 {
1726   \bool_if:NT \g__enumext_standar_bool
1727   {
1728     \tl_if_empty:NF \g__enumext_standar_series_tl
1729     {
1730       \__enumext_resume_counter:n { }
1731       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1732     }
1733   }
1734   \bool_if:NT \g__enumext_starred_bool
1735   {
1736     \tl_if_empty:NF \g__enumext_starred_series_tl
1737     {

```

```

1738         \__enumext_resume_counter:n { }
1739         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1740     }
1741 }
1742 }

```

(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*`.

```

1743 \cs_set_protected:Npn \__enumext_tmp:n #1
1744 {
1745     \keys_define:nn { enumext / #1 }
1746     {
1747         save-ans .code:n = \__enumext_storing_set:n {##1},
1748         save-ans .value_required:n = true,
1749     }
1750 }
1751 \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:`.

```

1752 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1753 {
1754     \msg_term:nnVV { enumext } { save-ans-log }
1755     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1756 }
1757 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1758 {
1759     \msg_term:nnVV { enumext } { save-ans-log-hook }
1760     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1761 }

```

(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.

```

1762 \cs_new_protected:Npn \__enumext_storing_set:n #1
1763 {
1764     \tl_set:Nx \l__enumext_store_name_tl {#1}
1765     \tl_if_empty:NTF \l__enumext_store_name_tl
1766     {
1767         \bool_lazy_or:nnT
1768         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1769         {
1770             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1771         }
1772     }
1773     {
1774         \bool_lazy_or:nnT
1775         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1776         {
1777             \__enumext_start_save_ans_msg:
1778             \__enumext_storing_exec:
1779         }
1780     }
1781 }

```

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, copy `{\store name}` into the global variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskeyenv` (§11.27). 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*`.

```

1782 \cs_new_protected:Nn \__enumext_storing_exec:
1783 {
1784   \bool_set_true:N \l__enumext_store_active_bool
1785   \bool_set_true:N \l__enumext_check_answers_bool
1786   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1787   \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1788   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1789   {
1790     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1791     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1792   }
1793   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1794   {
1795     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1796     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1797   }
1798   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1799   {
1800     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1801     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1802   }
1803 }

```

(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  \cs_set_protected:Npn \__enumext_tmp:n #1
no-store   {
1804       \keys_define:nn { enumext / #1 }
1805       {
1806         \keys_define:nn { enumext / #1 }
1807         {
1808           check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1809           check-ans .initial:n = false,
1810           check-ans .value_required:n = true,
1811           no-store .code:n = {
1812             \bool_set_false:N \__enumext_check_answers_bool
1813             \bool_set_false:N \__enumext_check_ans_key_bool
1814           },
1815           no-store .value_forbidden:n = true,
1816         }
1817       }
1818     \clist_map_inline:nn
1819     {
1820       level-1, level-2, level-3, level-4, enumext*
1821     }
1822     { \__enumext_tmp:n {#1} }

```

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

11.23.5 Set-up check answer mechanism

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

```

1823 \cs_new_protected:Nn \__enumext_check_ans_active:
1824 {
1825   \tl_if_empty:NF \__enumext_store_name_tl
1826   {
1827     \bool_if:NT \__enumext_check_answers_bool
1828     {
1829       \__enumext_check_ans_level:
1830     }
1831   }
1832 }

```

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

```

1833 \cs_new_protected:Nn \__enumext_check_ans_level:
1834 {
1835   \int_case:nn { \__enumext_level_int }
1836   {
1837     { 1 }{
1838       \bool_lazy_all:nT
1839       {
1840         { \bool_if_p:N \g__enumext_starred_bool }
1841         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1842       }
1843       {
1844         \int_gdecr:N \g__enumext_item_number_int
1845       }
1846     }
1847     { 2 }{
1848       \int_gdecr:N \g__enumext_item_number_int
1849     }
1850     { 3 }{
1851       \int_gdecr:N \g__enumext_item_number_int
1852     }
1853     { 4 }{
1854       \int_gdecr:N \g__enumext_item_number_int
1855     }
1856   }

```


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.

```

1857     \int_case:nn { \l__enumext_level_h_int }
1858     {
1859         { 1 }{
1860             \bool_lazy_all:nT
1861             {
1862                 { \bool_if_p:N \g__enumext_standar_bool }
1863                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1864             }
1865             {
1866                 \int_gdecr:N \g__enumext_item_number_int
1867             }
1868         }
1869     }
1870 }

```

(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.

```

1871 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1872 {
1873     \bool_lazy_and:nnT
1874     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1875     { \bool_if_p:N \g__enumext_standar_bool }
1876     {
1877         \bool_gset_true:N \g__enumext_check_ans_key_bool
1878     }
1879     \bool_lazy_and:nnT
1880     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1881     { \bool_if_p:N \g__enumext_starred_bool }
1882     {
1883         \bool_gset_true:N \g__enumext_check_ans_key_bool
1884     }
1885 }

```

(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:`.

```

1886 \cs_new_protected:Nn \__enumext_item_answer_diff:
1887 {
1888     \int_gset:Nn \g__enumext_item_answer_diff_int
1889     {
1890         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1891     }
1892 }

```

(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:`.

```

1893 \cs_new_protected:Nn \__enumext_check_ans_show:
1894 {
1895     \int_case:nn { \g__enumext_item_answer_diff_int }
1896     {
1897         { -1 }{ \__enumext_check_ans_msg_less: }
1898         { 0 }{ \__enumext_check_ans_msg_same_ok: }
1899         { 1 }{ \__enumext_check_ans_msg_greater: }
1900     }
1901 }
1902 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1903 {
1904     \msg_warning:nnee { enumext } { item-less-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_same_ok:
1908   {
1909     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1910     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1911   }
1912   \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1913   {
1914     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1915     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1916   }

```

(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:`.

```

1917 \cs_new_protected:Nn \__enumext_check_ans_log:
1918 {
1919   \int_case:nn { \g__enumext_item_answer_diff_int }
1920   {
1921     { -1 } { \__enumext_check_ans_log_msg_less: }
1922     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1923     { 1 } { \__enumext_check_ans_log_msg_greater: }
1924   }
1925 }
1926 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1927 {
1928   \msg_log:nneee { enumext } { item-less-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_same_ok:
1932 {
1933   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1934   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1935 }
1936 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1937 {
1938   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1939   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1940 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.23.6 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.

```

1941 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1942 {
1943   \int_compare:nNtT
1944   { \g__enumext_check_starred_cmd_int } = { 0 }
1945   {
1946     \msg_warning:nnnV
1947     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1948   }
1949   \int_compare:nNtT
1950   { \g__enumext_check_starred_cmd_int } > { 1 }
1951   {
1952     \msg_warning:nnnV
1953     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1954   }
1955   \int_gzero:N \g__enumext_check_starred_cmd_int
1956   \tl_clear:N \l__enumext_check_start_line_env_tl
1957 }

```

(End of definition for __enumext_check_starred_cmd:n.)

11.24 Writing .log, executing check-ans key and more

`__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, undefine the environment `anskeyenv` (§11.27) through the function `__enumext_undefine_anskey_env:` and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

1958 \cs_new_protected:Nn \__enumext_execute_after_env:
1959 {
1960   \int_compare:nNtT { \__enumext_level_int } = { 0 }
1961   {
1962     \tl_if_empty:NF \__enumext_store_name_tl
1963     {
1964       \__enumext_stop_save_ans_msg:
1965       \__enumext_item_answer_diff:
1966       \__enumext_log_global_vars:
1967       \__enumext_log_answer_vars:
1968       \bool_if:NTF \g__enumext_check_ans_key_bool
1969       {
1970         \__enumext_check_ans_show:
1971       }
1972       { \__enumext_check_ans_log: }
1973       \__enumext_undefine_anskey_env:
1974     }
1975     \__enumext_reset_global_vars:
1976   }
1977 }

```

• This function is passed to the function `__enumext_after_env:nn` for the environments `enumext` (§11.34) and `enumext*` (§11.37) and it is executed only when the environments are not nested or at some level of these..

(End of definition for `__enumext_execute_after_env:.`)

11.25 Keys and functions associated with storage

wrap-ans
wrap-opt
save-sep
mark-ans
mark-pos
show-ans
mark-ref
save-ref

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*`.

```

1978 \cs_set_protected:Npn \__enumext_tmp:n #1
1979 {
1980   \keys_define:nn { enumext / #1 }
1981   {
1982     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1983     wrap-ans .initial:n = \fbox{##1},
1984     wrap-ans .value_required:n = true,
1985     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1986     wrap-opt .initial:n = [{##1}],
1987     wrap-opt .value_required:n = true,
1988     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
1989     save-sep .initial:n = { , ~ },
1990     save-sep .value_required:n = true,
1991     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
1992     mark-ans .initial:n = \textasteriskcentered,
1993     mark-ans .value_required:n = true,
1994     mark-pos .choice:,
1995     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1996     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1997     %% mark-pos / unknown .code:n =
1998     %% \msg_error:nneee { enumext } { unknown-choice }
1999     %% { mark-pos } { left , right } { \exp_not:n {##1} },
2000     mark-pos .initial:n = right,
2001     mark-pos .value_required:n = true,
2002     show-ans .bool_set:N = \__enumext_show_answer_bool,
2003     show-ans .initial:n = false,
2004     show-ans .value_required:n = true,
2005     show-pos .bool_set:N = \__enumext_show_position_bool,
2006     show-pos .initial:n = false,
2007     show-pos .value_required:n = true,
2008     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,

```

```

2009         mark-ref .initial:n = \textasteriskcentered,
2010         mark-ref .value_required:n = true,
2011         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2012         save-ref .initial:n = false,
2013         save-ref .value_required:n = true,
2014     }
2015 }
2016 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {#1} }

```

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

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

```

2017 \cs_set_protected:Npn \l__enumext_tmp:n #1
2018 {
2019     \keys_define:nn { enumext / #1 }
2020     {
2021         mark-pos .choice:,
2022         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2023         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2024         mark-pos .initial:n = right,
2025         mark-pos .value_required:n = true,
2026         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2027         show-ans .initial:n = false,
2028         show-ans .value_required:n = true,
2029         show-pos .bool_set:N = \l__enumext_show_position_bool,
2030         show-pos .initial:n = false,
2031         show-pos .value_required:n = true,
2032     }
2033 }
2034 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

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

11.25.1 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⟩*.

```

2035 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2036 {
2037     \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2038     {
2039         \tl_clear:c { \l__enumext_store_save_key_ \__enumext_level: _tl }
2040         \tl_set:ce
2041             { \l__enumext_store_save_key_ \__enumext_level: _tl }
2042             { \__enumext_filter_save_key:n {#1} }
2043     }
2044 }
2045 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2046 {
2047     \bool_if:NF \l__enumext_store_save_key_vii_bool
2048     {
2049         \tl_clear:N \l__enumext_store_save_key_vii_tl
2050         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2051     }
2052 }

```

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

11.25.2 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.

```

2053 \cs_set_protected:Npn \__enumext_tmp:n #1
2054 {
2055   \keys_define:nn { enumext / enumext* }
2056   {
2057     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2058     save-key .value_required:n = true,
2059   }
2060   \keys_define:nn { enumext / #1 }
2061   {
2062     save-key .code:n = \__enumext_parse_save_key:n {##1},
2063     save-key .value_required:n = true,
2064   }
2065 }
2066 \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*`.

```

2067 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2068 {
2069   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2070   \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2071   \tl_set:ce
2072   { l__enumext_store_save_key_ \__enumext_level: _tl }
2073   { \__enumext_filter_save_key:n {#1} }
2074 }
2075 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2076 {
2077   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2078   \tl_clear:N \l__enumext_store_save_key_vii_tl
2079   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2080 }

```

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

11.25.3 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.

```

2081 \cs_new:Npn \__enumext_filter_save_key:n #1
2082 {
2083   \use:e
2084   {
2085     \keyval_parse:NNn
2086     \__enumext_filter_save_key_key:n
2087     \__enumext_filter_save_key_pair:nn {#1}
2088   }
2089 }

```

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.

```

2090 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2091 {
2092   \str_case:nnF {#1}
2093   {
2094     { resume } {} { resume* } {} { no-store } {}
2095   }
2096   { , { \exp_not:n {#1} } }
2097 }

```

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.

```

2098 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2099 {
2100   \str_case:nnF {#1}
2101   {
2102     { series } {} { resume } {} { save-ans } {}
2103     { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2104     { show-pos } {} { wrap-ans } {} { mark-ans } {} { wrap-opt } {}
2105     { save-sep } {} { mark-ref } {} { mini-env } {} { mini-sep } {}
2106     { mini-right } {} { mini-right* } {}
2107   }
2108   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2109 }

```

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

11.25.4 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.

```

2110 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2111 {
2112   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2113   {
2114     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2115   }
2116   { #1 }
2117 }
2118 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }

```

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

11.25.5 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.

```

2119 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2120 {
2121   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2122 }
2123 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }

```

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

11.25.6 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.

```

2124 \cs_new_protected:Nn \__enumext_store_level_open:
2125 {
2126   \bool_if:NT \l__enumext_check_answers_bool
2127   {
2128     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2129     {
2130       \__enumext_store_addto_seq:n
2131       {
2132         \item \begin{enumext}
2133       }
2134     }
2135     {
2136       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2137       {

```

```

2138         \item \begin{enumext} [
2139         ]
2140         \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2141         {
2142         ]
2143         }
2144         \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2145     }
2146 }
2147 }
2148 \cs_new_protected:Nn \__enumext_store_level_close:
2149 {
2150     \bool_if:NT \l__enumext_check_answers_bool
2151     {
2152         \__enumext_store_addto_seq:n { \end{enumext} }
2153     }
2154 }

```

(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.

```

2155 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2156 {
2157     \bool_if:NT \l__enumext_check_answers_bool
2158     {
2159         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2160         {
2161             \__enumext_store_addto_seq:n
2162             {
2163                 \item \mode_leave_vertical:
2164                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2165                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
2166             }
2167         }
2168         {
2169             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2170             {
2171                 \item \mode_leave_vertical:
2172                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2173                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
2174                 ]
2175             }
2176             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2177             {
2178             ]
2179             }
2180             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2181         }
2182     }
2183 }
2184 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2185 {
2186     \bool_if:NT \l__enumext_check_answers_bool
2187     {
2188         \__enumext_store_addto_seq:n { \end{enumext*} }
2189     }
2190 }

```

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

11.25.7 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`


```

2190 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2191 {
2192   \mode_leave_vertical:
2193   \skip_horizontal:n { -\dim_use:N #2 }
2194   \makebox[0pt][ r ]
2195   {
2196     \makebox[ \dim_use:N #1 ][ \__enumext_mark_position_str ]
2197     {
2198       \tl_use:N \__enumext_mark_answer_sym_tl
2199     }
2200   }
2201   \skip_horizontal:n { \dim_use:N #2 }
2202 }
2203 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

11.26 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*.

```

2204 \keys_define:nn { enumext / anskey }
2205 {
2206   break-col .bool_set:N = \__enumext_store_columns_break_bool,
2207   break-col .default:n = true,
2208   break-col .value_forbidden:n = true,
2209   item-join .int_set:N = \__enumext_store_item_join_int,
2210   item-join .value_required:n = true,
2211   item-star .bool_set:N = \__enumext_store_item_star_bool,
2212   item-star .default:n = true,
2213   item-star .value_forbidden:n = true,
2214   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
2215   item-sym* .value_required:n = true,
2216   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
2217   item-pos* .value_required:n = true,
2218 }

```

• 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 __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.

```

2219 \NewDocumentCommand \anskey { o +m }
2220 {
2221   \__enumext_anskey_safe_outer:
2222   \group_begin:
2223     \bool_if:NT \__enumext_check_answers_bool
2224     {
2225       \int_gincr:N \g__enumext_item_anskey_int
2226       \__enumext_anskey_safe_inner:n {#2}
2227       \__enumext_store_anskey_code:nn {#1} {#2}
2228     }
2229   \group_end:
2230 }

```

(End of definition for \anskey. This function is documented on page 11.)

11.26.1 Internal functions for the command

__enumext_anskey_safe_outer: 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.

__enumext_anskey_safe_inner:n

```

2231 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2232 {
2233   \bool_if:NF \__enumext_store_active_bool
2234   {

```

```

2235         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2236     }
2237     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2238     {
2239         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2240     }
2241     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2242     {
2243         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2244     }
2245 }

```

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.

```

2246 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2247 {
2248     \tl_if_empty:nT {#1}
2249     {
2250         \msg_error:nn { enumext } { anskey-empty-arg }
2251     }
2252     \int_incr:N \l__enumext_anskey_level_int
2253     \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2254     {
2255         \msg_error:nn { enumext } { anskey-nested }
2256     }
2257 }

```

(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.

```

2258 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2259 {
2260     \__enumext_store_addto_prop:n {#2}
2261     \bool_if:NT \l__enumext_store_ref_key_bool
2262     {
2263         \__enumext_store_internal_ref:
2264     }
2265     \__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`.

```

2266     \tl_if_novalue:nF {#1}
2267     {
2268         \keys_set:nn { enumext / anskey } {#1}
2269     }
2270     \tl_clear:N \l__enumext_store_anskey_arg_tl
2271     \bool_lazy_and:nnT
2272     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2273     { \bool_not_p:n { \l__enumext_starred_bool } }
2274     {
2275         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2276     }
2277     \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`.

```

2278     \bool_lazy_and:nnT
2279     { \bool_not_p:n { \l__enumext_starred_bool } }
2280     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2281     {
2282         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2283         {
2284             ( \exp_not:V \l__enumext_store_item_join_int )
2285         }
2286     }

```

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`.

```

2287 \bool_if:NTF \l__enumext_store_item_star_bool
2288 {
2289   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2290   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2291   {
2292     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2293     {
2294       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2295     }
2296   }
2297   \dim_compare:nT
2298   {
2299     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2300   }
2301   {
2302     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2303     {
2304       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2305     }
2306   }
2307   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2308 }
2309 {
2310   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2311 }

```

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`.

```

2312 \bool_lazy_and:nnT
2313 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2314 { \bool_if_p:N \l__enumext_hyperref_bool }
2315 {
2316   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2317   {
2318     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2319     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2320   }
2321 }
2322 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2323 }

```

(End of definition for `\l__enumext_store_anskey_code:nn`)

`\l__enumext_store_internal_ref:`

The function `\l__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`.

```

2324 \cs_new_protected:Nn \l__enumext_store_internal_ref:
2325 {
2326   \cs_set_protected:Npn \l__enumext_tmp:n ##1
2327   {
2328     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2329     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2330     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2331     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2332   }
2333   \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {##1} }
2334   \cs_set:Npn \l__enumext_tmp:n ##1
2335   { . \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.

```

2336 \bool_lazy_all:nT
2337 {
2338   { \bool_if_p:N \g__enumext_starred_bool }
2339   { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2340 }
2341 {

```

```

2342     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2343     { \tl_use:N \l__enumext_label_copy_vii_tl }
2344   }
2345   \bool_lazy_all:nT
2346   {
2347     { \bool_if_p:N \l__enumext_standar_bool }
2348     { \bool_if_p:N \g__enumext_starred_bool }
2349     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2350   }
2351   {
2352     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2353     {
2354       \tl_use:N \l__enumext_label_copy_vii_tl
2355       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2356     }
2357   }

```

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.

```

2358   \bool_lazy_all:nT
2359   {
2360     { \bool_if_p:N \l__enumext_standar_bool }
2361     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2362     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2363     { \bool_not_p:n { \l__enumext_starred_bool } }
2364   }
2365   {
2366     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2367     {
2368       \tl_use:N \l__enumext_label_copy_i_tl
2369       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2370     }
2371   }
2372   \cs_set:Npn \__enumext_tmp:n ##1
2373   { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2374   \bool_lazy_all:nT
2375   {
2376     { \bool_if_p:N \l__enumext_standar_bool }
2377     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2378     { \bool_not_p:n { \g__enumext_starred_bool } }
2379     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2380   }
2381   {
2382     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2383     {
2384       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2385       . \tl_use:N \l__enumext_label_copy_vii_tl
2386     }
2387   }

```

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

```

2388   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2389   {
2390     \l__enumext_store_name_tl \c_colon_str
2391     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2392   }

```

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.

```

2393   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2394   {
2395     \__enumext_newlabel:nn
2396     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2397     { \l__enumext_newlabel_arg_two_tl }
2398   }
2399   \l__enumext_store_write_aux_file_tl
2400 }

```

(End of definition for `__enumext_store_internal_ref:`)

_enumext_store_anskey_show_wrap:n

The function _enumext_store_anskey_show_wrap:n “wraps” the *⟨argument⟩* passed to _anskey when using the wrap-ans key.

```

2401 \\_cs_new_protected:Npn \\_enumext_store_anskey_show_wrap:n #1
2402 {
2403   \\par
2404   \\bool_if:NT \\l__enumext_starred_bool
2405   {
2406     \\cs_set:Nn \\_enumext_level: { vii }
2407   }
2408   \\_enumext_print_keyans_box:cc
2409   { \\l__enumext_labelwidth_ \\_enumext_level: _dim }
2410   { \\l__enumext_labelsep_ \\_enumext_level: _dim }
2411   \\_enumext_anskey_wrapper:n { #1 }
2412 }

```

(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.

```

2413 \\_cs_new_protected:Npn \\_enumext_store_anskey_show_left:n #1
2414 {
2415   \\bool_if:NT \\l__enumext_show_answer_bool
2416   {
2417     \\_enumext_store_anskey_show_wrap:n { #1 }
2418   }
2419   \\bool_if:NT \\l__enumext_show_position_bool
2420   {
2421     \\tl_set:Ne \\l__enumext_mark_answer_sym_tl
2422     {
2423       \\group_begin:
2424       \\exp_not:N \\normalfont
2425       \\exp_not:N \\footnotesize [ \\int_eval:n
2426       {
2427         \\prop_count:c { g__enumext_ \\_enumext_store_name_tl _prop }
2428       }
2429       ]
2430       \\group_end:
2431     }
2432     \\_enumext_store_anskey_show_wrap:n { #1 }
2433   }
2434 }

```

(End of definition for _enumext_store_anskey_show_left:n.)

11.27 The environment anskeyenv

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the **scontents** package, so to be able to have support at this point it is best to play a little with the internal code of **scontents** and *hooks*. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to *store-env*, if you are using the key *print-env* or the *write-out* key, sure, I can detect and modify it within the **enumext** and **enumext*** environments, but it would look strange not to have some keys available when running within these environments.
- A better (perhaps a bit paranoid) option is to define it within the environment in which the *save-ans* key is executed. and have it available only when that key is executed, here I would have absolute control of the keys and I make sure that *write-out* is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a fatal error, then the user will have to see how to solve the problem.

_enumext_undefine_anskey_env:

Detection of the **anskeyenv** environment outside the **enumext** and **enumext*** environments.

```

2435 \\_enumext_before_env:nn { enumext }
2436 {
2437   \\int_compare:nNnT { \\l__enumext_level_int } = { 0 }
2438   {
2439     \\cs_if_free:NF \\_scontents_anskeyenv_env_begin:
2440     {
2441       \\msg_fatal:nnn { enumext } { anskeyenv-defined } { anskeyenv }

```

```

2442     }
2443   }
2444 }
2445 \__enumext_before_env:nn { enumext* }
2446 {
2447   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
2448   {
2449     \cs_if_free:NF \__scontents_anskeyenv_env_begin:
2450     {
2451       \msg_fatal:nnn { enumext } { anskeyenv-defined } { anskeyenv }
2452     }
2453   }
2454 }

```

The function `__enumext_undefine_anskey_env:` will undefine the environment `anskeyenv` and will be passed to the function `__enumext_execute_after_env:` (§11.24) which is executed after the environment in which the key `save-ans` is active.

```

2455 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2456 {
2457   \cs_undefine:N \anskeyenv
2458   \cs_undefine:N \endanskeyenv
2459   \cs_undefine:N \__scontents_anskeyenv_env_begin:
2460   \cs_undefine:N \__scontents_anskeyenv_env_end:
2461 }

```

(End of definition for `__enumext_undefine_anskey_env:`.)

```

\__enumext_anskey_env_make:n
\__enumext_anskey_env_make:V
  anskeyenv
\__enumext_anskey_env_define_keys:
  \__enumext_rescan_anskey_env:n

```

The function `__enumext_anskey_env_make:n` creates the environment `anskeyenv` (custom version of `scontents` environment) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`. To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec:` (§11.23.1).

```

2462 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2463 {
2464   \newenvsc{anskeyenv}[store-env=#1,print-env=false]
2465   \__enumext_anskey_env_exec:
2466 }
2467 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }

```

The function `__enumext_anskey_env_define_keys:` will add the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` and will leave the keys `print-env`, `store-env` and `write-out` undefined. We will apply this function using the *hook* function `__enumext_before_env:nn`.

```

2468 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2469 {
2470   \keys_define:nn { scontents / scontents }
2471   {
2472     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2473     break-col .default:n = true,
2474     break-col .value_forbidden:n = true,
2475     item-join .int_gset:N = \g__enumext_store_item_join_int,
2476     item-join .value_required:n = true,
2477     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2478     item-star .default:n = true,
2479     item-star .value_forbidden:n = true,
2480     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2481     item-sym* .value_required:n = true,
2482     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2483     item-pos* .value_required:n = true,
2484     print-env .undefine:,
2485     store-env .undefine:,
2486     write-out .undefine:,
2487   }
2488 }

```

The function `__enumext_anskey_env_undefine_keys:` will leave the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` undefined. We will apply this function using the *hook* function `__enumext_after_env:nn`.

```

2489 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2490 {
2491   \keys_define:nn { scontents / scontents }
2492   {

```

```

2493         break-col .undefine:,
2494         item-join .undefine:,
2495         item-star .undefine:,
2496         item-sym* .undefine:,
2497         item-pos* .undefine:,
2498     }
2499 }

```

The function `__enumext_rescan_anskey_env:n` will be responsible for bringing the *body* of the environment saved in the sequence `\g__scontents_name_⟨store name⟩_seq` to pass it to our *sequence* and *prop list*.

```

2500 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2501 {
2502     \group_begin:
2503     \int_set:Nn \tex_newlinechar:D { `^^J }
2504     \__scontents_rescan_tokens:x
2505     {
2506         \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2507         #1
2508     }
2509 }

```

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

`__enumext_anskey_env_exec:` The function `__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our *keys*.

```

2510 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2511 {
2512     \__enumext_before_env:nn { anskeyenv }
2513     {
2514         \__enumext_anskey_env_define_keys:
2515     }

```

Now we will execute our actions after the `anskeyenv` environment is closed. We'll fetch the contents of the *environment body* that is now saved in `\g__scontents_name_⟨store name⟩_seq` and store it in the variable `\l__enumext_store_anskey_env_tl` then we execute the rest of the functions.

```

2516     \__enumext_after_env:nn { anskeyenv }
2517     {
2518         \tl_clear:N \l__enumext_store_anskey_env_tl
2519         \tl_clear:N \l__enumext_store_anskey_opt_tl
2520         \seq_gpop_right:cNT
2521         { \g__scontents_name_ \g__enumext_store_name_tl _seq } \l__enumext_store_anskey_env_tl
2522         { \seq_item:ce { \g__scontents_name_ \g__enumext_store_name_tl _seq } { -1 } }
2523         \__enumext_anskey_env_keys:
2524         \__enumext_anskey_env_store:
2525         \__enumext_anskey_env_clean:
2526         \__enumext_anskey_env_undefine_keys:
2527     }
2528 }

```

Using `\seq_gpop_right:cNT` is correct here, while we have control of the scope of the environment, we cannot know if a user is using the key `store-env={⟨store name⟩}` in some other environment. The last function `__enumext_anskey_env_undefine_keys:` is necessary so as not to hinder any `scontents` environment running within `enumext` or `enumext*`.

(End of definition for `__enumext_anskey_env_exec:.`)

`__enumext_anskey_env_keys:` The function `__enumext_anskey__env_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`.

```

2529 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2530 {
2531     \bool_lazy_and:nnT
2532     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2533     { \bool_not_p:n { \l__enumext_starred_bool } }
2534     {
2535         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2536     }

```

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

```

2537     \bool_lazy_and:nnT
2538     { \bool_not_p:n { \l__enumext_starred_bool } }

```



```

2539 { \int_compare:nNn { \g__enumext_store_item_join_int } > { 1 } }
2540 {
2541   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2542   {
2543     ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2544   }
2545 }

```

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

```

2546 \bool_if:NT \g__enumext_store_item_star_bool
2547 {
2548   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2549   {
2550     ,item-star,
2551   }
2552   \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2553   {
2554     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2555     {
2556       ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2557     }
2558   }
2559   \dim_compare:nT
2560   {
2561     \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2562   }
2563   {
2564     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2565     {
2566       ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2567     }
2568   }
2569 }
2570 }

```

The function `__enumext_anskey_env_store:` will be responsible for storing the content of the environment, we will execute the code within a group and only if the variable `\l__enumext_store_anskey_env_tl` is not empty using the function `\scontents_rescan_tokens:x` from package `scontents`.

```

2571 \cs_new_protected:Nn \__enumext_anskey_env_store:
2572 {
2573   \group_begin:
2574   \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2575   {
2576     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2577     {
2578       \exp_args:Ne
2579       \anskey{ \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl } }
2580     }
2581     {
2582       \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2583       \exp_args:Ne
2584       \anskey{ \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl } }
2585     }
2586   }
2587   \group_end:
2588 }

```

The function `__enumext_anskey_env_clean:` will return the global variables used by the `(keys)` to their initial state.

```

2589 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2590 {
2591   \bool_gset_false:N \g__enumext_store_columns_break_bool
2592   \int_gzero:N \g__enumext_store_item_join_int
2593   \bool_gset_false:N \g__enumext_store_item_star_bool
2594   \tl_gclear:N \g__enumext_store_item_symbol_tl
2595   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2596 }

```

(End of definition for `__enumext_anskey_env_keys:`, `__enumext_anskey_env_store:`, and `__enumext_anskey_env_clean:`.)

11.28 Common functions for keyans, keyans* and keyanspic

11.28.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the contents of the current $\langle label \rangle$ `__enumext_label_v_tl` for the `keyans` environment and the current $\langle label \rangle$ `__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 `__enumext_store_keyans_label_tl` variable, which will be passed to the $\langle prop list \rangle$ defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

2597 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2598 {
2599   \tl_clear:N \__enumext_store_keyans_label_tl
2600   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2601   {
2602     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_vi_tl }
2603   }
2604   {
2605     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_label_v_tl }
2606   }
2607   \tl_if_no_value:nF { #1 }
2608   {
2609     % Set save-sep
2610     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
2611     {
2612       \tl_put_right:Ne \__enumext_store_keyans_label_tl { \__enumext_store_keyans_item_opt_sep_tl }
2613     }
2614     \tl_put_right:Ne \__enumext_store_keyans_label_tl { #1 }
2615   }
2616   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
2617 }

```

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

11.28.2 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 $\langle label \rangle$. 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 $\langle labels \rangle$ and remove the dots “.” from them, we do not want to get double dots in our references.

```

2618 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2619 {
2620   \bool_if:NT \__enumext_store_ref_key_bool
2621   {
2622     \cs_set_protected:Npn \__enumext_tmp:n ##1
2623     {
2624       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2625       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2626       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2627       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2628     }
2629     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2630     \__enumext_keyans_store_ref_aux_i:
2631   }
2632 }

```

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

```

2633 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2634 {
2635   \bool_if:NT \g__enumext_starred_bool
2636   {
2637     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2638   }
2639   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2640   {
2641     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl

```

```

2642         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2643     }
2644     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2645     {
2646         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2647         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2648     }
2649     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2650     {
2651         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2652         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2653     }
2654     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2655     {
2656         \l__enumext_store_name_tl \c_colon_str
2657         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2658     }
2659     \__enumext_keyans_store_ref_aux_ii:
2660 }

```

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.

```

2661 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2662 {
2663     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2664     {
2665         \__enumext_newlabel:nn
2666         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2667         { \l__enumext_newlabel_arg_two_tl }
2668     }
2669     \l__enumext_store_write_aux_file_tl
2670 }

```

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

11.28.3 Storing content in sequence

`__enumext_keyans_addto_seq:n`
`__enumext_keyans_addto_seq_link:`

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

```

2671 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2672 {
2673     \tl_clear:N \l__enumext_store_keyans_label_tl
2674     \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
2675     {
2676         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2677     }
2678     {
2679         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2680     }
2681     \tl_if_novalue:nF { #1 }
2682     {
2683         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2684         {
2685             \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2686             {
2687                 \l__enumext_store_keyans_item_opt_sep_tl
2688             }
2689         }
2690         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2691     }
2692     \__enumext_keyans_addto_seq_link:
2693 }

```

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.

```

2694 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2695 {
2696   \bool_lazy_and:nnT
2697     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2698     { \bool_if_p:N \l__enumext_hyperref_bool }
2699     {
2700       \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2701       {
2702         \hfill \exp_not:N \hyperlink
2703         {
2704           \exp_not:V \l__enumext_newlabel_arg_one_tl
2705         }
2706         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2707       }
2708     }
2709     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2710     \bool_if:NT \l__enumext_check_answers_bool
2711     {
2712       \int_gincr:N \g__enumext_item_anskey_int
2713     }
2714 }

```

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

11.28.4 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.

```

2715 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2716 {
2717   \tl_if_novalue:nF { #1 }
2718   {
2719     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2720   }
2721   \bool_if:NT \l__enumext_show_answer_bool
2722   {
2723     \__enumext_keyans_show_ans:
2724   }
2725   \bool_if:NT \l__enumext_show_position_bool
2726   {
2727     \__enumext_keyans_show_pos:
2728   }
2729 }
2730 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2731 {
2732   \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2733   {
2734     \bool_lazy_or:nnT
2735       { \bool_if_p:N \l__enumext_show_answer_bool }
2736       { \bool_if_p:N \l__enumext_show_position_bool }
2737       {
2738         \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2739       }
2740   }
2741 }
2742 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2743 {
2744   \tl_put_left:Nn \l__enumext_label_v_tl
2745   {
2746     \__enumext_print_keyans_box:NN
2747     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2748   }
2749 }
2750 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2751 {
2752   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2753   {

```

```

2754     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2755     {
2756         \group_begin:
2757         \exp_not:N \normalfont
2758         \exp_not:N \footnotesize [ \int_eval:n
2759         {
2760             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2761         }
2762         ]
2763         \group_end:
2764     }
2765 }
2766 {
2767     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2768     {
2769         \group_begin:
2770         \exp_not:N \normalfont
2771         \exp_not:N \footnotesize [ \int_eval:n
2772         {
2773             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2774         }
2775         ]
2776         \group_end:
2777     }
2778 }
2779 \tl_put_left:Nn \l__enumext_label_v_tl
2780 {
2781     \__enumext_print_keyans_box:NN
2782     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2783 }
2784 }

```

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

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

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

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

```

item-pos*
2785 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2786 {
2787     \keys_define:nn { enumext / #1 }
2788     {
2789         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2790         item-sym* .value_required:n = true,
2791         item-sym* .initial:n = { $\star$ },
2792         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2793         item-pos* .value_required:n = true,
2794     }
2795 }
2796 \clist_map_inline:nn
2797 {
2798     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2799 }
2800 { \__enumext_tmp:nn #1 }

```

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

11.30 Redefining `\footnote` command

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

```

2801 \cs_new_protected:Nn \__enumext_footnotetext:nn
2802 {
2803     \footnotetext[#1]{#2}
2804 }
2805 \cs_new_protected:Nn \__enumext_renew_footnote:
2806 {
2807     \seq_gclear:N \g__enumext_footnote_arg_seq
2808     \seq_gclear:N \g__enumext_footnote_int_seq

```

```

2809 \RenewDocumentCommand \footnote { o +m }
2810 {
2811   \tl_if_novalue:nTF {##1}
2812   {
2813     \stepcounter{footnote}
2814     \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2815   }
2816   {
2817     \int_gset:Nn \g__enumext_footnote_int { ##1 }
2818   }
2819   \footnotemark [ \g__enumext_footnote_int ]
2820   \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2821   \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2822 }
2823 }
2824 \cs_new_protected:Nn \__enumext_print_footnote:
2825 {
2826   \seq_if_empty:NF \g__enumext_footnote_int_seq
2827   {
2828     \seq_map_pairwise_function:NNN
2829     \g__enumext_footnote_int_seq
2830     \g__enumext_footnote_arg_seq
2831     \__enumext_footnotetext:nn
2832   }
2833 }

```

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

11.31 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.

11.31.1 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: (§11.32)`.

```

2834 \cs_new_protected:Npn \__enumext_default_item:n #1
2835 {
2836   \tl_if_novalue:nTF {##1}
2837   {
2838     \bool_if:NT \l__enumext_check_answers_bool
2839     {
2840       \int_gincr:N \g__enumext_item_number_int
2841     }
2842     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2843     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2844   }
2845   {
2846     \bool_set_eq:cc
2847     { l__enumext_wrap_label_ \__enumext_level: _bool }
2848     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2849     \__enumext_item_std:w [##1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2850   }
2851 }

```

(End of definition for __enumext_default_item:n.)

`__enumext_starred_item:nn`

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

#1: \l__enumext_item_symbol_X_tl

#2: \l__enumext_item_symbol_sep_X_dim

First we will make a copy of `\l__enumext_item_symbol_#_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_#_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_#_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_#_bool` and the vars `\l__enumext_item_symbol_#_sep_#_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label:` (§11.32).

```

2852 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2853 {
2854   \tl_if_novalue:nF {#1}
2855   {
2856     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2857   }
2858   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2859   \tl_if_novalue:nTF {#2}
2860   {
2861     \dim_set_eq:cc
2862     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2863     { \__enumext_labelsep_ \__enumext_level: _dim }
2864   }
2865   {
2866     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2867   }
2868   \bool_if:NT \l__enumext_check_answers_bool
2869   {
2870     \int_gincr:N \g__enumext_item_number_int
2871   }
2872   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2873   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2874 }

```

(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 (§11.33.2).

```

2875 \cs_new_protected:Nn \__enumext_redefine_item:
2876 {
2877   \RenewDocumentCommand \item { s o o }
2878   {
2879     \bool_if:nTF {##1}
2880     {
2881       \__enumext_starred_item:nn {##2} {##3}
2882     }
2883     { \__enumext_default_item:n {##2} }
2884   }
2885 }

```

(End of definition for `__enumext_redefine_item:.`)

11.31.2 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`.

```

2886 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2887 {
2888   \tl_if_novalue:nTF { #1 }
2889   {
2890     \bool_set_true:N \__enumext_wrap_label_v_bool
2891     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```



```

2892     }
2893     {
2894         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2895         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2896     }
2897 }

```

(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>`.

```

2898 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2899 {
2900     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2901     \__enumext_keyans_show_left:n { #1 }
2902     \bool_set_true:N \l__enumext_wrap_label_v_bool
2903     \__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>`.

```

2904     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2905     \__enumext_keyans_addto_prop:n { #1 }
2906     \__enumext_keyans_store_ref:
2907     \__enumext_keyans_addto_seq:n { #1 }
2908     \int_gincr:N \g__enumext_check_starred_cmd_int
2909 }

```

(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 (§11.33.2).

```

2910 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2911 {
2912     \RenewDocumentCommand \item { s o }
2913     {
2914         \bool_if:nTF {##1}
2915         {
2916             \peek_remove_spaces:n
2917             {
2918                 \__enumext_keyans_starred_item:n {##2}
2919             }
2920         }
2921         {
2922             \__enumext_keyans_default_item:n {##2}
2923         }
2924     }
2925 }

```

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

11.32 Redefining `\makeLabel` command

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

11.32.1 Redefining `\makeLabel` for `enumext`

`__enumext_item_starred:`

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

```

2926 \cs_new_protected:Nn \__enumext_item_starred:
2927 {
2928     \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2929     {
2930         \mode_leave_vertical:
2931         \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }

```

```

2932         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2933         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2934     }
2935 }

```

(End of definition for __enumext_item_starred:.)

__enumext_make_label: The function __enumext_make_label: redefine \make_label for the enumext environment.

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext environment (§11.33.2).

```

2936 \cs_new_protected:Nn \__enumext_make_label:
2937 {
2938     \RenewDocumentCommand \make_label { m }
2939     {
2940         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2941         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2942         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2943         {
2944             \__enumext_item_starred:
2945             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2946         }
2947         { ##1 }
2948         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2949         \tl_gclear:N \g__enumext_item_symbol_tl
2950     }
2951 }

```

(End of definition for __enumext_make_label:.)

11.32.2 Redefining \make_label for keyans

__enumext_keyans_make_label: The function __enumext_keyans_make_label: redefine \make_label for keyans environment.

This function is passed to __enumext_list_arg_two_v: which is used in the definition of the keyans environment (§11.33.2).

```

2952 \cs_new_protected:Nn \__enumext_keyans_make_label:
2953 {
2954     \RenewDocumentCommand \make_label { m }
2955     {
2956         \tl_use:N \l__enumext_label_fill_left_v_tl
2957         \tl_use:N \l__enumext_label_font_style_v_tl
2958         \bool_if:NTF \l__enumext_wrap_label_v_bool
2959         {
2960             \__enumext_wrapper_label_v:n { ##1 }
2961         }
2962         { ##1 }
2963         \tl_use:N \l__enumext_label_fill_right_v_tl
2964     }
2965 }

```

(End of definition for __enumext_keyans_make_label:.)

11.33 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.

11.33.1 Calculation of \leftmargin and \itemindent

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

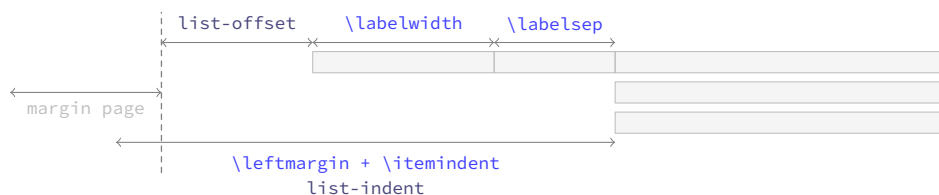
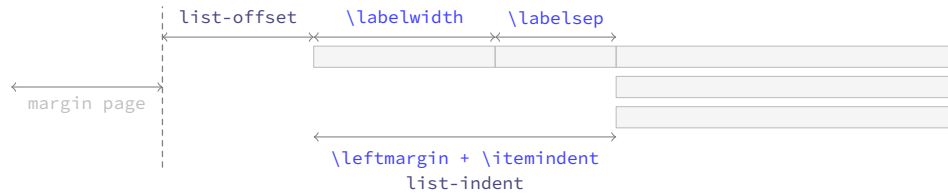
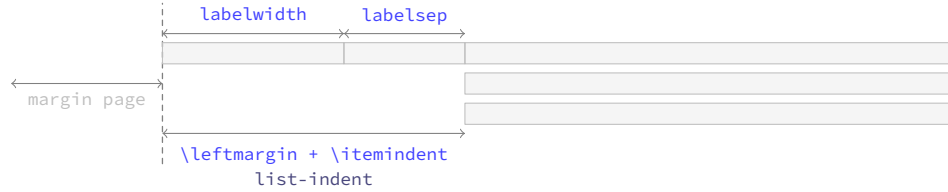


Figure 9: Representation of standard horizontal lengths in list environment.

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,

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

so that the left edge of the *label box* is at $\text{\leftmargin} + \text{\itemindent}$ minus $\text{\labelwidth} + \text{\labelsep}$. Thus, the handling of the margins by the package will be as shown in the figure 10. Where the default values will look like in the figure 11.

Figure 11: Default horizontal lengths in `enumext`.

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

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

```
#1: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__enumext_leftmargin_tmp_X_bool
```

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§11.33.2).

```
2966 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2967 {
2968   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2969   {
2970     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2971     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2972   }
2973   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2974   {
2975     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2976     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2977   }
2978 }
```

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

```
2978 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2979 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2980 {
2981   \dim_set:Nn #6 { #1 + #2 - #4 }
2982   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2983 }
2984 {
2985   \dim_compare:nNnT { #4 } = { #1 + #2 }
2986   { \dim_set:Nn #6 { \c_zero_dim } }
2987   \dim_compare:nNnT { #4 } < { #1 + #2 }
2988   { \dim_set:Nn #6 { #1 + #2 - #4 } }
2989   \dim_compare:nNnT { #4 } > { #1 + #2 }
2990   {
2991     \dim_set:Nn #6 { -#1 - #2 + #4 }
2992     \dim_set:Nn #6 { #6*-1 }
2993   }
2994   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2995 }
2996 }
2997 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

11.33.2 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.

```

2998 \cs_set_protected:Npn \__enumext_tmp:n #1
2999 {
3000   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3001   {
3002     \__enumext_calc_hspace:cccccc
3003     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3004     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3005     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3006     { l__enumext_leftmargin_tmp_#1_bool }
3007     \clist_map_inline:nn
3008       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3009       { \dim_set_eq:cc {###1} { l__enumext_###1_#1_dim } }
3010     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3011       { \skip_set_eq:cc {###1} { l__enumext_###1_#1_skip } }
3012     \usecounter { enumX#1 }
3013     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3014     \str_if_eq:nnTF {#1} { v }
3015     {
3016       \__enumext_keyans_redefine_item:
3017       \__enumext_keyans_make_label:
3018       \__enumext_keyans_ref:
3019       \__enumext_keyans_fake_item:
3020       \bool_if:cT { l__enumext_show_length_#1_bool }
3021       {
3022         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3023       }
3024     }
3025     {
3026       \__enumext_redefine_item:
3027       \__enumext_make_label:
3028       \__enumext_standar_ref:
3029       \__enumext_fake_item:
3030       \bool_if:cT { l__enumext_show_length_#1_bool }
3031       {
3032         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_int }
3033       }
3034     }
3035   }
3036 }
3037 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

3038 \cs_set_protected:Npn \__enumext_tmp:n #1
3039 {
3040   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3041   {
3042     \__enumext_calc_hspace:cccccc
3043     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3044     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3045     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3046     { l__enumext_leftmargin_tmp_#1_bool }
3047     \clist_map_inline:nn
3048       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3049       { \dim_set_eq:cc {###1} { l__enumext_###1_#1_dim } }
3050     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3051       { \skip_set_eq:cc {###1} { l__enumext_###1_#1_skip } }
3052     \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3053     \skip_zero:N \partopsep
3054     \usecounter { enumX#1 }
3055     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3056     \__enumext_starred_ref:
3057     \str_if_eq:nnTF {#1} { vii }

```

```

3058     {
3059         \__enumext_fake_item_vii:
3060         \bool_if:cT { l__enumext_show_length_vii_bool }
3061             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3062     }
3063     {
3064         \__enumext_fake_item_viii:
3065         \bool_if:cT { l__enumext_show_length_#1_bool }
3066             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3067     }
3068 }
3069 }
3070 \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:.)

11.34 The environment enumext

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

```

3071 \NewDocumentEnvironment{enumext}{0}{ }
3072 {
3073     \__enumext_safe_exec:
3074     \__enumext_parse_keys:n {#1}
3075     \__enumext_before_list:
3076     \__enumext_start_store_level:
3077     \__enumext_start_list:nn
3078     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3079     {
3080         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3081         \__enumext_before_keys_exec:
3082     }
3083     \__enumext_after_args_exec:
3084 }
3085 {
3086     \__enumext_stop_list:
3087     \__enumext_stop_store_level:
3088     \__enumext_after_list:
3089 }

```

(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:**.

```

3090 \cs_new_protected:Nn \__enumext_safe_exec:
3091 {
3092     \__enumext_internal_mini_page:
3093     \__enumext_is_not_nested:
3094     \int_incr:N \l__enumext_level_int
3095     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3096     { \msg_fatal:nn { enumext } { list-too-deep } }
3097     \bool_set_true:N \l__enumext_standar_bool
3098     \__enumext_is_on_first_level:
3099 }

```

(End of definition for __enumext_safe_exec:.)

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

```

3100 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3101 {
3102     \tl_if_novalue:nF {#1}
3103     {
3104         \str_clear:N \l__enumext_series_str

```

```

3105         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3106         {
3107             \keys_set:nn { enumext / level-1 } {#1}
3108             \__enumext_parse_series:n {#1}
3109         }
3110         {
3111             \exp_args:Ne \keys_set:nn
3112             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3113         }
3114         \__enumext_store_active_keys:n {#1}
3115     }
3116 }

```

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

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

```

3117 \cs_new_protected:Nn \__enumext_start_store_level:
3118 {
3119     \bool_lazy_all:nT
3120     {
3121         { \bool_if_p:N \l__enumext_store_active_bool }
3122         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3123         { \bool_not_p:n { \g__enumext_starred_bool } }
3124     }
3125     {
3126         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3127         {
3128             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3129             \__enumext_store_level_open:
3130         }
3131     }
3132     \bool_lazy_all:nT
3133     {
3134         { \bool_if_p:N \l__enumext_store_active_bool }
3135         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3136         { \bool_if_p:N \g__enumext_starred_bool }
3137     }
3138     {
3139         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3140         {
3141             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3142             \__enumext_store_level_open:
3143         }
3144     }
3145 }
3146 \cs_new_protected:Nn \__enumext_stop_store_level:
3147 {
3148     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3149     {
3150         \__enumext_store_level_close:
3151     }
3152 }

```

(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.

```

3153 \cs_new_protected:Nn \__enumext_before_list:
3154 {
3155     \__enumext_vspace_above:
3156     \__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.

```

3157     \__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.

```

3158 \dim_compare:nNt
3159 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3160 {
3161   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3162   {
3163     \linewidth
3164     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3165     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3166   }

```

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.

```

3167 \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3168 \int_gincr:N \g__enumext_minipage_stat_int
3169 \__enumext_mini_addvspace:
3170 \nointerlineskip\noindent
3171 \begin{__enumext_mini_env*}
3172 { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3173 }
3174 \__enumext_multicols_start:
3175 }

```

(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.

```

3176 \cs_new_protected:Nn \__enumext_multicols_start:
3177 {
3178   \int_compare:nNt
3179   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3180   {
3181     \dim_compare:nNt
3182     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3183     {
3184       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3185       {
3186         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3187         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3188         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3189         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3190       }
3191     }
3192     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3193     \skip_zero:N \multicolsep
3194     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3195     {
3196       \dim_zero:N \columnseprule
3197     }

```

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.

```

3198 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3199 {
3200   \__enumext_multi_addvspace:
3201 }
3202 \raggedcolumns

```



```

3203     \begin{multicols}{\int_use:c { \l__enumext_columns_ \l__enumext_level: _int } }
3204   }
3205 }

```

(End of definition for \l__enumext_multicols_start:.)

`\l__enumext_multicols_stop:` The function `\l__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.

```

3206 \cs_new_protected:Nn \l__enumext_multicols_stop:
3207 {
3208   \int_compare:nNtT
3209     { \int_use:c { \l__enumext_columns_ \l__enumext_level: _int } } > { 1 }
3210   {
3211     \end{multicols}
3212     \bool_if:cF { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3213     {
3214       \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \l__enumext_level: _skip } }
3215     }
3216   }
3217 }

```

(End of definition for \l__enumext_multicols_stop:.)

`\l__enumext_after_list:` The function `\l__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.

```

3218 \cs_new_protected:Nn \l__enumext_after_list:
3219 {
3220   \bool_if:cTF { \l__enumext_minipage_active_ \l__enumext_level: _bool }
3221   {
3222     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3223     {
3224       \msg_warning:nn { enumext } { missing-miniright }
3225       \miniright
3226     }
3227     \int_gzero:N \g__enumext_minipage_stat_int
3228     \end{__enumext_mini_env*}
3229     \par\addvspace { \l__enumext_minipage_after_skip }
3230   }
3231   { \l__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`.

```

3232   \l__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.

```

3233   \l__enumext_after_stop_list:
3234   \l__enumext_vspace_below:
3235   \bool_set_false:N \l__enumext_standar_bool
3236   \l__enumext_resume_save_counter:
3237 }

```

(End of definition for \l__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 `\l__enumext_execute_after_env:nn`.

```

3238 \l__enumext_after_env:nn {enumext} { \l__enumext_execute_after_env: }

```

11.35 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.

```

3239 \NewDocumentEnvironment{keyans}{0}{ }
3240 {
3241   \__enumext_keyans_safe_exec:
3242   \__enumext_keyans_parse_keys:n {#1}
3243   \__enumext_before_list_v:
3244   \__enumext_start_list:nn
3245     { \tl_use:N \__enumext_label_v_tl }
3246     {
3247       \__enumext_list_arg_two_v:
3248       \__enumext_before_keys_exec_v:
3249     }
3250   \__enumext_after_args_exec_v:
3251 }
3252 {
3253   \__enumext_check_starred_cmd:n { item }
3254   \__enumext_stop_list:
3255   \__enumext_after_list_v:
3256 }

```

(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.

```

3257 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3258 {
3259   \bool_if:NF \l__enumext_store_active_bool
3260   {
3261     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3262   }
3263   \int_incr:N \l__enumext_keyans_level_int
3264   \bool_set_true:N \l__enumext_keyans_env_bool
3265   \__enumext_keyans_save_start_line:
3266   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3267   \bool_set_false:N \l__enumext_store_active_bool
3268   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3269   {
3270     \msg_error:nn { enumext } { keyans-nested }
3271   }
3272   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3273   {
3274     \msg_error:nn { enumext } { keyans-wrong-level }
3275   }
3276 }

```

(End of definition for **__enumext_keyans_safe_exec:**)

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

```

3277 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3278 {
3279   \keys_set:nn { enumext / keyans } {#1}
3280 }

```

(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.

```

3281 \cs_new_protected:Nn \__enumext_before_list_v:
3282 {
3283   \__enumext_vspace_above_v:
3284   \__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.

```

3285   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3286   {
3287     \dim_set:Nn \l__enumext_minipage_left_v_dim

```

```

3288         {
3289             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3290         }

```

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*.

```

3291         \bool_set_true:N \l__enumext_minipage_active_v_bool
3292         \int_gincr:N \g__enumext_minipage_stat_int
3293         \__enumext_keyans_mini_addvspace:
3294         \nointerlineskip\noindent
3295         \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3296     }

```

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

```

3297     \__enumext_keyans_multicols_start:
3298 }

```

(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.

```

3299 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3300 {
3301     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3302     {

```

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

```

3303         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3304         {
3305             \dim_set:Nn \l__enumext_columns_sep_v_dim
3306             {
3307                 (
3308                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3309                 ) / \l__enumext_columns_v_int
3310                 - \l__enumext_listoffset_v_dim
3311             }
3312         }
3313         \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).

```

3314         \skip_zero:N \multicolsep
3315         \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.

```

3316         \bool_if:NF \l__enumext_minipage_active_v_bool
3317         {
3318             \__enumext_keyans_multi_addvspace:
3319         }
3320         \raggedcolumns
3321         \begin{multicols}{ \l__enumext_columns_v_int }
3322     }
3323 }

```

(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.

```

3324 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3325 {
3326     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3327     {
3328         \end{multicols}
3329         \bool_if:NF \l__enumext_minipage_active_v_bool

```

```

3330         {
3331         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3332         }
3333     }
3334 }

```

(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.

```

3335 \cs_new_protected:Nn \__enumext_after_list_v:
3336 {
3337     \bool_if:NTF \l__enumext_minipage_active_v_bool
3338     {
3339         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3340         {
3341             \msg_warning:nn { enumext } { missing-miniright }
3342             \miniright
3343         }
3344         \int_gzero:N \g__enumext_minipage_stat_int
3345         \end{\__enumext_mini_env*}
3346         \par\addvspace{ \l__enumext_minipage_after_skip }
3347     }
3348     { \__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.

```

3349     \bool_set_false:N \l__enumext_keyans_env_bool
3350     \__enumext_after_stop_list_v:
3351     \__enumext_vspace_below_v:
3352 }

```

(End of definition for `__enumext_after_list_v:`.)

11.36 The environment `keyanspic` and `\anspic`

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

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

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

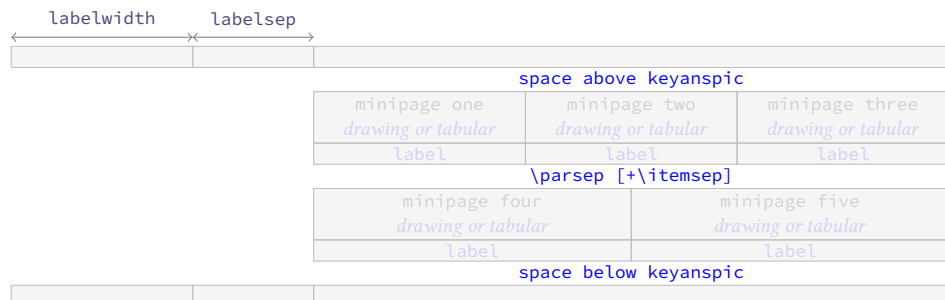


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

11.36.1 The command `\anspic`

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

```

3353 \NewDocumentCommand \anspic { s o +m }
3354 {

```

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

```

3355 \bool_if:NF \__enumext_store_active_bool
3356 {
3357   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3358 }
3359 \int_compare:nNt { \__enumext_level_int } > { 1 }
3360 {
3361   \msg_error:nn { enumext } { keyanspic-wrong-level }
3362 }
3363 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
3364 {
3365   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3366 }

```

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

```

3367 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3368 {
3369   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3370 }
3371 }

```

(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.

```

3372 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3373 {
3374   \stepcounter { enumXvi }
3375   #3 \\\
3376   \bool_if:nT { #1 }
3377   {
3378     \__enumext_keyans_addto_prop:n { #2 }
3379     \__enumext_keyans_store_ref:
3380     \__enumext_keyans_addto_seq:n { #2 }
3381     \int_gincr:N \__enumext_check_starred_cmd_int
3382     \bool_lazy_or:nnT
3383     { \bool_if_p:N \__enumext_show_answer_bool }
3384     { \bool_if_p:N \__enumext_show_position_bool }
3385     {
3386       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3387       \__enumext_keyans_show_left:n { #2 }
3388       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3389     }
3390   }
3391   \tl_use:N \__enumext_label_font_style_v_tl
3392   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3393 }

```

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

11.36.2 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.

```

3394 \NewDocumentEnvironment{keyanspic}{ o }
3395 {
3396   \__enumext_keyans_pic_safe_exec:
3397   \__enumext_start_list:nn
3398   { }
3399   {
3400     \__enumext_keyans_pic_arg_two:
3401   }

```

We apply the “adjusted” vertical spacing above the environment

```

3402 \vspace { \__enumext_keyans_pic_above_skip }
3403 }

```

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.

```

3404 {
3405   \tl_if_novalue:nTF { #1 }
3406   {
3407     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3408   }
3409   { \__enumext_keyans_pic_do:n { #1 } }
3410   \__enumext_stop_list:
3411   \__enumext_check_starred_cmd:n { anspic }
3412   \setcounter { enumXvi } { 0 }
3413   \vspace { \l__enumext_topsep_v_skip }
3414   %\bool_set_false:N \l__enumext_store_active_bool
3415 }

```

(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.

```

3416 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3417 {
3418   \int_incr:N \l__enumext_keyans_pic_level_int
3419   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3420   {
3421     \msg_error:nn { enumext } { keyanspic-nested }
3422   }
3423   \__enumext_keyans_save_start_line:
3424 }

```

(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`.

```

3425 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3426 {
3427   \dim_compare:nNnT { #1 } < { 0pt }
3428   { \skip_set:Nn #1 { -#1 } }
3429 }

```

(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.

```

3430 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3431 {

```

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.

```

3432   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3433   \__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.

```

3434   \skip_add:Nn \parsep { \itemsep }
3435   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3436   \dim_zero:N \labelwidth
3437   \dim_zero:N \listparindent
3438   \dim_zero:N \labelsep
3439   \skip_zero:N \partopsep
3440   \skip_zero:N \itemsep

```

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

```

3441   \__enumext_keyans_pic_skip_abs:N \parsep
3442   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3443   {

```

```

3444         \box_dp:N \strutbox
3445         + \l__enumext_topsep_v_skip
3446         - \parsep
3447     }
3448     \__enumext_item_std:w \scan_stop:
3449 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

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

```

3450 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3451 {
3452     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3453 }
3454 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

__enumext_keyans_pic_row:n

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

```

3455 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3456 {
3457     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3458     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3459     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3460     \int_step_inline:nnn
3461     { \l__enumext_keyans_pic_above_int + 1 }
3462     { \l__enumext_keyans_pic_below_int }
3463     {
3464         \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3465         \centering
3466         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3467         \__enumext_endminipage:
3468     }
3469     \par
3470 }

```

(End of definition for __enumext_keyans_pic_row:n.)

11.37 The environment `enumext*`

Generating horizontal list environments is NOT as simple as standard \TeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content passed to `\item`, we will also add the optional argument (*number*) 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 (*number*).

11.37.1 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.

```

3471 \cs_new_protected:Nn \enumext_starred_columns_set_vii:
3472 {
3473     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3474     {
3475         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3476         {
3477             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3478             / \l__enumext_columns_vii_int
3479         }
3480     }
3481     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3482     \dim_set:Nn \l__enumext_item_width_vii_dim
3483     {
3484         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )

```



```

3485         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3486         - \l__enumext_labelsep_vii_dim
3487     }
3488     \dim_zero_new:N \itemwidth
3489 }

```

(End of definition for \l__enumext_starred_columns_set_vii:.)

\l__enumext_starred_joined_item_vii:n

The function \l__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.

```

3490 \cs_new_protected:Npn \l__enumext_starred_joined_item_vii:n #1
3491 {
3492     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3493     \int_compare:nNtT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3494     {
3495         \msg_warning:nnee { enumext } { item-joined }
3496         { \int_use:N \l__enumext_joined_item_vii_int }
3497         { \int_use:N \l__enumext_columns_vii_int }
3498         \int_set:Nn \l__enumext_joined_item_vii_int
3499         {
3500             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3501         }
3502     }
3503     \int_compare:nNtT
3504     { \l__enumext_joined_item_vii_int }
3505     >
3506     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3507     {
3508         \msg_warning:nnee { enumext } { item-joined-columns }
3509         { \int_use:N \l__enumext_joined_item_vii_int }
3510         {
3511             \int_eval:n
3512             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3513         }
3514         \int_set:Nn \l__enumext_joined_item_vii_int
3515         {
3516             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3517         }
3518     }

```

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

```

3519     \int_compare:nNtTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3520     {
3521         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3522         \int_decr:N \l__enumext_joined_item_aux_vii_int
3523         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3524         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3525         \dim_set:Nn \l__enumext_joined_width_vii_dim
3526         {
3527             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3528             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3529                 + \l__enumext_columns_sep_vii_dim
3530             ) * \l__enumext_joined_item_aux_vii_int
3531         }
3532         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3533     }
3534     {
3535         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3536         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3537     }
3538 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n.)

\l__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.

```

3539 \cs_new_protected:Nn \l__enumext_start_mini_vii:
3540 {
3541     \dim_compare:nNtT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }

```

```

3542 {
3543   \dim_set:Nn \l__enumext_minipage_left_vii_dim
3544   {
3545     \linewidth
3546     - \l__enumext_minipage_right_vii_dim
3547     - \l__enumext_minipage_hsep_vii_dim
3548   }
3549   \bool_set_true:N \l__enumext_minipage_active_vii_bool
3550   \dim_gset_eq:NN
3551     \g__enumext_minipage_right_vii_dim
3552     \l__enumext_minipage_right_vii_dim
3553   \__enumext_mini_addvspace_vii:
3554   \nointerlineskip\noindent
3555   \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3556 }
3557 }

```

(End of definition for __enumext_start_mini_vii:.)

__enumext_stop_mini_vii: The function __enumext_stop_mini_vii: closes the __enumext_mini_env* environment on the left side, applies \hfill and sets the value of the variable \g__enumext_minipage_active_vii_bool to true which will be used in the function __enumext_after_star_env:nn to execute the __enumext__mini_env* on the “right side”.

```

3558 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3559 {
3560   \bool_if:NT \l__enumext_minipage_active_vii_bool
3561   {
3562     \end{__enumext_mini_env*}
3563     \hfill
3564     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3565   }
3566 }

```

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”.

```

3567 \__enumext_after_env:nn {enumext*}
3568 {
3569   \bool_if:NT \g__enumext_minipage_active_vii_bool
3570   {
3571     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3572     \par\addvspace { \g__enumext_minipage_right_skip }
3573     \bool_if:NF \g__enumext_minipage_center_vii_bool
3574     {
3575       \centering
3576     }
3577     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3578     \end{__enumext_mini_env*}
3579     \par\addvspace{ \g__enumext_minipage_after_skip }
3580   }
3581   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3582   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3583   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3584   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3585   \bool_gset_false:N \g__enumext_starred_bool
3586 }

```

(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.

```

3587 \NewDocumentEnvironment{enumext*}{ o }
3588 {
3589   \__enumext_safe_exec_vii:
3590   \__enumext_parse_keys_vii:n {#1}
3591   \__enumext_before_list_vii:
3592   \__enumext_start_store_level_vii:
3593   \__enumext_start_list:nn { }
3594   {
3595     \__enumext_list_arg_two_vii:
3596     \__enumext_before_keys_exec_vii:

```

```

3597     }
3598     \__enumext_starred_columns_set_vii:
3599     \item[] \scan_stop:
3600     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3601     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3602   }
3603   {
3604     \__enumext_stop_item_tmp_vii:
3605     \__enumext_remove_extra_parsep_vii:
3606     \__enumext_stop_list:
3607     \__enumext_stop_store_level_vii:
3608     \__enumext_after_list_vii:
3609   }

```

(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`.

```

3610 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3611 {
3612   \__enumext_internal_mini_page:
3613   \__enumext_is_not_nested:
3614   \int_incr:N \l__enumext_level_h_int
3615   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3616   {
3617     \msg_error:nn { enumext } { nested }
3618   }
3619   \bool_set_true:N \l__enumext_starred_bool
3620   \__enumext_is_on_first_level:
3621 }

```

(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⟩`.

```

3622 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3623 {
3624   \tl_if_novalue:nF {#1}
3625   {
3626     \str_clear:N \l__enumext_series_str
3627     \keys_set:nn { enumext / enumext* } {#1}
3628     \__enumext_parse_series:n {#1}
3629     \__enumext_store_active_keys_vii:n {#1}
3630   }
3631 }

```

(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`.

```

3632 \cs_new_protected:Nn \__enumext_before_list_vii:
3633 {
3634   \__enumext_vspace_above_vii:
3635   \__enumext_check_ans_active:
3636   \__enumext_before_args_exec_vii:
3637   \__enumext_start_mini_vii:
3638 }

```

(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.

```

3639 \cs_new_protected:Nn \__enumext_after_list_vii:
3640 {

```

```

3641     \__enumext_stop_mini_vii:
3642     \__enumext_after_stop_list_vii:
3643     \__enumext_check_ans_key_hook:
3644     \__enumext_vspace_below_vii:
3645     \bool_set_false:N \__enumext_starred_bool
3646     \__enumext_resume_save_counter:
3647 }

```

(End of definition for __enumext_after_list_vii:.)

```

\__enumext_start_store_level_vii:
\__enumext_stop_store_level_vii:

```

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

```

3648 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3649 {
3650     \bool_if:NT \__enumext_store_active_bool
3651     {
3652         \int_compare:nNt { \__enumext_level_int } > { \c_zero_int }
3653         {
3654             \__enumext_store_level_open_vii:
3655         }
3656     }
3657 }
3658 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3659 {
3660     \bool_if:NT \__enumext_store_active_bool
3661     {
3662         \int_compare:nNt { \__enumext_level_int } > { \c_zero_int }
3663         {
3664             \__enumext_store_level_close_vii:
3665         }
3666     }
3667 }

```

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

11.37.2 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 __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.

```

3668 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3669 {
3670     \__enumext_stop_item_tmp_vii:
3671     \int_incr:N \__enumext_item_column_pos_vii_int
3672     \int_gincr:N \g__enumext_item_count_all_vii_int
3673     \__enumext_item_peek_args_vii:
3674 }

```

(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).

```

3675 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3676 {
3677     \peek_meaning:NTF (
3678     { \__enumext_joined_item_vii:w }
3679     { \__enumext_joined_item_vii:w (1) }
3680 }

```

(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.

```

3681 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)

```

```

3682 {
3683     \__enumext_starred_joined_item_vii:n {#1}
3684     \peek_meaning_remove:NTF *
3685     { \__enumext_starred_item_vii:w }
3686     { \__enumext_standar_item_vii:w }
3687 }

```

(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].

```

3688 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3689 {
3690     \bool_set_false:N \l__enumext_item_starred_vii_bool
3691     \peek_meaning:NTF [
3692     {
3693         \bool_set_eq:NN
3694         \l__enumext_wrap_label_vii_bool
3695         \l__enumext_wrap_label_opt_vii_bool
3696         \__enumext_start_item_vii:w
3697     }
3698     {
3699         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3700         \legacy_if_set_true:n { @noitemarg }
3701         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3702     }
3703 }

```

(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

```

3704 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3705 {
3706     \bool_set_true:N \l__enumext_item_starred_vii_bool
3707     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3708     \peek_meaning:NTF [
3709     { \__enumext_starred_item_vii_aux_i:w }
3710     { \__enumext_starred_item_vii_aux_ii:w }
3711 }
3712 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w {#1}
3713 {
3714     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3715     \__enumext_starred_item_vii_aux_ii:w
3716 }
3717 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3718 {
3719     \peek_meaning:NTF [
3720     { \__enumext_starred_item_vii_aux_iii:w }
3721     {
3722         \dim_set_eq:NN
3723         \l__enumext_item_symbol_sep_vii_dim
3724         \l__enumext_labelsep_vii_dim
3725         \legacy_if_set_true:n { @noitemarg }
3726         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3727     }
3728 }
3729 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w {#1}
3730 {
3731     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3732     \legacy_if_set_true:n { @noitemarg }
3733     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3734 }

```

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

11.37.3 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.

```

3735 \cs_new_protected_nopar:Npn __enumext_start_item_vii:w [#1]
3736 {
3737   \cs_set_eq:NN __enumext_stop_item_tmp_vii: __enumext_stop_item_vii:
3738   \legacy_if:nT { @noitemarg }
3739   {
3740     \legacy_if_set_false:n { @noitemarg }
3741     \legacy_if:nT { @nmbrrlist }
3742     {
3743       \bool_if:NT \__enumext_hyperref_bool
3744       {
3745         \legacy_if_set_true:n { @hyper@item }
3746       }
3747       \refstepcounter{enumXvii}
3748       \bool_if:NT \__enumext_check_answers_bool
3749       {
3750         \int_gincr:N \g__enumext_item_number_int
3751       }
3752     }
3753   }

```

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

```

3754   \group_begin:
3755   \lrbox{ \l__enumext_item_text_vii_box }
3756   \bool_if:NF \l__enumext_footnotes_key_bool
3757   {
3758     \__enumext_renew_footnote:
3759   }
3760   \bool_if:NT \l__enumext_item_starred_vii_bool
3761   {
3762     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3763     {
3764       \tl_gset_eq:NN
3765       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3766     }
3767     \mode_leave_vertical:
3768     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3769     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3770     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3771     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3772   }
3773   \group_begin:
3774   \tl_use:N \l__enumext_label_font_style_vii_tl
3775   \bool_if:NTF \l__enumext_wrap_label_vii_bool
3776   {
3777     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3778     { \__enumext_wrapper_label_vii:n {#1} }
3779   }
3780   {
3781     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3782   }
3783   \group_end:
3784   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3785   \tl_use:N \l__enumext_after_list_args_vii_tl
3786   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3787   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3788   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip

```

```

3789         \tl_use:N \l__enumext_fake_item_indent_vii_tl
3790     }

```

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

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

```

3791 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3792 {
3793     \__enumext_endminipage:
3794     \endlrbox
3795     \group_end:
3796     \box_set_wd:Nn \l__enumext_item_text_vii_box
3797     {
3798         \l__enumext_joined_width_vii_dim
3799         + \l__enumext_labelwidth_vii_dim
3800         + \l__enumext_labelsep_vii_dim
3801     }
3802     \int_set:Nn \hbadness { 10000 }
3803     \box_use:N \l__enumext_item_text_vii_box
3804     \bool_if:NF \l__enumext_footnotes_key_bool
3805     {
3806         \__enumext_print_footnote:
3807     }
3808     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3809     {
3810         \par\noindent
3811         \int_zero:N \l__enumext_item_column_pos_vii_int
3812     }
3813     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3814 }

```

(End of definition for `__enumext_stop_item_vii:.`)

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

```

3815 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3816 {
3817     \int_compare:nNnT
3818     {
3819         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3820     }
3821     =
3822     { \c_zero_int }
3823     {
3824         \par
3825         \vspace{ -\l__enumext_itemsep_vii_skip }
3826         \int_gzero:N \g__enumext_item_count_all_vii_int
3827     }
3828 }

```

(End of definition for `__enumext_remove_extra_parsep_vii:.`)

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

```

3829 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

11.38 The environment `keyans*`

11.38.1 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.

```

3830 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3831 {
3832     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3833     {
3834         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3835         {
3836             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3837             / \l__enumext_columns_viii_int

```



```

3838     }
3839   }
3840   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3841   \dim_set:Nn \l__enumext_item_width_viii_dim
3842   {
3843     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3844     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3845     - \l__enumext_labelsep_viii_dim
3846   }
3847   \dim_zero_new:N \itemwidth
3848 }

```

(End of definition for \l__enumext_starred_columns_set_viii:.)

\l__enumext_starred_joined_item_viii:n

The function \l__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.

```

3849 \cs_new_protected:Npn \l__enumext_starred_joined_item_viii:n #1
3850 {
3851   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3852   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3853   {
3854     \msg_warning:nnee { enumext } { item-joined }
3855     { \int_use:N \l__enumext_joined_item_viii_int }
3856     { \int_use:N \l__enumext_columns_viii_int }
3857     \int_set:Nn \l__enumext_joined_item_viii_int
3858     {
3859       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3860     }
3861   }
3862   \int_compare:nNnT
3863   { \l__enumext_joined_item_viii_int }
3864   >
3865   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3866   {
3867     \msg_warning:nnee { enumext } { item-joined-columns }
3868     { \int_use:N \l__enumext_joined_item_viii_int }
3869     {
3870       \int_eval:n
3871       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3872     }
3873     \int_set:Nn \l__enumext_joined_item_viii_int
3874     {
3875       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3876     }
3877   }
3878   \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3879   {
3880     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3881     \int_decr:N \l__enumext_joined_item_aux_viii_int
3882     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3883     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3884     \dim_set:Nn \l__enumext_joined_width_viii_dim
3885     {
3886       \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3887       + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3888         + \l__enumext_columns_sep_viii_dim
3889         ) * \l__enumext_joined_item_aux_viii_int
3890     }
3891     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3892   }
3893   {
3894     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3895     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3896   }
3897 }

```

(End of definition for \l__enumext_starred_joined_item_viii:n.)

\l__enumext_start_mini_viii:

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

\l__enumext_stop_mini_viii:

```

3898 \cs_new_protected:Nn \l__enumext_start_mini_viii:
3899 {

```

```

3900 \dim_compare:nNt { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3901 {
3902   \dim_set:Nn \l__enumext_minipage_left_viii_dim
3903   {
3904     \linewidth
3905     - \l__enumext_minipage_right_viii_dim
3906     - \l__enumext_minipage_hsep_viii_dim
3907   }
3908   \bool_set_true:N \l__enumext_minipage_active_viii_bool
3909   \dim_gset_eq:NN
3910     \g__enumext_minipage_right_viii_dim
3911     \l__enumext_minipage_right_viii_dim
3912   \__enumext_mini_addvspace_viii:
3913   \nointerlineskip\noindent
3914   \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3915 }
3916 }
3917 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3918 {
3919   \bool_if:NT \l__enumext_minipage_active_viii_bool
3920   {
3921     \end{__enumext_mini_env*}
3922     \hfill
3923     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3924   }
3925 }
3926 \__enumext_after_env:nn {keyans*}
3927 {
3928   \bool_if:NT \g__enumext_minipage_active_viii_bool
3929   {
3930     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3931     \par\addvspace { \g__enumext_minipage_right_skip }
3932     \bool_if:NF \g__enumext_minipage_center_viii_bool
3933     {
3934       \centering
3935     }
3936     \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3937     \end{__enumext_mini_env*}
3938     \par\addvspace{ \g__enumext_minipage_after_skip }
3939   }
3940   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3941   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3942   \tl_gclear:N \g__enumext_miniright_code_viii_tl
3943   \dim_gzero:N \g__enumext_minipage_right_viii_dim
3944 }

```

(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.

```

3945 \NewDocumentEnvironment{keyans*}{ o }
3946 {
3947   \__enumext_safe_exec_viii:
3948   \__enumext_parse_keys_viii:n {#1}
3949   \__enumext_before_list_viii:
3950   \__enumext_start_list:nn { }
3951   {
3952     \__enumext_list_arg_two_viii:
3953     \__enumext_before_keys_exec_viii:
3954   }
3955   \__enumext_starred_columns_set_viii:
3956   \item[] \scan_stop:
3957   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3958   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3959 }
3960 {
3961   \__enumext_stop_item_tmp_viii:
3962   \__enumext_remove_extra_parsep_viii:
3963   \__enumext_check_starred_cmd:n { item }
3964   \__enumext_stop_list:

```

```

3965   \__enumext_after_list_viii:
3966   }

```

(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.

```

3967 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3968 {
3969   \int_incr:N \l__enumext_keyans_level_h_int
3970   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3971   {
3972     \msg_error:nn { enumext } { nested }
3973   }
3974   \__enumext_keyans_save_start_line:
3975   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3976   \bool_set_false:N \l__enumext_store_active_bool
3977   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3978   {
3979     \msg_error:nn { enumext } { keyans-wrong-level }
3980   }
3981 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

3982 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3983 {
3984   \tl_if_novalue:nF {#1}
3985   {
3986     \keys_set:nn { enumext / keyans* } {#1}
3987   }
3988 }

```

(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`.

```

3989 \cs_new_protected:Nn \__enumext_before_list_viii:
3990 {
3991   \__enumext_vspace_above_viii:
3992   \__enumext_before_args_exec_viii:
3993   \__enumext_start_mini_viii:
3994 }

```

(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.

```

3995 \cs_new_protected:Nn \__enumext_after_list_viii:
3996 {
3997   \__enumext_stop_mini_viii:
3998   \__enumext_after_stop_list_viii:
3999   \__enumext_vspace_below_viii:
4000 }

```

(End of definition for `__enumext_after_list_viii:`)

11.38.2 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`.

```
4001 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4002 {
4003   \__enumext_stop_item_tmp_viii:
4004   \int_incr:N \l__enumext_item_column_pos_viii_int
4005   \int_gincr:N \g__enumext_item_count_all_viii_int
4006   \__enumext_item_peek_args_viii:
4007 }
```

(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).

```
4008 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4009 {
4010   \peek_meaning:NTF (
4011     { \__enumext_joined_item_viii:w }
4012     { \__enumext_joined_item_viii:w (1) }
4013   }
```

(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`.

```
4014 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4015 {
4016   \__enumext_starred_joined_item_viii:n {#1}
4017   \peek_meaning_remove:NTF *
4018     { \__enumext_starred_item_viii:w }
4019     { \__enumext_standar_item_viii:w }
4020 }
```

(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]`.

```
4021 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4022 {
4023   \bool_set_false:N \l__enumext_item_starred_viii_bool
4024   \peek_meaning:NTF [
4025     {
4026       \bool_set_eq:NN
4027         \l__enumext_wrap_label_viii_bool
4028         \l__enumext_wrap_label_opt_viii_bool
4029       \__enumext_start_item_viii:w
4030     }
4031     {
4032       \bool_set_true:N \l__enumext_wrap_label_viii_bool
4033       \legacy_if_set_true:n { @noitemarg }
4034       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4035     }
4036   }
```

(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]`.

```

4037 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4038 {
4039   \bool_set_true:N \l__enumext_item_starred_viii_bool
4040   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4041   \peek_meaning:NTF [
4042     { \__enumext_starred_item_viii_aux_i:w }
4043     { \__enumext_starred_item_viii_aux_ii:w }
4044   ]

```

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`.

```

4045 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4046 {
4047   \tl_clear:N \l__enumext_store_keyans_label_tl
4048   \tl_if_no_value:nF { #1 }
4049   {
4050     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4051     {
4052       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4053       \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
4054     }
4055     \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
4056   }
4057   \__enumext_starred_item_viii_aux_ii:w
4058 }
4059 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4060 {
4061   \legacy_if_set_true:n { @noitemarg }
4062   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4063 }

```

(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.

```

4064 \cs_new_protected:Nn \__enumext_starred_item_exec:
4065 {
4066   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
4067   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
4068   \__enumext_keyans_store_ref:
4069   \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
4070   \__enumext_keyans_addto_seq_link:
4071   \int_gincr:N \g__enumext_check_starred_cmd_int
4072   \bool_if:NT \l__enumext_show_answer_bool
4073   {
4074     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4075   }
4076   \bool_if:NT \l__enumext_show_position_bool
4077   {
4078     \tl_set:Ne \l__enumext_mark_answer_sym_tl
4079     {
4080       \group_begin:
4081       \exp_not:N \normalfont
4082       \exp_not:N \footnotesize [ \int_eval:n
4083         {
4084           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4085         }
4086       ]
4087       \group_end:
4088     }
4089     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4090   }
4091 }

```

(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.

```

4092 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4093 {
4094   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4095   \legacy_if:nT { @noitemarg }
4096   {
4097     \legacy_if_set_false:n { @noitemarg }
4098     \legacy_if:nT { @nmbrrlist }
4099     {
4100       \bool_if:NT \l__enumext_hyperref_bool
4101       {
4102         \legacy_if_set_true:n { @hyper@item }
4103       }
4104       \refstepcounter{enumXviii}
4105     }
4106   }

```

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

```

4107   \group_begin:
4108   \lrbox{ \l__enumext_item_text_viii_box }
4109   \bool_if:NF \l__enumext_footnotes_key_bool
4110   {
4111     \__enumext_renew_footnote:
4112   }
4113   \bool_if:NT \l__enumext_item_starred_viii_bool
4114   {
4115     \__enumext_starred_item_exec:
4116   }
4117   \group_begin:
4118   \tl_use:N \l__enumext_label_font_style_viii_tl
4119   \bool_if:NTF \l__enumext_wrap_label_viii_bool
4120   {
4121     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4122     { \__enumext_wrapper_label_viii:n {#1} }
4123   }
4124   {
4125     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
4126   }
4127   \group_end:
4128   \skip_horizontal:N \l__enumext_labelsep_viii_dim
4129   \tl_use:N \l__enumext_after_list_args_viii_tl
4130   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4131   \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4132   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4133   \bool_if:NT \l__enumext_item_starred_viii_bool
4134   {
4135     \tl_use:N \l__enumext_fake_item_indent_viii_tl
4136     \__enumext_keyans_show_item_opt:
4137     \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4138   }
4139   {
4140     \tl_use:N \l__enumext_fake_item_indent_viii_tl
4141   }
4142 }

```

(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.

```

4143 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4144 {
4145   \__enumext_endminipage:
4146   \endlrbox
4147   \group_end:
4148   \box_set_wd:Nn \l__enumext_item_text_viii_box
4149   {
4150     \l__enumext_joined_width_viii_dim
4151     + \l__enumext_labelwidth_viii_dim
4152     + \l__enumext_labelsep_viii_dim

```

```

4153     }
4154     \int_set:Nn \hbadness { 10000 }
4155     \box_use:N \l__enumext_item_text_viii_box
4156     \bool_if:NF \l__enumext_footnotes_key_bool
4157     {
4158         \__enumext_print_footnote:
4159     }
4160     \int_compare:nNnTF
4161     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4162     {
4163         \par\noindent
4164         \int_zero:N \l__enumext_item_column_pos_viii_int
4165     }
4166     { \hspace{ \l__enumext_columns_sep_viii_dim } }
4167 }

```

(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.

```

4168 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4169 {
4170     \int_compare:nNnTF
4171     {
4172         \int_mod:nn
4173         { \g__enumext_item_count_all_viii_int }
4174         { \l__enumext_columns_viii_int }
4175     }
4176     =
4177     { \c_zero_int }
4178     {
4179         \par
4180         \vspace{ -\l__enumext_itemsep_viii_skip }
4181         \int_gzero:N \g__enumext_item_count_all_viii_int
4182     }
4183 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`)

11.39 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.

```

4184 \NewDocumentCommand \getkeyans { m }
4185 {
4186     \exp_args:Ne \__enumext_getkeyans_aux:n
4187     { \tl_to_str:e { \text_expand:n {#1} } }
4188 }

```

(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.

```

4189 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4190 {
4191     \str_if_in:nnTF {#1} { : }
4192     {
4193         \use:e
4194         {
4195             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4196             { {##1} {##2} }
4197         }
4198         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4199     }
4200     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4201 }

```

(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⟩*.

```

4202 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4203 {
4204   \prop_if_exist:cF { g__enumext_#1_prop }
4205   { \msg_error:nnn { enumext } { undefined-storage-anskey } { #1 } }
4206   \group_begin:
4207     \prop_item:cn { g__enumext_#1_prop }{#2}
4208   \group_end:
4209 }

```

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

11.40 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the *⟨sequence⟩* defined by the `save-ans` key. The first thing we will do is define a set of *⟨filtered keys⟩* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

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

```

4210 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4211 \keys_define:nn { enumext / print }
4212 {
4213   print* .code:n      = \keys_precompile:neN { enumext / enumext* }
4214                     { \__enumext_filter_save_key:n { #1 } }
4215                     \l__enumext_print_keyans_starred_tl, % starred cmd
4216   print* .initial:n   = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4217   print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4218                     { \__enumext_filter_save_key:n { #1 } }
4219                     \l__enumext_print_keyans_i_tl,
4220   print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4221   print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4222                     { \__enumext_filter_save_key:n { #1 } }
4223                     \l__enumext_print_keyans_ii_tl,
4224   print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4225   print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4226                     { \__enumext_filter_save_key:n { #1 } }
4227                     \l__enumext_print_keyans_iii_tl,
4228   print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4229   print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4230                     { \__enumext_filter_save_key:n { #1 } }
4231                     \l__enumext_print_keyans_iv_tl,
4232   print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4233   print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4234                     { \__enumext_filter_save_key:n { #1 } }
4235                     \l__enumext_print_keyans_vii_tl, % starred nested
4236   print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4237 }

```

- The reason for storing *⟨keys⟩* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its optional argument, except those related to the first opening level.

`\printkeyans` Create a user command to print “all stored content” in *⟨sequence⟩* for `\anskey`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `__enumext_printkeyans:nnn`.

```

4238 \NewDocumentCommand \printkeyans { s O{} m }
4239 {
4240   \group_begin:
4241     \tl_use:N \l__enumext_print_keyans_i_tl
4242     \tl_use:N \l__enumext_print_keyans_ii_tl
4243     \tl_use:N \l__enumext_print_keyans_iii_tl
4244     \tl_use:N \l__enumext_print_keyans_iv_tl
4245     \tl_use:N \l__enumext_print_keyans_vii_tl
4246     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }

```

```

4247   \group_end:
4248   }

```

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

```
\__enumext_printkeyans:nnn
```

The internal function `__enumext_printkeyans:nnn` will check for the existence of the `(sequence)`, if it does not exist it will return an error message, then it will check if not empty.

```

4249 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4250 {
4251   \seq_if_exist:cTF { g__enumext_#3_seq }
4252   {
4253     \seq_if_empty:cF { g__enumext_#3_seq }
4254     {
4255       %%\seq_show:c { g__enumext_#3_seq }

```

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

```

4256       \bool_if:nTF {#1}
4257       {
4258         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4259         {
4260           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4261         }
4262         {
4263           \tl_use:N \l__enumext_print_keyans_starred_tl
4264           \begin{enumext*}[#2]
4265             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4266             \end{enumext*}
4267           }
4268       }

```

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

```

4269       {
4270         \begin{enumext}[#2]
4271         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4272         \end{enumext}
4273       }
4274     }
4275   {
4276     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4277   }
4278 }
4279 }

```

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

11.41 The command `\setenumext`

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

```

4280 \keys_define:nn { enumext / meta-families }
4281 {
4282   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4283   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4284   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4285   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4286   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4287   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4288   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4289   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4290   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4291   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4292   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4293   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4294   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4295   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4296 }

```

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

```
4297 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4298 {
4299   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4300   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4301 }
```

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

```
4302 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4303 {
4304   \tl_if_novalue:nTF {#1}
4305   {
4306     \seq_map_inline:Nn \c__enumext_all_families_seq
4307   }
4308   {
4309     \seq_clear:N \l__enumext_setkey_tmpa_seq
4310     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4311     \int_set:Nn \l__enumext_setkey_tmpa_int
4312     {
4313       \seq_count:N \l__enumext_setkey_tmpb_seq
4314     }
4315     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4316     {
4317       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4318       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4319       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4320       {
4321         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4322       }
4323     }
4324     {
4325       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4326     }
4327     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4328     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4329     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4330   }
4331   {
4332     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4333   }
4334 }
```

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

`__enumext_set_parse:n`
`__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```
4335 \cs_new_protected:Npn \__enumext_set_parse:n #1
4336 {
4337   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4338   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } %<- max level
4339   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4340   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4341   {
4342     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4343     { \tl_trim_spaces:n {#1} }
4344   }
4345   { \__enumext_set_error:nn {#1} { } }
4346 }
4347 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4348 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

11.42 Messages

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

```
4349 \msg_new:nnn { enumext } { package-load }
4350 {
4351   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4352 }
4353 \msg_new:nnn { enumext } { package-not-load }
4354 {
```

```

4355     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4356   }
4357   \msg_new:nnn { enumext } { package-load-foot }
4358   {
4359     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4360   }

```

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

```

4361   \msg_new:nnn { enumext } { counters }
4362   {
4363     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4364     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4365   }

```

Message used by reserved **anskeyenv** environment by **enumext** package.

```

4366   \msg_new:nnn { enumext } { anskeyenv-defined }
4367   {
4368     The ~ '#1' ~ environment ~ is ~ reserved ~ by ~ \\
4369     'enumext'~ package, ~ it ~ cannot ~ be ~ continued.
4370   }

```

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

```

4371   \msg_new:nnn { enumext } { store-prop }
4372   {
4373     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4374   }
4375   \msg_new:nnn { enumext } { store-seq }
4376   {
4377     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4378   }
4379   \msg_new:nnn { enumext } { store-int }
4380   {
4381     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4382   }
4383   \msg_new:nnn { enumext } { prop-seq-int-hook }
4384   {
4385     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4386     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4387     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4388   }
4389   \msg_new:nnn { enumext } { item-answer-hook }
4390   {
4391     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4392     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4393     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4394   }

```

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

```

4395   \msg_new:nnn { enumext } { invalid-key }
4396   {
4397     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4398   }
4399   \msg_new:nnn { enumext } { unknown-key-family }
4400   {
4401     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4402   }

```

Messages used in length calculation.

```

4403   \msg_new:nnn { enumext } { width-negative }
4404   {
4405     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4406     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4407   }
4408   \msg_new:nnn { enumext } { width-zero }
4409   {
4410     Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4411     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4412   }

```

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

```

4413   \msg_new:nnn { enumext } { list-lengths }
4414   {
4415     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4416     \__enumext_show_length:nnn { dim } { labelsep } { #1}

```

```

4417     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4418     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4419     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4420     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4421     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4422     \__enumext_show_length:nnn { skip } { topsep } {#1}
4423     \__enumext_show_length:nnn { skip } { parsep } {#1}
4424     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4425     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4426     *****
4427 }

```

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

```

4428 \msg_new:nnn { enumext } { list-lengths-not-nested }
4429 {
4430     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4431     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4432     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4433     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4434     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4435     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4436     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4437     \__enumext_show_length:nnn { skip } { topsep } {#1}
4438     \__enumext_show_length:nnn { skip } { parsep } {#1}
4439     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4440     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4441     *****
4442 }

```

Messages used by `ref` key.

```

4443 \msg_new:nnn { enumext } { key-ref-empty }
4444 {
4445     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4446 }

```

Messages used by `save-ans` key.

```

4447 \msg_new:nnn { enumext } { save-ans-empty }
4448 {
4449     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4450 }
4451 \msg_new:nnn { enumext } { save-ans-log }
4452 {
4453     * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4454 }
4455 \msg_new:nnn { enumext } { save-ans-log-hook }
4456 {
4457     * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4458 }
4459 \msg_new:nnn { enumext } { save-ans-hook }
4460 {
4461     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4462 }

```

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

```

4463 \msg_new:nnn { enumext } { need-save-ans }
4464 {
4465     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4466 }
4467 \msg_new:nnn { enumext } { items-same-answer }
4468 {
4469     *****\\
4470     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str
4471     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\\
4472     *****
4473 }
4474 \msg_new:nnn { enumext } { item-greater-answer }
4475 {
4476     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4477     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\\
4478     Items ~ > ~ Answers.
4479 }

```

```

4480 \msg_new:nnn { enumext } { item-less-answer }
4481 {
4482   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4483   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4484   Items ~ < ~ Answers.
4485 }

```

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

```

4486 \msg_new:nnn { enumext } { missing-starred }
4487 {
4488   Missing ~ '\c_backslash_str #1*' ~ #2.
4489 }
4490 \msg_new:nnn { enumext } { many-starred }
4491 {
4492   Many ~ '\c_backslash_str #1*' ~ #2.
4493 }

```

Messages used by `\printkeyans*` command.

```

4494 \msg_new:nnn { enumext } { print-starred }
4495 {
4496   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4497   #2 ~ environment ~ \msg_line_context:.
4498 }

```

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

```

4499 \msg_new:nnn { enumext } { list-too-deep }
4500 {
4501   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
4502   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4503 }

```

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

```

4504 \msg_new:nnn { enumext } { anskey-empty-arg }
4505 {
4506   Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4507 }
4508 \msg_new:nnn { enumext } { anskey-wrong-place }
4509 {
4510   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
4511   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4512 }
4513 \msg_new:nnn { enumext } { anskey-nested }
4514 {
4515   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4516 }
4517 \msg_new:nnn { enumext } { anskey-nested-env }
4518 {
4519   The ~ environment ~ anskey* ~ can't ~ be ~ nested ~ \msg_line_context:.
4520 }
4521 \msg_new:nnn { enumext } { anspic-wrong-place }
4522 {
4523   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
4524   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4525 }
4526 \msg_new:nnn { enumext } { command-wrong-place }
4527 {
4528   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
4529   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4530 }

```

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

```

4531 \msg_new:nnn { enumext } { keyans-nested }
4532 {
4533   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4534 }
4535 \msg_new:nnn { enumext } { keyans-wrong-level }
4536 {
4537   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context::~ \\
4538   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4539 }
4540 \msg_new:nnn { enumext } { wrong-place }
4541 {
4542   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context::~ \\

```

```

4543     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
4544   }
4545   \msg_new:nnn { enumext } { keyanspic-nested }
4546   {
4547     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
4548   }
4549   \msg_new:nnn { enumext } { keyanspic-wrong-level }
4550   {
4551     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4552     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4553   }

```

Messages used by `\getkeyans` command.

```

4554   \msg_new:nnn { enumext } { undefined-storage-anskey }
4555   {
4556     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:~.
4557   }

```

Messages used by `\miniright` command.

```

4558   \msg_new:nnn { enumext } { missing-miniright }
4559   {
4560     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~\\
4561     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4562   }
4563   \msg_new:nnn { enumext } { wrong-miniright-place }
4564   {
4565     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4566     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4567   }
4568   \msg_new:nnn { enumext } { wrong-miniright-use }
4569   {
4570     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4571     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4572   }

```

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

```

4573   \msg_new:nnn { enumext } { nested }
4574   {
4575     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:~.
4576   }
4577   \msg_new:nnn { enumext } { item-joined }
4578   {
4579     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:~.
4580   }
4581   \msg_new:nnn { enumext } { item-joined-columns }
4582   {
4583     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:~.
4584   }

```

11.43 Finish package

Finish package implementation.

```

4585   \file_input_stop:
4586   </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>	216
<code>\+</code>	208
<code>\-</code>	208
<code>\\</code>	224, 2506, 3375, 4363, 4368, 4385, 4386, 4391, 4392, 4405, 4410, 4415, 4430, 4469, 4470, 4471, 4476, 4477, 4482, 4483, 4501, 4510, 4523, 4528, 4537, 4542, 4551, 4560, 4565, 4570
A	
<code>above</code>	1385
<code>above*</code>	1385
<code>\addvspace</code>	1039, 1067, 1183, 1262, 1325, 1331, 1359, 1376, 3214, 3229, 3331, 3346, 3572, 3579, 3931, 3938
<code>after</code>	878
<code>align</code>	495
<code>\Alph</code>	36, 41
<code>\Alph</code>	447, 556, 601, 669, 4232
<code>\alph</code>	36, 41
<code>\alph</code>	448, 554, 4224
<code>\anskey</code>	12, 72, 2219, 2579, 2584
<code>\anskeyenv</code>	2457
<code>anskeyenv</code>	13, 2462
<code>\anspic</code>	15, 96, 3353
<code>\anspic*</code>	66
<code>\arabic</code>	30, 36
<code>\arabic</code>	446, 553, 600, 4216, 4220, 4236
B	
<code>\baselineskip</code>	48
<code>\baselineskip</code>	2164, 2172
<code>before</code>	878
<code>before*</code>	878
<code>below</code>	1385
<code>below*</code>	1385
bool commands:	
<code>\bool_gset_false:N</code>	322, 323, 324, 2591, 2593, 3581, 3585, 3940
<code>\bool_gset_true:N</code>	236, 245, 981, 1877, 1883, 3564, 3582, 3923, 3941
<code>\bool_if:NTF</code>	387, 399, 416, 1407, 1421, 1434, 1445, 1456, 1467, 1478, 1489, 1542, 1559, 1564, 1572, 1599, 1637, 1642, 1649, 1653, 1675, 1680, 1688, 1695, 1726, 1734, 1827, 1968, 2037, 2047, 2126, 2150, 2157, 2185, 2223, 2233, 2261, 2287, 2404, 2415, 2419, 2546, 2620, 2635, 2710, 2721, 2725, 2838, 2868, 2942, 2958, 3020, 3030, 3060, 3065, 3148, 3198, 3212, 3220, 3259, 3316, 3329, 3337, 3355, 3560, 3569, 3573, 3650, 3660, 3743, 3748, 3756, 3760, 3775, 3804, 3919, 3928, 3932, 4072, 4076, 4100, 4109, 4113, 4119, 4133, 4156
<code>\bool_if:nTF</code>	1360, 1377, 2879, 2914, 2978, 3376, 4256
<code>\bool_if_p:N</code>	254, 268, 1706, 1707, 1715, 1716, 1840, 1862, 1874, 1875, 1880, 1881, 2272, 2313, 2314, 2338, 2347, 2348, 2360, 2376, 2532, 2697, 2698, 2735, 2736, 3121, 3134, 3136, 3383, 3384
<code>\bool_lazy_all:nTF</code>	252, 266, 1838, 1860, 2336, 2345, 2358, 2374, 3119, 3132
<code>\bool_lazy_and:nnTF</code>	232, 241, 1705, 1714, 1873, 1879, 2271, 2278, 2312, 2531, 2537, 2696
<code>\bool_lazy_or:nnTF</code>	1767, 1774, 2734, 3382
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 60, 70, 91, 96, 97, 102, 103, 106, 126, 129, 132, 133, 142, 143, 144, 152, 153, 167, 178, 180
<code>\bool_not_p:n</code>	233, 242, 2273, 2279, 2363, 2378, 2533, 2538, 3122, 3123, 3135
<code>\bool_set_eq:NN</code>	2846, 2894, 3693, 4026
<code>\bool_set_false:N</code>	396, 1812, 1813, 3235, 3267, 3349, 3414, 3432, 3645, 3690, 3976, 4023
<code>\bool_set_true:N</code>	259, 273, 378, 382, 488, 806, 1391, 1396, 1662, 1784, 1785, 2069, 2077, 2842, 2872, 2890, 2902, 3097, 3128, 3141, 3167, 3264, 3291, 3549, 3619, 3699, 3706, 3707, 3908, 4032, 4039, 4040
box commands:	
<code>\box_dp:N</code>	1079, 1083, 1087, 1098, 1102, 1113, 1122, 1128, 1138, 1151, 1157, 1163, 1194, 1195, 1196, 1199, 1209, 1213, 1222, 1229, 1234, 1242, 1271, 1272, 1275, 1282, 1295, 1303, 1309, 1317, 3444
<code>\box_new:N</code>	67, 173
<code>\box_set_wd:Nn</code>	3796, 4148
<code>\box_use:N</code>	3803, 4155
<code>\box_wd:N</code>	454
C	
<code>\c</code>	216, 217, 706, 708, 720, 722
<code>\catcode</code>	2506
<code>\cB</code>	217
<code>\cE</code>	217
<code>\centering</code>	1362, 1379, 3465, 3575, 3934
<code>check-ans</code>	1804
Document class:	
<code>article</code>	42
clist commands:	
<code>\clist_const:Nn</code>	185
<code>\clist_map_function:nN</code>	3452
<code>\clist_map_inline:Nn</code>	494, 748, 811, 877, 892, 973, 1401
<code>\clist_map_inline:nn</code>	45, 56, 75, 81, 93, 105, 131, 161, 184, 519, 536, 816, 987, 1507, 1751, 1818, 2016, 2034, 2066, 2333, 2629, 2796, 3007, 3010, 3037, 3047, 3050, 3070, 4338
<code>\columnbreak</code>	73
<code>\columnbreak</code>	2275
<code>columns</code>	957
<code>columns-sep</code>	957
<code>\columnsep</code>	92, 95
<code>\columnsep</code>	3192, 3313
<code>\columnseprule</code>	92, 95
<code>\columnseprule</code>	3196, 3315
Commands provide by enumext:	
<code>\anskey</code>	28, 63, 69, 70, 72, 74-76, 80, 82, 91, 103, 113, 114, 119
<code>\anspic*</code>	28, 29, 66, 70, 80, 81, 96, 98, 99, 113, 114
<code>\anspic</code>	70, 96, 98, 119
<code>\getkeyans</code>	70, 113, 120
<code>\item*</code>	28, 29, 66, 70, 80, 81, 84, 85, 104, 111, 113, 114
<code>\itemwidth</code>	99, 100, 106, 107
<code>\item</code>	84, 85, 100, 103-105, 107, 110
<code>\miniright</code>	27, 46, 53, 54, 92, 93, 95, 96, 120
<code>\printkeyans*</code>	114

122 / 133

```

\c__enumext_all_families_seq .. 116, 4297, 4306,
    4328
\__enumext_anskey__env_keys: ..... 78
\__enumext_anskey_env_clean: .. 79, 2525, 2529,
    2589
\__enumext_anskey_env_define_keys: 77, 2462,
    2468, 2514
\__enumext_anskey_env_exec: 78, 2465, 2510, 2510
\__enumext_anskey_env_keys: .. 2523, 2529, 2529
\__enumext_anskey_env_make:n 63, 77, 1787, 2462,
    2462, 2467
\__enumext_anskey_env_store: .. 79, 2524, 2529,
    2571
\__enumext_anskey_env_undefine_keys: . 77, 78,
    2489, 2526
\__enumext_anskey_env_undefine_keys:\__-
    enumext_rescan_anskey_env:n ..... 2462
\l__enumext_anskey_level_int .. 28, 2252, 2253
\__enumext_anskey_safe_inner:n .. 72, 73, 2226,
    2231, 2246
\__enumext_anskey_safe_outer: . 72, 2221, 2231,
    2231
\__enumext_anskey_wrapper:n ..... 1982, 2411
\__enumext_at_begin_document:n .. 33, 190, 190,
    347, 353
\__enumext_before_args_exec: 44, 893, 893, 3156
\__enumext_before_args_exec_v: .. 45, 909, 909,
    3284
\__enumext_before_args_exec_vii: .. 925, 925,
    3636
\__enumext_before_args_exec_viii: 929, 3992
\__enumext_before_env:nn 77, 194, 198, 2435, 2445,
    2512
\__enumext_before_keys_exec: 44, 893, 897, 3081
\__enumext_before_keys_exec_v: .. 45, 909, 913,
    3248
\__enumext_before_keys_exec_vii ..... 925
\__enumext_before_keys_exec_vii: 45, 933, 3596
\__enumext_before_keys_exec_viii: .. 45, 937,
    3953
\__enumext_before_list: ... 91, 3075, 3153, 3153
\__enumext_before_list_v: . 94, 3243, 3281, 3281
\__enumext_before_list_vii: ... 102, 3591, 3632,
    3632
\__enumext_before_list_viii: .. 109, 3949, 3989,
    3989
\l__enumext_before_no_starred_key_v_tl 915
\l__enumext_before_no_starred_key_vii-
    tl ..... 935
\l__enumext_before_no_starred_key_viii-
    tl ..... 939
\l__enumext_before_starred_key_v_tl ... 911
\l__enumext_before_starred_key_vii_tl . 927
\l__enumext_before_starred_key_viii_tl 931
\__enumext_calc_hspace:NNNNNNN 88, 2966, 2966,
    2997, 3002, 3042
\__enumext_check_ans_active: 64, 91, 1823, 1823,
    3157, 3635
\g__enumext_check_ans_item_tl ..... 81
\g__enumext_check_ans_key_bool 65, 66, 142, 322,
    1877, 1883, 1968
\l__enumext_check_ans_key_bool 65, 84, 85, 142,
    1808, 1813, 1874, 1880
\__enumext_check_ans_key_hook: 65, 1871, 1871,
    3232, 3643
\__enumext_check_ans_level: 64, 1823, 1829, 1833
\__enumext_check_ans_log: .. 65-67, 1917, 1917,
    1972
\__enumext_check_ans_log_msg_greater: 1917,
    1923, 1936
\__enumext_check_ans_log_msg_less: 1917, 1921,
    1926
\__enumext_check_ans_log_msg_same_ok: 1917,
    1922, 1931
\__enumext_check_ans_msg_greater: 1893, 1899,
    1912
\__enumext_check_ans_msg_less: 1893, 1897, 1902
\__enumext_check_ans_msg_same_ok: 1893, 1898,
    1907
\__enumext_check_ans_show: .. 65, 67, 1893, 1893,
    1970
\g__enumext_check_ans_show_bool ..... 93
\l__enumext_check_answers_bool 63, 64, 72, 142,
    1785, 1812, 1827, 2126, 2150, 2157, 2185, 2223, 2710,
    2838, 2868, 3748
\__enumext_check_starred_cmd:n 32, 66, 81, 1941,
    1941, 3253, 3411, 3963
\g__enumext_check_starred_cmd_int 142, 1944,
    1950, 1955, 2908, 3381, 4071
\l__enumext_check_start_line_env_tl 142, 287,
    294, 301, 1947, 1953, 1956
\l__enumext_columns_sep_v_dim 3303, 3305, 3313
\l__enumext_columns_sep_vii_dim .. 3473, 3475,
    3484, 3529, 3813
\l__enumext_columns_sep_viii_dim . 3832, 3834,
    3843, 3888, 4166
\l__enumext_columns_v_int 1190, 3301, 3309, 3321,
    3326
\l__enumext_columns_vii_int .. 3478, 3481, 3485,
    3493, 3497, 3500, 3506, 3512, 3516, 3808, 3819
\l__enumext_columns_viii_int . 3837, 3840, 3844,
    3852, 3856, 3859, 3865, 3871, 3875, 4161, 4174
\l__enumext_counter_i_tl ..... 41, 433
\l__enumext_counter_ii_tl ..... 41, 434
\l__enumext_counter_iii_tl ..... 41, 435
\l__enumext_counter_iv_tl ..... 41, 436
\c__enumext_counter_style_tl ..... 30, 46, 214
\g__enumext_counter_styles_tl . 27, 36, 64, 444,
    462
\l__enumext_counter_v_tl ..... 41, 437, 682
\l__enumext_counter_vi_tl ..... 41, 438
\l__enumext_counter_vii_tl ..... 41, 439, 612
\l__enumext_counter_viii_tl ..... 41, 440, 629
\l__enumext_current_widest_dim 27, 64, 468, 545,
    592, 663, 667
\__enumext_default_item:n ... 2834, 2834, 2883
\__enumext_define_counters:Nn 26, 424, 424, 433,
    434, 435, 436, 437, 438, 439, 440
\__enumext_endminipage: . 33, 353, 356, 368, 3467,
    3793, 4145
\g__enumext_envir_name_tl 31, 147, 260, 274, 330,
    1755, 1760, 1770, 1905, 1910, 1915, 1929, 1934, 1939
\__enumext_execute_after_env: 32, 33, 62, 65-67,
    77, 1958, 1958, 3238, 3829
\__enumext_fake_item: ..... 817, 817, 3029
\l__enumext_fake_item_indent_v_dim 836, 841
\l__enumext_fake_item_indent_v_tl 838, 2891,
    2895, 2903
\l__enumext_fake_item_indent_vii_dim 848, 853
\l__enumext_fake_item_indent_vii_tl 850, 3789

```

`\l__enumext_fake_item_indent_viii_dim` . 860, 865, 4137
`\l__enumext_fake_item_indent_viii_tl` .. 862, 4135, 4140
`\l__enumext_fake_item_indent_X_tl` 94
`__enumext_fake_item_vii:` 817, 845, 3059
`__enumext_fake_item_viii:` 817, 857, 3064
`__enumext_filter_save_key:n` .. 69, 2042, 2050, 2073, 2079, 2081, 2081, 4214, 4218, 4222, 4226, 4230, 4234
`__enumext_filter_save_key_key:n` .. 69, 2081, 2086, 2090
`__enumext_filter_save_key_pair:nn` 70, 2081, 2087, 2098
`__enumext_filter_series:n` 57, 1508, 1508, 1550, 1562, 1567
`__enumext_filter_series_key:n` 57, 1508, 1513, 1517
`__enumext_filter_series_pair:nn` .. 57, 1508, 1514, 1526
`\g__enumext_footnote_arg_seq` . 162, 2807, 2820, 2830
`\g__enumext_footnote_int` . 162, 2814, 2817, 2819, 2821
`\g__enumext_footnote_int_seq` . 162, 2808, 2821, 2826, 2829
`__enumext_footnotes_key_bool` 34
`\l__enumext_footnotes_key_bool` 29, 34, 105, 152, 382, 387, 396, 3756, 3804, 4109, 4156
`__enumext_footnotetext:nn` ... 2801, 2801, 2831
`__enumext_getkeyans:nn` .. 114, 4198, 4202, 4202
`__enumext_getkeyans_aux:n` 113, 4186, 4189, 4189
`\l__enumext_hyperref_bool` 29, 34, 152, 378, 399, 416, 2314, 2698, 3743, 4100
`__enumext_hypertarget:nn` 34, 373, 401, 405, 421
`__enumext_if_is_int:n` 206
`__enumext_if_is_int:nTF` 206, 701, 715
`__enumext_internal_mini_page:` .. 34, 358, 358, 3092, 3612
`__enumext_is_not_nested:` 31, 90, 226, 226, 3093, 3613
`__enumext_is_on_first_level:` . 31, 90, 226, 250, 3098, 3620
`\g__enumext_item_anskey_int` 72, 82, 142, 317, 344, 345, 1890, 2225, 2712
`__enumext_item_answer_diff:` 65–67, 1886, 1886, 1965
`\g__enumext_item_answer_diff_int` . 65, 66, 151, 318, 1888, 1895, 1919
`\l__enumext_item_column_pos_vii_int` 103, 3500, 3506, 3512, 3516, 3523, 3671, 3808, 3811
`\l__enumext_item_column_pos_viii_int` .. 110, 3859, 3865, 3871, 3875, 3882, 4004, 4161, 4164
`\l__enumext_item_column_pos_X_int` 165
`\g__enumext_item_count_all_vii_int` 103, 3524, 3672, 3819, 3826
`\g__enumext_item_count_all_viii_int` 110, 3883, 4005, 4173, 4181
`\g__enumext_item_count_all_X_int` 165
`\g__enumext_item_number_int` 64, 65, 142, 316, 343, 345, 1844, 1848, 1851, 1854, 1866, 1890, 2840, 2870, 3750
`__enumext_item_peek_args_vii:` 103, 3673, 3675, 3675
`__enumext_item_peek_args_viii:` .. 110, 4006, 4008, 4008
`__enumext_item_starred:` .. 86, 2926, 2926, 2944
`\l__enumext_item_starred_vii_bool` 3690, 3706, 3760
`\l__enumext_item_starred_viii_bool` 4023, 4039, 4113, 4133
`\l__enumext_item_starred_X_bool` 165
`__enumext_item_std:w` 33, 84–86, 98, 347, 351, 2843, 2849, 2873, 2891, 2895, 2903, 3448
`\g__enumext_item_symbol_aux_vii_tl` 3714, 3762, 3765, 3769, 3771
`\g__enumext_item_symbol_aux_X_tl` 165
`\l__enumext_item_symbol_sep_vii_dim` .. 3723, 3731, 3768, 3770
`\g__enumext_item_symbol_tl` 26, 85, 57, 2858, 2932, 2949
`\l__enumext_item_symbol_vii_tl` 3765
`\l__enumext_item_text_vii_box` 3755, 3796, 3803
`\l__enumext_item_text_viii_box` 4108, 4148, 4155
`\l__enumext_item_text_X_box` 165
`\l__enumext_item_width_vii_dim` ... 3482, 3527, 3535, 3536
`\l__enumext_item_width_viii_dim` .. 3841, 3886, 3894, 3895
`\l__enumext_item_width_X_dim` 165
`\l__enumext_itemindent_X_dim` 68
`\l__enumext_itemsep_vii_skip` 3825
`\l__enumext_itemsep_viii_skip` 4180
`\l__enumext_joined_item_aux_vii_int` .. 3521, 3522, 3523, 3524, 3530
`\l__enumext_joined_item_aux_viii_int` . 3880, 3881, 3882, 3883, 3889
`\l__enumext_joined_item_aux_X_int` 165
`__enumext_joined_item_vii:w` .. 103, 3678, 3679, 3681, 3681
`\l__enumext_joined_item_vii_int` .. 3492, 3493, 3496, 3498, 3504, 3509, 3514, 3519, 3521, 3527
`__enumext_joined_item_viii:w` . 110, 4011, 4012, 4014, 4014
`\l__enumext_joined_item_viii_int` . 3851, 3852, 3855, 3857, 3863, 3868, 3873, 3878, 3880, 3886
`\l__enumext_joined_item_X_int` 165
`\l__enumext_joined_width_vii_dim` . 3525, 3532, 3535, 3786, 3798
`\l__enumext_joined_width_viii_dim` 3884, 3891, 3894, 4130, 4150
`\l__enumext_joined_width_X_dim` 165
`__enumext_keyans_addto_prop:n` 80, 2597, 2597, 2905, 3378
`__enumext_keyans_addto_seq:n` . 81, 2671, 2671, 2907, 3380
`__enumext_keyans_addto_seq_link:` 2671, 2692, 2694, 4070
`__enumext_keyans_anspic_code:nnn` . 97, 3369, 3372, 3372
`__enumext_keyans_default_item:n` .. 85, 2886, 2886, 2922
`\l__enumext_keyans_env_bool` 34, 3122, 3135, 3264, 3349
`__enumext_keyans_fake_item:` .. 817, 833, 3019
`\l__enumext_keyans_item_opt_tl` 111, 106, 2719, 2732, 2738, 4055
`\l__enumext_keyans_level_h_int` .. 28, 622, 649,

2649, 3969, 3970
 \l__enumext_keyans_level_int .. 28, 1343, 2237, 2644, 3263, 3268, 3363
 __enumext_keyans_make_label: 37, 87, 2952, 2952, 3017
 __enumext_keyans_mini_addvspace: 52, 95, 1247, 1247, 3293
 __enumext_keyans_mini_right_cmd:n 54, 1345, 1368, 1368
 __enumext_keyans_mini_set_vskip: . 51, 1185, 1185, 1249
 __enumext_keyans_multi_addvspace: 95, 1041, 1052, 3318
 __enumext_keyans_multi_set_vskip: 48, 1041, 1041, 1054
 __enumext_keyans_multicols_start: 95, 3297, 3299, 3299
 __enumext_keyans_multicols_stop: . 95, 1372, 3324, 3324, 3348
 __enumext_keyans_parse_keys:n 3242, 3277, 3277
 \l__enumext_keyans_pic_above_int . 137, 3458, 3459, 3461
 \l__enumext_keyans_pic_above_skip .. 98, 137, 3402, 3442
 __enumext_keyans_pic_arg_two: 98, 3400, 3430, 3430
 \l__enumext_keyans_pic_below_int . 137, 3458, 3459, 3462
 \l__enumext_keyans_pic_body_seq .. 97-99, 137, 3367, 3407, 3466
 __enumext_keyans_pic_do:n 99, 3407, 3409, 3450, 3450, 3454
 \l__enumext_keyans_pic_level_int .. 28, 1335, 2241, 2600, 2639, 2674, 2752, 3418, 3419
 __enumext_keyans_pic_row:n 99, 3452, 3455, 3455
 __enumext_keyans_pic_safe_exec: .. 98, 3396, 3416, 3416
 __enumext_keyans_pic_skip_abs:N .. 98, 3425, 3425, 3441
 \l__enumext_keyans_pic_width_dim . 137, 3457, 3464
 __enumext_keyans_redefine_item: .. 86, 2910, 2910, 3016
 __enumext_keyans_ref: 40, 674, 692, 3018
 __enumext_keyans_ref:n 40, 671, 674, 674
 __enumext_keyans_safe_exec: . 3241, 3257, 3257
 __enumext_keyans_save_start_line: . 32, 281, 281, 3265, 3423, 3974
 __enumext_keyans_show_ans: .. 2715, 2723, 2742
 __enumext_keyans_show_item_opt: . 2715, 2730, 2903, 3392, 4136
 __enumext_keyans_show_left:n . 86, 2715, 2715, 2901, 3387
 __enumext_keyans_show_pos: .. 2715, 2727, 2750
 __enumext_keyans_starred_item:n .. 86, 2898, 2898, 2918
 __enumext_keyans_store_ref: .. 80, 2618, 2618, 2906, 3379, 4068
 __enumext_keyans_store_ref_aux_i: 80, 2618, 2630, 2633
 __enumext_keyans_store_ref_aux_ii: 81, 2618, 2659, 2661
 \l__enumext_keyans_tpa_tl . 28, 106, 2900, 2904
 __enumext_keyans_wrapper_opt:n .. 1985, 2738
 \l__enumext_label_copy_i_tl .. 2368, 2637, 2642, 2647, 2652
 \l__enumext_label_copy_v_tl 2647
 \l__enumext_label_copy_vi_tl 2642
 \l__enumext_label_copy_vii_tl 2343, 2354, 2385, 2637
 \l__enumext_label_copy_viii_tl 2652
 \l__enumext_label_copy_X_tl 154
 \l__enumext_label_fill_left_v_tl 2956
 \l__enumext_label_fill_left_X_tl 94
 \l__enumext_label_fill_right_v_tl 2963
 \l__enumext_label_fill_right_X_tl 94
 \l__enumext_label_font_style_v_tl 2957, 3391
 \l__enumext_label_font_style_vii_tl ... 3774
 \l__enumext_label_font_style_viii_tl .. 4118
 \l__enumext_label_i_tl 537
 \l__enumext_label_ii_tl 537
 \l__enumext_label_iii_tl 537
 \l__enumext_label_iv_tl 537
 __enumext_label_style:Nnn 26, 36, 457, 457, 472, 542, 589, 660, 664
 \l__enumext_label_v_tl .. 80, 81, 657, 2605, 2679, 2744, 2779, 2900, 2904, 3245, 3386, 3388
 \l__enumext_label_vi_tl . 80, 81, 657, 2602, 2676, 3386, 3388, 3392
 \l__enumext_label_vii_tl . 584, 3701, 3726, 3733
 \l__enumext_label_viii_tl 584, 4034, 4062, 4066
 \l__enumext_label_width_by_box .. 64, 453, 454
 __enumext_label_width_by_box:Nn 36, 451, 451, 456, 468, 725
 \l__enumext_labelsep_i_dim ... 2747, 2782, 4074, 4089
 \l__enumext_labelsep_v_dim 3308
 \l__enumext_labelsep_vii_dim . 3477, 3486, 3528, 3724, 3784, 3800
 \l__enumext_labelsep_viii_dim 3836, 3845, 3887, 4128, 4137, 4152
 \l__enumext_labelwidth_i_dim . 2747, 2782, 4074, 4089
 \l__enumext_labelwidth_v_dim 3308
 \l__enumext_labelwidth_vii_dim ... 3477, 3485, 3528, 3777, 3781, 3799
 \l__enumext_labelwidth_viii_dim .. 3836, 3844, 3887, 4121, 4125, 4151
 \l__enumext_leftmargin_tmp_v_bool . 98, 3432
 \l__enumext_leftmargin_tmp_X_bool 68
 \l__enumext_leftmargin_tmp_X_dim 68
 \l__enumext_leftmargin_X_dim 68
 __enumext_level: 202, 202, 566, 569, 570, 579, 581, 820, 824, 828, 895, 899, 903, 907, 990, 992, 994, 996, 1029, 1031, 1033, 1035, 1039, 1072, 1075, 1094, 1103, 1109, 1114, 1118, 1129, 1133, 1134, 1139, 1175, 1179, 1352, 1358, 1405, 1407, 1409, 1412, 1419, 1421, 1423, 1426, 2037, 2039, 2041, 2069, 2070, 2072, 2128, 2136, 2140, 2144, 2406, 2409, 2410, 2842, 2843, 2847, 2848, 2849, 2856, 2858, 2862, 2863, 2866, 2872, 2873, 2928, 2931, 2933, 2940, 2941, 2942, 2945, 2948, 3078, 3080, 3128, 3141, 3148, 3159, 3161, 3164, 3165, 3167, 3172, 3179, 3182, 3184, 3186, 3187, 3188, 3189, 3192, 3198, 3203, 3209, 3212, 3214, 3220
 \l__enumext_level_h_int .. 28, 234, 256, 269, 605, 642, 1841, 1857, 2362, 2379, 3614, 3615
 \l__enumext_level_int . 90, 28, 204, 243, 255, 270, 360, 1002, 1146, 1339, 1835, 1863, 1960, 2339, 2349,

2355, 2361, 2369, 2377, 2384, 2437, 2447, 3032, 3094,
 3095, 3105, 3112, 3126, 3139, 3194, 3272, 3359, 3652,
 3662, 3977
 __enumext_list_arg_two_i: 2998
 __enumext_list_arg_two_ii: 2998
 __enumext_list_arg_two_iii: 2998
 __enumext_list_arg_two_iv: 2998
 __enumext_list_arg_two_v: . 86, 2998, 3247, 3433
 __enumext_list_arg_two_vii: 3038, 3595
 __enumext_list_arg_two_viii: 3038, 3952
 \l__enumext_listoffset_v_dim 3310
 \l__enumext_listparindent_vii_dim 3787
 \l__enumext_listparindent_viii_dim ... 4131
 __enumext_log_answer_vars: . 33, 332, 340, 1967
 __enumext_log_global_vars: . 33, 332, 332, 1966
 __enumext_make_label: 37, 84, 85, 87, 2936, 2936,
 3027
 \l__enumext_mark_answer_sym_tl . 71, 132, 1991,
 2198, 2421, 2754, 2767, 4078
 \l__enumext_mark_position_str 132, 1995, 1996,
 2022, 2023, 2196
 \l__enumext_mark_ref_sym_tl .. 132, 2008, 2319,
 2706
 __enumext_mini_addvspace: .. 51, 92, 1168, 1168,
 3169
 __enumext_mini_addvspace_vii: 53, 1321, 1321,
 3553
 __enumext_mini_addvspace_viii: 53, 1321, 1327,
 3912
 __enumext_mini_env* 358
 __enumext_mini_right_cmd:n 54, 1347, 1349, 1349
 __enumext_mini_set_vskip: . 49, 1069, 1069, 1170
 __enumext_mini_set_vskip_vii: 52, 1264, 1264,
 1323
 __enumext_mini_set_vskip_viii: 52, 1264, 1286,
 1329
 __enumext_minipage:w 33, 353, 355, 364, 3464, 3786,
 4130
 \l__enumext_minipage_active_v_bool .. 95, 96,
 3291, 3316, 3329, 3337
 \g__enumext_minipage_active_vii_bool .. 101,
 3564, 3569, 3581
 \l__enumext_minipage_active_vii_bool . 3549,
 3560
 \g__enumext_minipage_active_viii_bool 3923,
 3928, 3940
 \l__enumext_minipage_active_viii_bool 3908,
 3919
 \g__enumext_minipage_active_X_bool ... 165
 \l__enumext_minipage_active_X_bool 82
 \g__enumext_minipage_after_skip 82, 1268, 1280,
 3579, 3938
 \l__enumext_minipage_after_skip 49, 50, 93, 96,
 82, 1085, 1100, 1120, 1136, 1151, 1157, 1163, 1177,
 1187, 1196, 1199, 1211, 1229, 1240, 1256, 1288, 1301,
 1315, 3229, 3346
 \g__enumext_minipage_center_vii_bool . 3573,
 3582
 \g__enumext_minipage_center_viii_bool 3932,
 3941
 \g__enumext_minipage_center_X_bool ... 165
 \l__enumext_minipage_hsep_v_dim ... 94, 3289
 \l__enumext_minipage_hsep_vii_dim 3547
 \l__enumext_minipage_hsep_viii_dim ... 3906
 \l__enumext_minipage_left_skip 49, 95, 82, 1077,
 1092, 1111, 1126, 1173, 1183, 1188, 1194, 1203, 1220,
 1232, 1252, 1262, 1266, 1271, 1275, 1289, 1293, 1307,
 1325, 1331
 \l__enumext_minipage_left_v_dim 94, 3287, 3295
 \l__enumext_minipage_left_vii_dim 3543, 3555
 \l__enumext_minipage_left_viii_dim 3902, 3914
 \l__enumext_minipage_left_X_dim 82
 \g__enumext_minipage_right_skip 82, 1267, 1272,
 1276, 3572, 3931
 \l__enumext_minipage_right_skip . 49, 82, 1081,
 1096, 1116, 1131, 1189, 1195, 1207, 1225, 1236, 1290,
 1297, 1311, 1359, 1376
 \l__enumext_minipage_right_v_dim .. 94, 1370,
 1375, 3285, 3289
 \g__enumext_minipage_right_vii_dim 100, 3551,
 3571, 3584
 \l__enumext_minipage_right_vii_dim 100, 3541,
 3546, 3552
 \g__enumext_minipage_right_viii_dim .. 3910,
 3930, 3943
 \l__enumext_minipage_right_viii_dim .. 3900,
 3905, 3911
 \g__enumext_minipage_right_X_dim 165
 \g__enumext_minipage_right_X_skip 165
 \g__enumext_minipage_stat_int . 92, 95, 82, 1364,
 1381, 3168, 3222, 3227, 3292, 3339, 3344
 \g__enumext_miniright_code_vii_tl 101, 3577,
 3583
 \g__enumext_miniright_code_viii_tl 3936, 3942
 \g__enumext_miniright_code_X_tl 165
 __enumext_multi_addvspace: . 48, 92, 1024, 1024,
 3200
 __enumext_multi_set_vskip: . 47, 988, 988, 1026
 \l__enumext_multicols_above_ii_skip ... 1007
 \l__enumext_multicols_above_iii_skip .. 1013
 \l__enumext_multicols_above_iv_skip ... 1019
 \l__enumext_multicols_above_v_skip 1043, 1057,
 1067
 \l__enumext_multicols_above_X_skip 76
 \l__enumext_multicols_below_v_skip 1047, 1061,
 3331
 \l__enumext_multicols_below_X_skip 76
 __enumext_multicols_start: 92, 3174, 3176, 3176
 __enumext_multicols_stop: 93, 1354, 3206, 3206,
 3231
 __enumext_newlabel:nn 29, 35, 75, 409, 409, 2395,
 2665
 \l__enumext_newlabel_arg_one_tl 29, 35, 75, 80,
 154, 2318, 2388, 2396, 2654, 2666, 2704
 \l__enumext_newlabel_arg_two_tl 29, 35, 74, 154,
 2342, 2352, 2366, 2382, 2397, 2641, 2646, 2651, 2667
 __enumext_parse_keys:n ... 58, 3074, 3100, 3100
 __enumext_parse_keys_vii:n 58, 3590, 3622, 3622
 __enumext_parse_keys_viii:n . 3948, 3982, 3982
 __enumext_parse_save_key:n 69, 2062, 2067, 2067
 __enumext_parse_save_key_vii:n 69, 2057, 2067,
 2075
 __enumext_parse_serie:n 102
 __enumext_parse_series:n .. 58, 90, 1538, 1538,
 3108, 3628
 __enumext_parse_store_keys:n 90
 \l__enumext_parsep_i_skip 1005, 1007, 1149, 1197
 \l__enumext_parsep_ii_skip ... 1011, 1013, 1155

```

\l__enumext_parsep_iii_skip .. 1017, 1019, 1161
\l__enumext_parsep_vii_skip ..... 3788
\l__enumext_parsep_viii_skip ..... 4132
\l__enumext_partopsep_v_skip . 1059, 1063, 1223,
    1227, 1234, 1238, 1254, 1258
\l__enumext_partopsep_viii_skip ..... 1299
\__enumext_phantomsection: 34, 373, 402, 406, 422
\__enumext_print_footnote: ... 2801, 2824, 3806,
    4158
\__enumext_print_keyans_box:NN 71, 2190, 2190,
    2203, 2408, 2746, 2781, 4074, 4089
\l__enumext_print_keyans_i_tl ... 4219, 4241
\l__enumext_print_keyans_ii_tl ... 4223, 4242
\l__enumext_print_keyans_iii_tl .. 4227, 4243
\l__enumext_print_keyans_iv_tl ... 4231, 4244
\l__enumext_print_keyans_starred_tl 114, 115,
    122, 4215, 4263
\l__enumext_print_keyans_vii_tl 114, 4235, 4245
\l__enumext_print_keyans_X_tl ..... 122
\__enumext_printkeyans:nnn 114, 115, 4246, 4249,
    4249
\__enumext_redefine_item: . 85, 2875, 2875, 3026
\l__enumext_ref_key_arg_tl 38, 46, 217, 559, 560,
    573, 604, 607, 618, 624, 635, 676, 677, 688
\l__enumext_ref_the_count_tl . 38, 46, 566, 569,
    572, 612, 614, 617, 629, 631, 634, 682, 684, 687
\__enumext_regex_counter_style: .. 30, 38, 212,
    212, 567, 613, 630, 683
\__enumext_register_counter_style:Nn .. 441,
    441, 446, 447, 448, 449, 450
\__enumext_remove_extra_parsep_vii: .. 3605,
    3815, 3815
\__enumext_remove_extra_parsep_viii: . 3962,
    4168, 4168
\__enumext_renew_footnote: ... 2801, 2805, 3758,
    4111
\l__enumext_renew_the_count_v_tl 685, 694, 696
\l__enumext_renew_the_count_vii_tl 615, 644,
    646
\l__enumext_renew_the_count_viii_tl 632, 651,
    653
\l__enumext_renew_the_count_X_tl ..... 46
\__enumext_rescan_anskey_env:n 78, 2500, 2579,
    2584
\__enumext_reset_global_bool: ... 308, 311, 320
\__enumext_reset_global_int: ... 308, 310, 314
\__enumext_reset_global_tl: ... 308, 312, 326
\__enumext_reset_global_vars: . 32, 67, 308, 308,
    1975
\l__enumext_resume_active_bool 58, 60, 57, 1542,
    1662
\__enumext_resume_counter: .. 59, 60, 1660, 1666,
    1673
\__enumext_resume_counter:n . 58, 60, 1631, 1636,
    1660, 1660, 1730, 1738
\__enumext_resume_counter_save_ans: .. 60, 61,
    1660, 1671, 1703
\__enumext_resume_counter_series: 60, 61, 1660,
    1669, 1686
\g__enumext_resume_int ... 57, 1583, 1677, 1678
\__enumext_resume_last:n .. 58, 1538, 1544, 1557
\l__enumext_resume_name_tl 57, 1579, 1587, 1590,
    1606, 1614, 1617, 1663, 1664, 1692, 1699
\__enumext_resume_save_counter: 58, 1570, 1570,
    3236, 3646
\__enumext_resume_series:n . 59, 1502, 1627, 1627
\__enumext_resume_starred: . 61, 1503, 1724, 1724
\g__enumext_resume_vii_int . 102, 57, 1610, 1682,
    1683
\__enumext_safe_exec: .. 34, 90, 3073, 3090, 3090
\__enumext_safe_exec_vii: . 34, 3589, 3610, 3610
\__enumext_safe_exec_viii: ... 3947, 3967, 3967
\l__enumext_series_name_tl ..... 60
\l__enumext_series_str . 59, 90, 1500, 1540, 1548,
    1549, 1551, 1553, 1574, 1577, 1581, 1601, 1604, 1608,
    3104, 3626
\__enumext_set_error:nn ..... 4335, 4345, 4347
\__enumext_set_parse:n ..... 4318, 4335, 4335
\l__enumext_setkey_tmpa_int ... 117, 4311, 4315
\l__enumext_setkey_tmpa_seq .. 117, 4309, 4319,
    4325, 4327, 4329, 4342
\l__enumext_setkey_tmpa_tl ... 117, 4317, 4321
\l__enumext_setkey_tmpb_seq .. 117, 4310, 4313,
    4317, 4318
\l__enumext_setkey_tmpb_tl 117, 4337, 4339, 4340
\l__enumext_show_answer_bool . 132, 2002, 2026,
    2415, 2721, 2735, 3383, 4072
\__enumext_show_length:nnn .. 44, 220, 220, 4416,
    4417, 4418, 4419, 4420, 4421, 4422, 4423, 4424, 4425,
    4431, 4432, 4433, 4434, 4435, 4436, 4437, 4438, 4439,
    4440
\l__enumext_show_position_bool 132, 2005, 2029,
    2419, 2725, 2736, 3384, 4076
\g__enumext_standar_bool 31, 90, 34, 233, 236, 254,
    323, 1572, 1637, 1649, 1675, 1688, 1726, 1862, 1875
\l__enumext_standar_bool . 90, 93, 34, 2347, 2360,
    2376, 3097, 3235
\l__enumext_standar_first_bool 31, 90, 34, 259,
    1559, 1706, 1768, 1775
\__enumext_standar_item_vii:w . 103, 104, 3686,
    3688, 3688
\__enumext_standar_item_viii:w 110, 4019, 4021,
    4021
\__enumext_standar_ref: .... 38, 557, 577, 3028
\__enumext_standar_ref:n .... 38, 549, 557, 557
\g__enumext_standar_series_tl . 57, 1561, 1562,
    1728, 1731
\g__enumext_starred_bool 31, 102, 34, 242, 245, 268,
    324, 1599, 1642, 1653, 1680, 1695, 1734, 1840, 1881,
    2338, 2348, 2378, 2635, 3123, 3136, 3585
\l__enumext_starred_bool .. 102, 34, 2273, 2279,
    2363, 2404, 2533, 2538, 3619, 3645
\__enumext_starred_columns_set_vii: .. 3471,
    3471, 3598
\__enumext_starred_columns_set_viii: . 3830,
    3830, 3955
\l__enumext_starred_first_bool ... 31, 34, 273,
    1564, 1715, 1768, 1775
\__enumext_starred_item:nn ... 2852, 2852, 2881
\__enumext_starred_item_exec: . 111, 4064, 4064,
    4115
\__enumext_starred_item_vii:w . 103, 104, 3685,
    3704, 3704
\__enumext_starred_item_vii_aux_i:w .. 3704,
    3709, 3712
\__enumext_starred_item_vii_aux_ii:w . 3704,
    3710, 3715, 3717
\__enumext_starred_item_vii_aux_iii:w 3704,
    3720, 3729

```


__enumext_starred_item_viii:w 110, 111, 4018, 4037, 4037
 __enumext_starred_item_viii_aux_i:w . 111, 4037, 4042, 4045
 __enumext_starred_item_viii_aux_ii:w . 111, 4037, 4043, 4057, 4059
 __enumext_starred_joined_item_vii:n 100, 103, 3490, 3490, 3683
 __enumext_starred_joined_item_viii:n . 107, 110, 3849, 3849, 4016
 __enumext_starred_ref: 39, 602, 640, 3056
 __enumext_starred_ref:n 39, 596, 602, 602
 \g__enumext_starred_series_tl . 57, 1566, 1567, 1736, 1739
 __enumext_start_from:NNn 41, 699, 699, 712, 734
 \l__enumext_start_i_int 1678, 1690, 1709
 __enumext_start_item_tmp_vii: 101, 3601, 3668, 3668
 __enumext_start_item_tmp_viii: . . 108, 3958, 4001, 4001
 __enumext_start_item_vii:w 104, 105, 3696, 3701, 3726, 3733, 3735, 3735
 __enumext_start_item_viii:w . . 110, 4029, 4034, 4062, 4092, 4092
 \g__enumext_start_line_tl 31, 142, 261, 275, 329, 1905, 1910, 1915, 1929, 1934, 1939
 __enumext_start_list:nn 33, 87, 98, 347, 349, 3077, 3244, 3397, 3593, 3950
 __enumext_start_mini_vii: 102, 3539, 3539, 3637
 __enumext_start_mini_viii: . . 109, 3898, 3898, 3993
 __enumext_start_save_ans_msg: 62, 1752, 1752, 1777
 __enumext_start_store_level: . 91, 3076, 3117, 3117
 __enumext_start_store_level_vii: 103, 3592, 3648, 3648
 \l__enumext_start_vii_int . . . 1683, 1697, 1718
 \l__enumext_start_X_int 94, 729
 __enumext_stop_item_tmp_vii: . . 101, 103, 105, 3600, 3604, 3670, 3737
 __enumext_stop_item_tmp_viii: 108, 110, 3957, 3961, 4003, 4094
 __enumext_stop_item_vii: 105, 106, 3737, 3791, 3791
 __enumext_stop_item_viii: 112, 4094, 4143, 4143
 __enumext_stop_list: . . 33, 347, 350, 3086, 3254, 3410, 3606, 3964
 __enumext_stop_mini_vii: 101, 102, 3558, 3558, 3641
 __enumext_stop_mini_viii: 109, 3898, 3917, 3997
 __enumext_stop_save_ans_msg: . 62, 1752, 1757, 1964
 __enumext_stop_store_level: . . 91, 3087, 3117, 3146
 __enumext_stop_store_level_vii: . 103, 3607, 3648, 3658
 \l__enumext_store_active_bool . 28, 63, 90, 102, 106, 1707, 1716, 1784, 2233, 3121, 3134, 3259, 3267, 3355, 3414, 3650, 3660, 3976
 __enumext_store_active_keys:n . . 68, 69, 2035, 2035, 3114
 __enumext_store_active_keys_vii:n 68, 69, 102, 2035, 2045, 3629
 __enumext_store_addto_prop:n 70, 80, 2110, 2110, 2118, 2260, 2616, 4067
 __enumext_store_addto_seq:n 70, 81, 2119, 2119, 2123, 2130, 2144, 2152, 2161, 2179, 2187, 2322, 2709
 \l__enumext_store_anskey_arg_tl 28, 73, 74, 106, 2270, 2275, 2277, 2282, 2289, 2292, 2302, 2307, 2310, 2316, 2322
 __enumext_store_anskey_code:nn . 72, 73, 2227, 2258, 2258
 \l__enumext_store_anskey_env_tl . . 78, 79, 106, 2518, 2521, 2574, 2579, 2584
 \l__enumext_store_anskey_opt_tl . . 78, 79, 106, 2519, 2535, 2541, 2548, 2554, 2564, 2576, 2582
 __enumext_store_anskey_safe_outer: 72
 __enumext_store_anskey_show_left:n 76, 2265, 2413, 2413
 __enumext_store_anskey_show_wrap:n 76, 2401, 2401, 2417, 2432
 \g__enumext_store_columns_break_bool . 2472, 2532, 2591
 \l__enumext_store_columns_break_bool . 2206, 2272
 \l__enumext_store_columns_join_int . . . 106
 __enumext_store_internal_ref: . . 73, 74, 2263, 2324, 2324
 \g__enumext_store_item_join_int . . 2475, 2539, 2543, 2592
 \l__enumext_store_item_join_int . . 2209, 2280, 2284
 \g__enumext_store_item_star_bool . 2477, 2546, 2593
 \l__enumext_store_item_star_bool . 2211, 2287
 \g__enumext_store_item_symbol_sep_dim 2482, 2561, 2566, 2595
 \l__enumext_store_item_symbol_sep_dim 2216, 2299, 2304
 \g__enumext_store_item_symbol_tl . 2480, 2552, 2556, 2594
 \l__enumext_store_item_symbol_tl . 2214, 2290, 2294
 \l__enumext_store_keyans_item_opt_sep_tl 1988, 2610, 2612, 2683, 2687, 4050, 4052
 \l__enumext_store_keyans_item_opt_tl . . 106
 \l__enumext_store_keyans_label_tl 28, 80, 81, 111, 106, 2599, 2602, 2605, 2612, 2614, 2616, 2673, 2676, 2679, 2685, 2690, 2700, 2709, 4047, 4052, 4053, 4066, 4067, 4069
 __enumext_store_level_close: . 70, 2124, 2148, 3150
 __enumext_store_level_close_vii: 2155, 2183, 3664
 __enumext_store_level_open: . . 70, 2124, 2124, 3129, 3142
 __enumext_store_level_open_vii: . 2155, 2155, 3654
 \g__enumext_store_name_tl . 28, 63, 93, 106, 328, 335, 336, 337, 338, 1760, 1786, 1904, 1909, 1914, 1928, 1933, 1938, 1962, 2521, 2522
 \l__enumext_store_name_tl 28, 62, 64, 106, 1593, 1596, 1620, 1623, 1711, 1720, 1755, 1764, 1765, 1786, 1787, 1788, 1790, 1791, 1793, 1795, 1796, 1798, 1800, 1801, 1825, 2112, 2114, 2121, 2390, 2391, 2427, 2656, 2657, 2760, 2773, 4084
 \l__enumext_store_ref_key_bool 73, 2011, 2261, 2313, 2620, 2697

```

\l__enumext_store_save_key_vii_bool .. 2047,
    2077
\l__enumext_store_save_key_vii_tl 2049, 2050,
    2078, 2079, 2159, 2169, 2175, 2179
\l__enumext_store_save_key_X_bool ..... 68
\l__enumext_store_save_key_X_tl .. 68, 69, 122
\l__enumext_store_upper_level_X_bool .. 122
\l__enumext_store_write_aux_file_tl 29, 75, 81,
    154, 2393, 2399, 2663, 2669
\__enumext_storing_exec: . 62, 63, 77, 1762, 1778,
    1782
\__enumext_storing_set:n .. 62, 1747, 1762, 1762
\l__enumext_the_counter_v_tl ..... 684
\l__enumext_the_counter_vii_tl ..... 614
\l__enumext_the_counter_viii_tl ..... 631
\l__enumext_the_counter_X_tl ..... 46
\__enumext_tmp:n 41, 45, 50, 56, 68, 75, 76, 81, 88, 93,
    94, 105, 123, 131, 157, 161, 165, 184, 812, 816, 1496,
    1507, 1743, 1751, 1804, 1822, 1978, 2016, 2017, 2034,
    2053, 2066, 2326, 2333, 2334, 2355, 2369, 2372, 2384,
    2622, 2629, 2998, 3037, 3038, 3070
\__enumext_tmp:nn 473, 494, 495, 523, 524, 536, 729,
    748, 793, 811, 869, 877, 878, 892, 957, 973, 974, 987,
    1385, 1401, 2785, 2800
\__enumext_tmp:nnn 537, 553, 554, 555, 556, 584, 600,
    601
\__enumext_tmp:nnnnn 749, 774, 777, 780, 782, 784,
    787, 790
\__enumext_tmp:w ..... 4195, 4198
\l__enumext_tmpa_vii_int ..... 3481, 3484
\l__enumext_tmpa_viii_int ..... 3840, 3843
\l__enumext_tmpa_X_int ..... 165
\l__enumext_topsep_v_skip 1045, 1049, 1192, 1205,
    1213, 1218, 1238, 1242, 3413, 3445
\l__enumext_topsep_vii_skip .. 1269, 1278, 1282
\l__enumext_topsep_viii_skip . 1291, 1313, 1317
\__enumext_undefine_anskey_env: . 67, 77, 1973,
    2435, 2455
\l__enumext_vspace_a_star_v_bool ..... 1434
\l__enumext_vspace_a_star_vii_bool ... 1456
\l__enumext_vspace_a_star_viii_bool ... 1467
\l__enumext_vspace_a_star_X_bool ..... 94
\__enumext_vspace_above: .. 55, 1402, 1402, 3155
\__enumext_vspace_above_v: . 56, 1430, 1430, 3283
\l__enumext_vspace_above_v_skip .. 1432, 1436,
    1438
\__enumext_vspace_above_vii: .. 56, 1452, 1452,
    3634
\l__enumext_vspace_above_vii_skip 1454, 1458,
    1460
\__enumext_vspace_above_viii: . 56, 1452, 1463,
    3991
\l__enumext_vspace_above_viii_skip 1465, 1469,
    1471
\l__enumext_vspace_b_star_v_bool ..... 1445
\l__enumext_vspace_b_star_vii_bool ... 1478
\l__enumext_vspace_b_star_viii_bool ... 1489
\l__enumext_vspace_b_star_X_bool ..... 94
\__enumext_vspace_below: .. 55, 1416, 1416, 3234
\__enumext_vspace_below_v: . 56, 1441, 1441, 3351
\l__enumext_vspace_below_v_skip .. 1443, 1447,
    1449
\__enumext_vspace_below_vii: .. 56, 1474, 1474,
    3644
\l__enumext_vspace_below_vii_skip 1476, 1480,
    1482
\__enumext_vspace_below_viii: . 56, 1474, 1485,
    3999
\l__enumext_vspace_below_viii_skip 1487, 1491,
    1493
\__enumext_widest_from:nnn .. 41, 713, 713, 728,
    740
\g__enumext_widest_label_tl 27, 36, 64, 461, 465,
    469
\l__enumext_wrap_label_opt_v_bool ..... 2894
\l__enumext_wrap_label_opt_vii_bool 104, 3695
\l__enumext_wrap_label_opt_viii_bool .. 110,
    4028
\l__enumext_wrap_label_opt_X_bool ..... 94
\l__enumext_wrap_label_v_bool 2890, 2894, 2902,
    2958
\l__enumext_wrap_label_vii_bool .. 104, 3694,
    3699, 3707, 3775
\l__enumext_wrap_label_viii_bool . 110, 4027,
    4032, 4040, 4119
\l__enumext_wrap_label_X_bool ..... 94
\__enumext_wrapper_label_v:n ..... 2960, 3392
\__enumext_wrapper_label_vii:n ..... 3778
\__enumext_wrapper_label_viii:n ..... 4122
\__enumext_zero_parsep: ... 50, 1089, 1144, 1144
enumext* ..... 5, 3587
enumXi ..... 433
enumXii ..... 433
enumXiii ..... 433
enumXiv ..... 433
enumXv ..... 433
enumXvi ..... 433
enumXvii ..... 433
enumXviii ..... 433
Environments provide by enumext:
anskeyenv ..... 28, 63, 67, 76–78, 117
enumext* .. 25, 26, 28–31, 34, 35, 38, 39, 41–46, 52, 53,
    56–59, 61, 62, 64, 65, 67–76, 78, 80, 83, 89–91, 102, 103,
    105–107, 109, 112, 114, 115, 118, 120
enumext 25, 26, 28, 30, 31, 34–42, 44–52, 54, 55, 57–59, 61,
    62, 64, 65, 67–76, 78, 80, 83–88, 90, 91, 93, 94, 98–100,
    103, 114, 115, 117, 119
keyans* 25, 26, 28–32, 35, 38–46, 52, 53, 56, 63, 66, 68, 70,
    80, 83, 89, 109, 118, 120
keyanspic .. 25, 26, 28, 29, 32, 35, 37, 40, 54, 63, 66, 70,
    80–82, 96–98, 119
keyans 25, 26, 28, 29, 31, 32, 35–37, 40–42, 44–46, 48, 51,
    52, 54–56, 63, 66, 68, 70, 80–82, 86–88, 93, 94, 96–98,
    100, 109, 118, 119
Environments:
list ..... 30, 33, 87, 90
lrbox ..... 99, 105, 106, 112
minipage .. 30, 33, 34, 46, 48, 49, 96–99, 105, 106, 112
multicols ..... 47–49, 54, 92, 93, 95, 96
scontents ..... 77, 78
exp commands:
\exp_after:wN ..... 4198
\exp_args:Ne ..... 2578, 2583, 3111, 4186
\exp_not:N . 54, 464, 572, 617, 634, 687, 826, 840, 841,
    852, 853, 864, 865, 2318, 2424, 2425, 2702, 2757, 2758,
    2770, 2771, 4081, 4082, 4195
\exp_not:n 263, 277, 289, 296, 303, 572, 573, 617, 618,
    634, 635, 687, 688, 827, 1524, 1536, 1999, 2096, 2108,

```

2284, 2294, 2304, 2318, 2319, 2396, 2543, 2556, 2566, 2666, 2704, 2706	
F	
\fbox	1983
file commands:	
\file_input_stop:	4585
first	878
font	473
\footnote	83
\footnote	83, 2809
\footnotemark	2819
\footnotesize	2425, 2758, 2771, 4082
\footnotetext	2803
G	
\getkeyans	15, 113, 4184
group commands:	
\group_begin:	2222, 2423, 2502, 2573, 2756, 2769, 3754, 3773, 4080, 4107, 4117, 4206, 4240
\group_end:	2229, 2430, 2587, 2763, 2776, 3783, 3795, 4087, 4127, 4147, 4208, 4247
H	
\hbadness	3802, 4154
hbox commands:	
\hbox_set:Nn	453
\hfill	503, 507, 512, 513, 1356, 1374, 2318, 2702, 3563, 3922
hook commands:	
\hook_gput_code:nnn	9, 192, 196, 200, 371
\hook_gset_rule:nnnn	372
\hspace	3813, 4166
\hyperlink	74, 81
\hyperlink	2318, 2702
\hypertarget	34
\hypertarget	401
I	
\IfHyperBoolean	379
\IfPackageLoadedTF	11, 19, 375, 389
\ignorespaces	829
\inputlineno	263, 277, 289, 296, 303
int commands:	
\int_add:Nn	3523, 3882
\int_case:nn	1002, 1146, 1835, 1857, 1895, 1919
\int_compare:nNnNTF	360, 605, 622, 642, 649, 1071, 1190, 1335, 1339, 1343, 1943, 1949, 1960, 2237, 2241, 2253, 2437, 2447, 2600, 2639, 2644, 2649, 2674, 2752, 3095, 3105, 3126, 3139, 3178, 3194, 3208, 3222, 3268, 3272, 3301, 3326, 3339, 3359, 3363, 3419, 3493, 3503, 3519, 3615, 3652, 3662, 3808, 3817, 3852, 3862, 3878, 3970, 3977, 4160, 4170, 4315
\int_compare_p:nNn	234, 243, 255, 256, 269, 270, 1841, 1863, 2280, 2339, 2349, 2361, 2362, 2377, 2379, 2539
\int_decr:N	3522, 3881
\int_eval:n	345, 2114, 2391, 2425, 2657, 2758, 2771, 3013, 3055, 3511, 3870, 4082
\int_from_alph:n	707, 721
\int_from_roman:n	709, 723
\int_gadd:Nn	3524, 3883
\int_gdecr:N	1844, 1848, 1851, 1854, 1866
\int_gincr:N	1677, 1682, 2225, 2712, 2840, 2870, 2908, 3168, 3292, 3381, 3672, 3750, 4005, 4071
\int_gset:Nn	1888, 2817
\int_gset_eq:NN	1576, 1583, 1589, 1595, 1603, 1610, 1616, 1622, 2814
\int_gzero:N	316, 317, 318, 1364, 1381, 1955, 2592, 3227, 3344, 3826, 4181
\int_if_exist:NTF	1551, 1587, 1593, 1614, 1620, 1798
\int_incr:N	2252, 3094, 3263, 3418, 3614, 3671, 3969, 4004
\int_mod:nn	3819, 4172
\int_new:N	28, 29, 30, 31, 32, 33, 57, 58, 82, 98, 112, 119, 139, 140, 148, 149, 150, 151, 162, 168, 169, 170, 171, 172, 1553, 1801
\int_set:Nn	703, 707, 709, 1690, 1697, 1709, 1718, 2503, 3458, 3459, 3481, 3492, 3498, 3514, 3802, 3840, 3851, 3857, 3873, 4154, 4311
\int_set_eq:NN	1678, 1683, 3521, 3880
\int_sign:n	1890
\int_step_function:nnN	2355, 2369, 2384
\int_step_inline:nnn	3460
\int_to_roman:n	204, 2335, 2373
\int_use:N	338, 343, 344, 1072, 1692, 1699, 1711, 1720, 3013, 3032, 3055, 3112, 3179, 3188, 3203, 3209, 3496, 3497, 3509, 3855, 3856, 3868
\int_zero:N	3811, 4164
\c_one_int	3481, 3500, 3506, 3512, 3516, 3519, 3840, 3859, 3865, 3871, 3875, 3878
\c_zero_int	2339, 2349, 2361, 2362, 2377, 2379, 3652, 3662, 3822, 4177
\item	33, 45, 46, 71, 84, 96, 98, 99, 101, 108
\item	84, 85, 103, 105, 109, 112, 351, 2132, 2138, 2163, 2171, 2277, 2676, 2679, 2877, 2912, 3599, 3601, 3956, 3958, 4069
\item*	5, 14, 66, 2910
item-pos*	2785
item-sym*	2785
\itemindent	88
\itemindent	87
itemindent	793
\itemsep	97, 98
\itemsep	3434, 3440
\itemwidth	3488, 3532, 3536, 3847, 3891, 3895
K	
keyans	13, 3239
keyans*	13, 3945
keyanspic	14, 3394
Keys for command provide by enumext:	
break-col	72, 73, 77, 78
item-join	72, 73, 77, 78
item-pos*	72, 74, 77, 79
item-star	72, 74, 77, 79
item-sym*	72, 74, 77, 79
Keys for environments provide by enumext:	
above*	27, 55, 56
above	27, 55, 56, 91, 94, 102, 109
after	44-46, 93, 96, 102, 109
align	27, 37, 86, 105
before*	44, 45, 91, 102, 109
before	44, 45, 94
below*	27, 55, 56
below	27, 55, 56, 93, 96, 102, 109
check-ans	28-31, 62-67, 70, 82, 84, 85, 91, 93, 106, 118
columns-sep	46, 92, 95
columns	27, 46, 49, 55, 92, 95
first	44-46, 105
font	36, 86, 105
item-pos*	83
item-sym*	26, 83, 85

item*-sep	85
itemindent	27, 42, 43, 86, 105
itemsep	42, 89
labelsep	36, 84, 88, 105
labelwidth	36, 38, 40, 41, 88
label	26, 27, 36, 37, 41, 99
lisparindent	89
list-indent	27, 42, 43, 98
list-offset	42, 43
listparindent	42, 105
mark-ans	28, 67, 70, 76
mark-pos	67, 68
mark-ref	28, 67, 70, 74
mini-env	27, 34, 46, 54, 55, 70, 83, 92, 94, 100, 102, 107, 109
mini-right*	27, 46, 70, 101
mini-right	27, 46, 53, 70, 101
mini-rigth*	30
mini-rigth	30
mini-sep	27, 46, 70, 92, 94
no-store	29, 62-64, 69, 72
noitemsep	42, 50
nosep	42, 50
parindent	89
parsep	42, 89, 105
partopsep	42
ref	26, 30, 37, 38, 40, 118
resume*	26, 57, 58, 61-63, 69, 93
resume	26, 33, 57-63, 69, 70, 93, 102
rightmargin	42
save-ans	28, 33, 57-62, 64, 65, 67-70, 72, 77, 80, 81, 85, 90, 94, 96, 97, 102, 109, 111, 113, 114, 118
save-key	28, 57, 69, 90
save-pos	70
save-ref	29, 35, 67, 70, 73, 74, 80, 81, 86, 111
save-sep	67, 70, 111
series	26, 57-61, 70, 90, 93
show-ans	28, 67, 68, 70, 71, 73, 76, 86, 111
show-length	31, 44, 117, 118
show-pos	28, 67, 68, 71, 73, 76, 82, 86, 111
start	27, 30, 41, 57
store-key	68
topsep	42
widest	27, 30, 41
wrap-ans	67, 70, 71, 76
wrap-label*	36, 84, 86, 104, 105, 110
wrap-label	36, 86, 104, 105, 110
wrap-opt	67, 70
keys commands:	
\keys_define:nn	475, 497, 526, 539, 586, 657, 731, 751, 795, 814, 871, 880, 959, 976, 1387, 1498, 1745, 1806, 1980, 2019, 2055, 2060, 2204, 2470, 2491, 2787, 4211, 4280
\l_keys_key_str	4401
\keys_precompile:nnN	114, 4210, 4213, 4217, 4221, 4225, 4229, 4233
\keys_set:nn	489, 982, 1392, 1397, 1639, 1644, 1731, 1739, 2268, 2582, 3107, 3111, 3279, 3627, 3986, 4282, 4283, 4284, 4285, 4286, 4287, 4288, 4289, 4290, 4291, 4292, 4293, 4294, 4332
keyval commands:	
\keyval_parse:NNn	1512, 2085
L	
label	537, 584, 657

Labels provide by enumext:	
\Alph*	36
\Roman*	36
\alph*	36
\arabic*	30, 36
\roman*	36
\labelsep	98
\labelsep	3435, 3438
labelsep	473
\labelwidth	36, 98
\labelwidth	3435, 3436
labelwidth	473
\leftmargin	88
\leftmargin	87, 3435
legacy commands:	
\legacy_if:nTF	3738, 3741, 4095, 4098
\legacy_if_gset_false:n	365
\legacy_if_set_false:n	3740, 4097
\legacy_if_set_true:n	3700, 3725, 3732, 3745, 4033, 4061, 4102
\linewidth	92, 94
\linewidth	3163, 3289, 3457, 3484, 3545, 3843, 3904
\list	33
\list	349
list-indent	793
list-offset	793
\listparindent	3437
listparindent	793
\lrbox	3755, 4108

M

\makebox	99
\makebox	2194, 2196, 2932, 3769, 3777, 3781, 4121, 4125
\makelabel	84, 86, 87, 99
\makelabel	86, 87, 2938, 2954
\makesavenoteenv	395
mark-ans	1978
mark-pos	1978, 2017
mark-ref	1978
mini-env	957
mini-sep	957
\minipage	33
\minipage	355
\miniright	10, 53, 1333, 3225, 3342
\miniright*	10
mode commands:	
\mode_if_vertical:TF	1027, 1055, 1171, 1250
\mode_leave_vertical:	826, 840, 852, 864, 2163, 2171, 2192, 2930, 3767
msg commands:	
\msg_error:nn	2250, 2255, 3270, 3274, 3361, 3421, 3617, 3972, 3979, 4295
\msg_error:nnn	562, 609, 626, 679, 1337, 1341, 1366, 1383, 1651, 1655, 1770, 4200, 4205, 4277, 4348
\msg_error:nnnn	2235, 2239, 2243, 3261, 3357, 3365, 4260
\msg_error:nnnnn	1998
\msg_fatal:nn	3096
\msg_fatal:nnn	427, 2441, 2451
\msg_info:nnn	13, 16, 21, 24, 377, 391
\msg_line_context:	4373, 4377, 4381, 4405, 4410, 4415, 4430, 4445, 4449, 4453, 4457, 4461, 4465, 4471, 4477, 4483, 4497, 4501, 4506, 4510, 4515, 4519, 4523, 4528, 4533, 4537, 4542, 4547, 4551, 4556, 4560, 4565, 4570, 4575, 4579, 4583

\msg_log:nnn	1790, 1795, 1800
\msg_log:nnnnn	342, 1928, 1933, 1938
\msg_log:nnnnnn	334
\msg_new:nnn	4349, 4353, 4357, 4361, 4366, 4371, 4375, 4379, 4383, 4389, 4395, 4399, 4403, 4408, 4413, 4428, 4443, 4447, 4451, 4455, 4459, 4463, 4467, 4474, 4480, 4486, 4490, 4494, 4499, 4504, 4508, 4513, 4517, 4521, 4526, 4531, 4535, 4540, 4545, 4549, 4554, 4558, 4563, 4568, 4573, 4577, 4581
\msg_term:nnnn	1754, 1759, 3022, 3032, 3061, 3066
\msg_term:nnnnn	1909
\msg_warning:nn	3224, 3341
\msg_warning:nnnn	1946, 1952, 2970, 2975, 3495, 3508, 3854, 3867
\msg_warning:nnnnn	1904, 1914
\multicolsep	92, 95
\multicolsep	3193, 3314
N	
\NeedsTeXFormat	3
\newcounter	430
\NewDocumentCommand	1333, 2219, 3353, 4184, 4238, 4302
\NewDocumentEnvironment	3071, 3239, 3394, 3587, 3945
\newenvsc	2464
\newlabel	35
\newlabel	413
no-store	1804
\noindent	101, 108
\noindent	3170, 3294, 3554, 3600, 3810, 3913, 3957, 4163
\nointerlineskip	3170, 3294, 3554, 3913
noitemsep	749
\nopagebreak	1038, 1066, 1182, 1261, 1324, 1330
\normalfont	2424, 2757, 2770, 4081
nosep	749
P	
Packages:	
enumext	25, 37, 62, 88, 96, 117
enumitem	35, 36
expl3	99
footnotehyper	34
hyperref	29, 30, 34, 35, 74, 81, 105, 116
lua-visual-debug	49
multicol	25, 116
scontents	25, 76, 79
shortlst	99
\par	1038, 1066, 1182, 1261, 1324, 1330, 1359, 1376, 2403, 3214, 3229, 3331, 3346, 3469, 3572, 3579, 3810, 3824, 3931, 3938, 4163, 4179
\parindent	3787, 4131
\parsep	47, 50, 97, 98
\parsep	2164, 2172, 3052, 3434, 3441, 3446
parsep	749
\parskip	3788, 4132
\partopsep	98
\partopsep	3053, 3439
partopsep	749
peek commands:	
\peek_meaning:N	3677, 3691, 3708, 3719, 4010, 4024, 4041
\peek_meaning_remove:N	3684, 4017
\peek_remove_spaces:n	2916
\phantomsection	34
\phantomsection	402
prg commands:	
\prg_do_nothing:	406

\prg_new_protected_conditional:Npnn	206
\prg_replicate:nn	223
\prg_return_false:	210
\prg_return_true:	209
\printkeyans	16, 114, 4238
prop commands:	
\prop_count:N	336, 2114, 2391, 2427, 2657, 2760, 2773, 4084
\prop_gput_if_not_in:Nnn	2112
\prop_if_exist:N	1788, 4204
\prop_item:Nn	4207
\prop_new:N	1791
\ProvidesExplPackage	4
R	
\raggedcolumns	3202, 3320
\ref	74, 80
ref	537, 584, 657
\refstepcounter	3747, 4104
regex commands:	
\regex_match:nnTF	208, 706, 708, 720, 722
\regex_replace_once:nnN	216
\renewcommand	572, 617, 634, 687
\RenewDocumentCommand	2809, 2877, 2912, 2938, 2954
\RequirePackage	17, 25
resume	1496
resume*	1496
rightmargin	793
\Roman	36, 41
\Roman	449
\roman	36, 41
\roman	450, 555, 4228
S	
save-ans	1743
save-key	2053
save-ref	1978
save-sep	1978
scan commands:	
\scan_stop:	98, 3448, 3599, 3956, 4195, 4198
scontents internal commands:	
__scontents_anskeyenv_env_begin:	2439, 2449, 2459
__scontents_anskeyenv_env_end:	2460
__scontents_rescan_tokens:n	2504
seq commands:	
\seq_clear:N	4309
\seq_const_from_clist:Nn	4297
\seq_count:N	337, 3407, 4313
\seq_gclear:N	2807, 2808
\seq_gpop_right:NNTF	2520
\seq_gput_right:Nn	2121, 2820, 2821
\seq_if_empty:N	2826, 4253, 4327
\seq_if_exist:N	1793, 4251
\seq_if_in:NnTF	4258
\seq_item:Nn	2522, 3466
\seq_map_function:NN	4318
\seq_map_inline:Nn	4265, 4271, 4306, 4328, 4329
\seq_map_pairwise_function:NNN	2828
\seq_new:N	120, 121, 137, 163, 164, 1796
\seq_pop_left:NN	4317
\seq_put_right:Nn	3367, 4325, 4342
\seq_set_from_clist:Nn	4310
\seq_set_map_e:NNn	4319
\seq_show:N	4255

series	1496
\setcounter	717, 721, 723, 3013, 3055, 3412
\setenumext	6, 115, 4302
\setlength	2165, 2173
show-ans	1978, 2017
show-length	869
show-pos	2017
skip commands:	
\skip_add:Nn	1007, 1013, 1019, 1029, 1033, 1057, 1061, 1151, 1157, 1163, 1173, 1177, 1199, 1252, 1256, 3434
\skip_eval:n	2164, 2172
\skip_gset:Nn	1272, 1276, 1280
\skip_gzero_new:N	1267, 1268
\skip_horizontal:N	841, 853, 865, 3770, 3784, 4128
\skip_horizontal:n	827, 2193, 2201, 2931, 2933, 3768, 4137
\skip_if_eq:nnTF	1005, 1011, 1017, 1074, 1108, 1149, 1155, 1161, 1192, 1197, 1218, 1269, 1291, 1404, 1418, 1432, 1443, 1454, 1465, 1476, 1487
\skip_new:N	78, 79, 83, 84, 85, 86, 87, 141, 182
\skip_set:Nn	990, 994, 1043, 1047, 1077, 1081, 1085, 1092, 1096, 1100, 1111, 1116, 1120, 1126, 1131, 1136, 1194, 1195, 1196, 1203, 1207, 1211, 1220, 1225, 1229, 1232, 1236, 1240, 1271, 1275, 1293, 1297, 1301, 1307, 1311, 1315, 3428, 3442
\skip_set_eq:NN	3011, 3051, 3052, 3787, 3788, 4131, 4132
\skip_use:N	992, 996, 1031, 1035, 1039, 1059, 1063, 1075, 1094, 1103, 1109, 1114, 1118, 1129, 1133, 1134, 1139, 1175, 1179, 1205, 1405, 1409, 1412, 1419, 1423, 1426, 3214
\skip_zero:N	3053, 3193, 3314, 3439, 3440
\skip_zero_new:N	1187, 1188, 1189, 1266, 1288, 1289, 1290
\c_zero_skip	1005, 1011, 1017, 1075, 1109, 1149, 1155, 1161, 1192, 1197, 1218, 1269, 1291, 1405, 1419, 1432, 1443, 1454, 1465, 1476, 1487
\small	4216, 4220, 4224, 4228, 4232, 4236
\star	2791
start	729
\stepcounter	2813, 3374
str commands:	
\c_backslash_str	4373, 4377, 4381, 4385, 4386, 4387, 4391, 4392, 4488, 4492, 4496, 4510, 4511, 4515, 4523, 4524, 4528, 4529, 4560, 4561, 4565, 4570, 4571
\c_colon_str	2390, 2656, 4195
\c_left_brace_str	4453, 4457, 4470, 4476, 4482
\c_right_brace_str	4453, 4457, 4470, 4476, 4482
\str_case:nn	228, 283
\str_case:nnTF	1519, 1528, 2092, 2100
\str_clear:N	3104, 3626
\str_count:n	223
\str_if_empty:N	1540, 1581, 1608
\str_if_eq:nnTF	3014, 3057
\str_if_in:nnTF	4191
\str_new:N	136, 177
\str_set:Nn	529, 530, 531, 1995, 1996, 2022, 2023
\string	395
\strutbox	1079, 1083, 1087, 1098, 1102, 1113, 1122, 1128, 1138, 1151, 1157, 1163, 1194, 1195, 1196, 1199, 1209, 1213, 1222, 1229, 1234, 1242, 1271, 1272, 1275, 1282, 1295, 1303, 1309, 1317, 3444

T	
TeX and L ^A T _E X 2 _ε commands:	
\@auxout	411
\@currentvar	228, 283
\protected@write	411
tex commands:	
\tex_newlinechar:D	2503
text commands:	
\text_expand:n	4187
\textasteriskcentered	1992, 2009
\thepage	417
tl commands:	
\c_space_tl	2738, 4415, 4430, 4453, 4457
\tl_clear:N	502, 508, 1956, 2039, 2049, 2070, 2078, 2270, 2518, 2519, 2599, 2673, 4047
\tl_clear_new:N	459
\tl_const:Nn	46, 443
\tl_gclear:N	328, 329, 330, 1561, 1566, 2594, 2949, 3583, 3771, 3942
\tl_gclear_new:N	1548
\tl_gput_right:Nn	444
\tl_greplace_all:Nnn	465
\tl_gset:Nn	260, 261, 274, 275, 1549, 1562, 1567, 1786, 3714
\tl_gset_eq:NN	461, 2858, 3764
\tl_if_blank:nTF	3762
\tl_if_empty:N	560, 579, 607, 624, 644, 651, 677, 694, 1574, 1579, 1601, 1606, 1664, 1728, 1736, 1765, 1825, 1962, 2128, 2159, 2290, 2552, 2574, 2576, 2610, 2683, 2732, 2928, 4050, 4340
\tl_if_empty:nTF	1629, 2248
\tl_if_exist:N	1634
\tl_if_novalue:nTF	2266, 2607, 2681, 2717, 2811, 2836, 2854, 2859, 2888, 3102, 3405, 3624, 3984, 4048, 4304
\tl_map_inline:Nn	214, 462
\tl_new:N	43, 48, 49, 52, 53, 59, 61, 62, 63, 65, 66, 99, 100, 101, 107, 108, 109, 110, 111, 113, 114, 115, 116, 117, 118, 122, 125, 127, 128, 134, 135, 145, 146, 147, 154, 155, 156, 159, 176, 179
\tl_put_left::N	2541
\tl_put_left:Nn	2136, 2169, 2275, 2535, 2548, 2554, 2564, 2744, 2779, 4066, 4069
\tl_put_right:Nn	460, 570, 615, 632, 685, 2140, 2175, 2277, 2282, 2289, 2292, 2302, 2307, 2310, 2316, 2342, 2352, 2366, 2382, 2388, 2393, 2602, 2605, 2612, 2614, 2641, 2646, 2651, 2654, 2663, 2676, 2679, 2685, 2690, 2700, 4052, 4053
\tl_remove_all:Nn	4339
\tl_remove_once:Nn	2330, 2626
\tl_replace_all:Nnn	464
\tl_reverse:N	2329, 2331, 2625, 2627
\tl_set:Nn	54, 287, 294, 301, 429, 503, 507, 512, 513, 559, 604, 676, 824, 838, 850, 862, 1663, 1764, 2040, 2050, 2071, 2079, 2421, 2719, 2754, 2767, 2856, 4055, 4078, 4337
\tl_set_eq:NN	470, 565, 568, 612, 614, 629, 631, 682, 684, 2328, 2624, 2637, 2900, 2904, 3386, 3388
\tl_to_str:n	1634, 1640, 1645, 4187
\tl_trim_spaces:n	460, 4325, 4337, 4343
\tl_use:N	466, 469, 581, 646, 653, 696, 895, 899, 903, 907, 911, 915, 919, 923, 927, 931, 935, 939, 943, 947, 951, 955, 2198, 2335, 2343, 2354, 2368, 2373, 2385, 2843, 2849, 2873, 2891, 2895, 2903, 2940, 2941, 2948, 2956, 2957, 2963, 3078, 3245, 3391, 3577, 3774, 3785,

3789, 3936, 4118, 4129, 4135, 4140, 4241, 4242, 4243, 4244, 4245, 4263, 4321	\usecounter 3012, 3054
token commands:	V
\token_to_str:N 413	\value 1577, 1583, 1590, 1596, 1604, 1610, 1617, 1623
\topsep 2165, 2173	\vspace 366, 1409, 1412, 1423, 1426, 1436, 1438, 1447, 1449,
topsep <u>749</u>	1458, 1460, 1469, 1471, 1480, 1482, 1491, 1493, 2164,
\typeout 381, 384, 394, 395	2172, 3402, 3413, 3825, 4180
U	W
\u 217	widest <u>729</u>
use commands:	wrap-ans <u>1978</u>
\use:N 224, 2945, 3080	wrap-label <u>473</u>
\use:n 1510, 2083, 4193	wrap-label* <u>473</u>
\use_none:nn 405	wrap-opt <u>1978</u>