

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-14^{*}

©2024 by Pablo González[†]

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

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

Abstract

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

Contents

1	Introduction	2	4	The storage system	9
1.1	Description and usage	3	4.1	Keys for storage	9
1.2	The concept of left margin	3	4.2	Keys for internal label and ref	10
1.3	User interface	3	4.3	Keys for debugging and checking	10
1.3.1	Internal counters	3	4.4	The command <code>\anskey</code>	10
1.3.2	Support for multicol	4	4.5	The environment <code>keyans</code>	10
1.3.3	Support for minipage	4	4.5.1	The <code>\item*</code> in <code>keyans</code>	11
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	4.6	The environment <code>keyanspic</code>	11
1.3.5	Support for <code>\footnote</code>	4	4.6.1	The command <code>\anspic</code>	12
2	The environment <code>enumext</code>	4	4.7	Printing stored content	12
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	12
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	12
3	The command <code>\setenumext</code>	5	5	Full examples	13
3.1	Keys for <code>label</code> and <code>ref</code>	6	6	The way of non-enumerated lists	16
3.2	Keys for spaces	6	7	References	18
3.2.1	Vertical spaces	7	8	Change history	18
3.2.2	Horizontal spaces	7	9	Index of Documentation	19
3.3	Keys for add code	8	10	Implementation	21
3.4	Keys for start and resume	8	11	Index of Implementation	104
3.5	Keys for multicol	8			
3.6	Keys for minipage	8			
3.6.1	The command <code>\miniright</code>	9			
3.6.2	The key <code>miniright</code>	9			

Motivation and acknowledgments

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

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

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all \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 2024-05-14.

[†]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 (l^{pp}l), version 1.3 or later (<https://www.latex-project.org/l^{pp}l.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[3] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TE_X. 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 \LaTeX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like `exam`[1] and packages like `xsim`[2] do the job perfectly, but they don’t always fit the basic day to day needs.

In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

4. Related to Linux
- (a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

Sometimes we are also interested in showing the “answers” along with the questions:

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

4. Related to Linux
- (a) You use linux?

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

Or we are interested in referring to a specific question and its “answer”, for example:

The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.

Or we are interested in printing all the “answers”:

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

3. Third type of questions

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

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


(A) value

(B) value


(C) value

(D) value


(E) value
4. Question with image and label below:




(A)




(B)



(C)



(D)



(E)

5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value



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

1. (B), $x = 5$

2. (D)

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

5. (D), “other note”

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

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

1.1 Description and usage

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

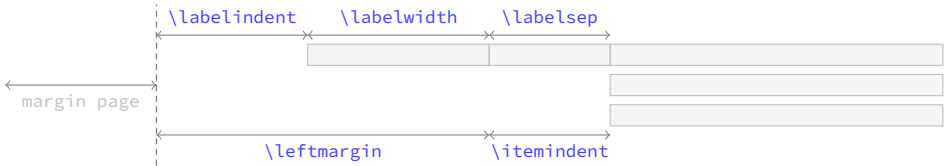


Figure 1: Representation of horizontal lengths in `enumitem`.

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

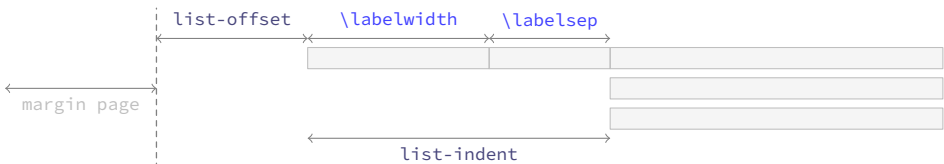


Figure 2: Representation of horizontal lengths concept in `enumext`.

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Support for 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.

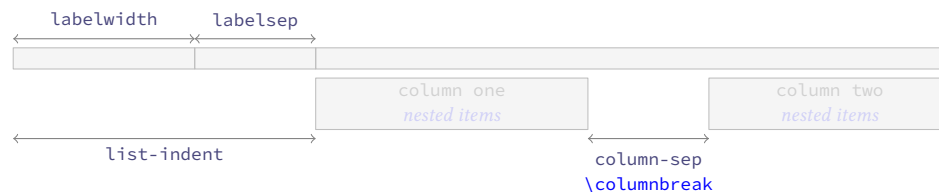


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §3.5).

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

1.3.3 Support for minipage

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



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

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

1.3.4 The \label and \ref system

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

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

1.3.5 Support for \footnote

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

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

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

2 The environment enumext

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

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

The `enumext` is an “*enumerated list*” environment that works in the same way as the standard `enumerate` environment provided by L^AT_EX, `\item` and `\item[⟨custom⟩]` commands work in the usual way.

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

Example

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

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

2.1 The `\item*` in `enumext`

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

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

3.1 Keys for label and ref

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

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

- This key is intended to give the basic structure with which the *label* will be displayed, and the and the form in which it is used by standard “*label and ref*” and the “*internal reference*” system with the *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., `\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⟩]`.

3.2 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`.

3.2.1 Vertical spaces

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

Set the *vertical space* added to both the top and bottom of the list. Internally sets the value of `\topsep` for the current level. The default values for first level are 8.0pt 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.

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

Set the *vertical space* between paragraphs within an item. Internally sets the value of `\parsep` for the current level. The default values for first level are 4.0pt 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.

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

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

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

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

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

`noitemsep` *value forbidden* default: *not used*

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

`nosep` *value forbidden* default: *not used*

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

`itemindent` = {*rigid length*} default: 0pt

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

`rightmargin` = {*rigid length*} default: 0pt

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

`listparindent` = {*rigid length*} default: 0pt

Sets the *horizontal space* indentation, beyond `list-indent`, for second and subsequent paragraphs within a list item. Internally sets the value of `\listparindent` for the current level.

`list-offset` = {*rigid length*} default: 0pt

Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.

`list-indent = {⟨rigid length⟩}` default: `labelwidth + labelsep`

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.

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

3.3 Keys for add code

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

3.4 Keys for start and resume

`start = {⟨integer | string⟩}` default: `1`

Sets the *start value* of the numbering on the current level. Internally `⟨string⟩` is passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

`resume ⟨value forbidden⟩` default: *not used*

Sets the *start* to value from the previous of the counter defined by `label` key for the “*first level*”. This `⟨key⟩` does not receive an argument. The `⟨key⟩` can be overwritten using the `start` key. If the `save-ans` key is present and `{⟨store name⟩}` exist, the numbering will continue according to this key. This key is “*only*” available for the “*first level*” of `enumext`.

3.5 Keys for multicol

`columns = {⟨integer⟩}` default: `1`

Set the *number of columns* to be used by the `multicols` 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 `multicols` 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 `multicols` will not work as expected, prefer the use of `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key.

3.6 Keys for minipage

`mini-env = {⟨rigid length⟩}` default: *not used*

Sets the *width* of the `minipage` environment on the “*right side*”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “*left side*”, taking `\linewidth` as the maximum reference value.

`mini-sep = {⟨rigid length⟩}` default: `0.3333em`

Sets the *space between* the `minipage` environment on the “*left side*” and the `minipage` environment on the “*right side*”. This separation is applied together with `\hfill`.

3.6.1 The command `\miniright`

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

3.6.2 The key `miniright`

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

`miniright` = {⟨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 the command `\centering`.

`miniright*` = {⟨code for drawing or tabular⟩} default: not used

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

4 The storage system

The entire mechanism for “storing content” it is activated according to `save-ans` key on the “first level” of `enumext` environment. Only when this *key* is “active” the `\anskey` command and the environments `keyans` and `keyanspic` are available.

<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \begin{keyans} ... \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \begin{keyanspic} ... \end{keyanspic} \end{enumext}</pre>
--	--

4.1 Keys for storage

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

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

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

Wraps the *current argument* passed `\anskey` command to referenced by {#1}. 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. 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 the double ‘{##1}’.

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

Sets the *text symbol* that will separate the current ⟨label⟩ defined by the `label` key from the ⟨optional argument⟩ (if present), when storing them in the ⟨store name⟩ defined by the `save-ans` key for the `\item*` command in the `keyans` and `keyans*` environment and for the `\anspic` command in the `keyanspic` environment. 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.

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

Sets the *symbol* to be displayed in the left margin of the “stored content” in ⟨store name⟩ set by `save-ans` key when using `show-ans` key.

`mark-pos` = {⟨left | right⟩} default: left

Sets the aligned of the *symbol* defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

4.2 Keys for internal 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.

4.3 Keys for debugging and checking

`show-ans = {⟨true | false⟩}`

default: *false*

Displays the *current* ⟨*argument*⟩ passed to `\anskey` in `enumext` environment, the current ⟨*label*⟩ for `\item*` in `keyans` environment and the current ⟨*label*⟩ for `\anspic*` in `keyanspic` environment at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown in square brackets.

`show-pos = {⟨true | false⟩}`

default: *false*

Displays the *position* occupied by the “*stored content*” by `\anskey` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment in ⟨*store name*⟩ set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans = {⟨true | false⟩}`

default: *false*

Enables the *checking answer* mechanism. This key works under the logic that each question will contain “*only one answer*”, it is intended to be used in conjunction with `no-store` key.

`no-store` ⟨*value forbidden*⟩

default: *not used*

This is a *meta-key* that does not receive an argument. This key is used in conjunction with `check-ans` and is designed to be used with nested levels of `enumext` in which the `\anskey` command will not be used.

4.4 The command \anskey

`\anskey` `\anskey{⟨content⟩}`

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

Example

- | | |
|---|---|
| <ul style="list-style-type: none"> ★ 1. Text containing our instructions or questions. <li style="margin-left: 20px;">* first answer 2. Text containing our instructions or questions. <li style="margin-left: 20px;">(a) Question. <li style="margin-left: 40px;">* second answer | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. <li style="margin-left: 20px;">* third answer 4. Text containing our instructions or questions. <li style="margin-left: 20px;">* fourth answer |
|---|---|

```
\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}
```

4.5 The environment keyans

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

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

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

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

The $\langle keys \rangle$ set in the optional argument of the environment are the same (almost) as those of the `enumext` environment and have higher precedence than those set by `\setenumext[<keyans>]{<key = val>}`. If the optional argument is not passed or the $\langle keys \rangle$ 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*)`.

4.5.1 The `\item*` in `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 $\langle store name \rangle$ set by `save-ans` key in the “first level” of the `enumext` environment. The starred version ‘`*`’ cannot be separated by spaces ‘`␣`’ from the command, i.e. `\item*` and the optional argument does “not support” verbatim content. By design it is assumed that the starred version ‘`*`’ will only appear “once” within the environment.

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

Example

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

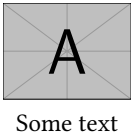
  \item Text containing a question and image.
  \begin{keyans}[nosep,mini-env={0.4\linewidth}]
    \item Choice
    \item Choice
    \item Choice
    \item Choice
    \item* [<note>] Correct choice
    \miniright
    \includegraphics[scale=0.25]{example-image-a}

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

1. Text containing a question.

(A) Choice
* (B) Correct choice
(C) Choice
(D) Choice
2. Text containing a question and image.

(A) Choice
(B) Choice
(C) Choice
(D) Choice
* (E) [note] Correct choice



4.6 The environment `keyanspic`

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

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

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

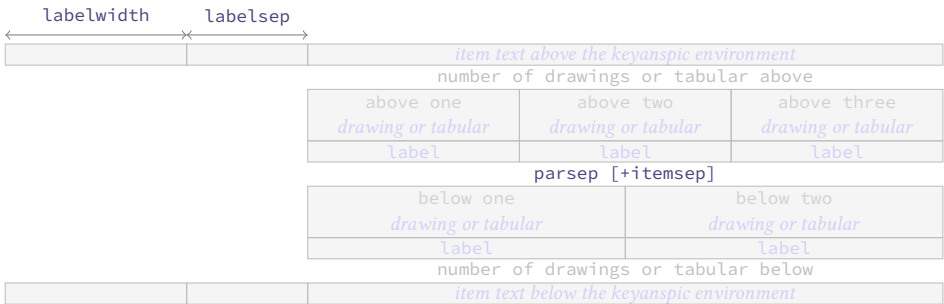


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

4.6.1 The command `\anspic`

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

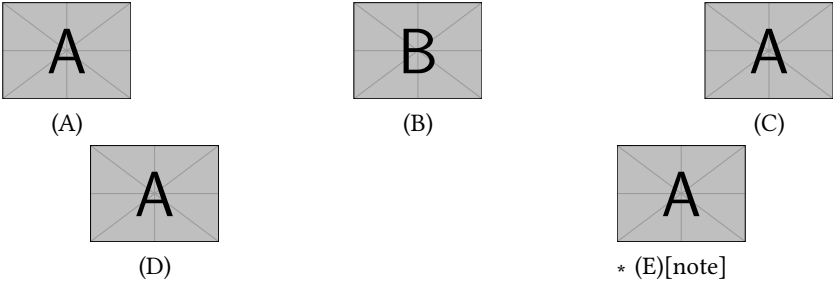
The `\anspic` command take three arguments, the *starred version* ‘`*`’ store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command `\getkeyans`

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

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

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

4.7.2 The command `\printkeyans`

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

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

Internally it places the “*stored content*” inside the `enumext` environment with default values for `label` key are the same as those of the `enumext` environment along with the keys: `nosep`, `first=\small`, `font=\small` for all levels, except for the first one that adds the `columns=2` key.

The optional argument allows to handle the *keys* “on the first level” of the `enumext` environment encapsulated by the command. If need to pass options for nested levels use `\setenumext[<print , level>]{<store name>}`.

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1] $3(x + y + z)$
2. True False

(a) \LaTeXe is cool?

[2] Very True!
3. Related to Linux

(a) You use linux?
- [3] Yes

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

*
2. (a) Very True!

*
3. (a) Yes

*

(b) i. very good

*

ii. obsolete

*


5 Full examples

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

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

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

Example 1

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

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

A 36 km/h.

B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

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

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

B

C

D

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


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

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

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

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

B

C

D

36 km/h.

360 km/h.

27,8 km/h.

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

B

C

D

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

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

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

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

B

C

D

36 km/h.

360 km/h.

27,8 km/h.

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

B

C

D

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

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

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

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

1. B
2. A
3. B
4. A
- *

*

*

*

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

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

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

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

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

A

 value

B

 value

C

 value

D

 value

E

 value
4. Question with image and label below:



A



B



C



D



E

5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

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

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

 Factor $x^2 - 2x + 1$

2

 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

3

 True False

(a)

 $\alpha > \delta$

(b)

~~ETX~~ze is cool?

4

 Related to Linux

(a)

 You use linux?

(b)

 Usually uses the package manager?

(c)

 Rate the following package and class

i.

 xsim-exam

ii.

 xsim

iii.

 exsheets

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

1. $(x - 1)^2$
2. $3(x + y + z)$
3. (a) False
- (b) Very True!
4. (a) Yes
- (b) Yes, dnf
- (c) i. doesn't exist for now :(
- ii. very good
- iii. obsolete

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

Fake itemize environment

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

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

Fake description environment

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

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

- Something** A short one-line description.

This is an entry without a label.

Something A short one-line description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

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

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

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

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

7 References

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

8 Change history

v1.0 2024-05-14 – First public release.

9 Index of Documentation

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

C

Document class:

article

book

exam

letter

report

\columnbreak

\columnsep

Commands provide by enumext:

\anskey

\anspic*

\anspic

\getkeyans

\item*

\item

\miniright

\printkeyans

\setenumext

Counters defined by enumext:

enumXiii

enumXii

enumXiv

enumXi

enumXviii

enumXvii

enumXvi

enumXv

E

Environments provide by enumext:

enumext*

enumext

keyans*

keyanspic

keyans

Environments:

enumerate

list

minipage

multicols

I

\item

\itemsep

K

Keys for environments provide by enumext:

above*

above

after

align

before*

before

below*

below

check-ans

columns-sep

columns

first

font

item-pos*

item-sym*

itemindent

itemsep

labelsep

labelwidth

label

list-indent

list-offset

listparindent

mark-ans

mark-pos

mark-ref

mini-env

mini-sep

miniright*

miniright

no-store

noitemsep

nosep

parsep

partopsep

ref

resume

rightmargin

save-ans

save-ref

save-sep

show-ans

show-length

show-pos

start

topsep

widest

wrap-ans

wrap-label*

wrap-label

wrap-opt

L

\label

Labels provide by enumext:

\Alph*

\Roman*

\alph*

\arabic*

\roman*

\labelsep

\labelwidth

\linewidth

\listparindent

P

Packages:

enumerate

enumext

enumitem

footnotehyper

hyperref

l3prop

l3seq

©2024 by Pablo González L

20 / 115

multicol	1, 2, 5, 19	\ref	5
xsim	3	\rightmargin	8
\parsep	8		
\partopsep	8		
R		T	
\raggedcolumns	5	\topsep	8

10 Implementation

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

- 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.

10.1 General conventions

Variables containing `i`, `ii`, `iii` and `iv` are associated by level with the `enumext` environment, variables containing `v` are associated with the `keyans` environment, variables containing `vi` are associated with the `keyanspic` environment, variables containing `vii` are associated with the `enumext*` environment and variables containing `viii` are associated with the `keyans*` environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital “X”.

The temporary function `__enumext_tmp:n` is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

10.2 Initial set up

Start the DocStrip guards.

```
1 <*package>
```

Identify the internal prefix (L^AT_EX3 DocStrip convention) for l3doc class.

```
2 <@@=enumext>
```

10.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L^AT_EX to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-14}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

10.4 Definition of variables

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

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

```
20 \int_new:N \l__enumext_level_int
21 \int_new:N \l__enumext_level_h_int
22 \int_new:N \l__enumext_keyans_level_int
23 \int_new:N \l__enumext_keyans_level_h_int
24 \int_new:N \l__enumext_keyans_pic_level_int
25 \bool_new:N \l__enumext_starred_bool
26 \bool_new:N \g__enumext_starred_bool
```

```

27 \bool_new:N \l__enumext_standar_bool
28 \bool_new:N \g__enumext_standar_bool
29 \bool_new:N \l__enumext_keyans_env_bool

```

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

```

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

```

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

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

The initial values of these variables are set by the function `__enumext_define_counters:Nn` and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§10.8).

```

30 \cs_set_protected:Npn \__enumext_tmp:n #1
31 {
32   \tl_new:c { \l__enumext_counter_#1_tl }
33 }
34 \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.)

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\g__enumext_item_symbol_tl
\g__enumext_series_standar_default_tl
\g__enumext_series_starred_default_tl

```

The boolean variable `\l__enumext_resume_bool` is used by `resume` key, the value from which the environment’s will start is stored in the integer variable `\g__enumext_resume_int` (§10.21). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.26).

```

35 \int_new:N \g__enumext_resume_int
36 \int_new:N \g__enumext_resume_vii_int
37 \tl_new:N \g__enumext_item_symbol_tl
38 \tl_new:N \g__enumext_series_standar_default_tl
39 \tl_new:N \g__enumext_series_starred_default_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` (§10.12) and `label` (§10.10) keys.

```

40 \dim_new:N \l__enumext_current_widest_dim
41 \tl_new:N \g__enumext_counter_styles_tl
42 \tl_new:N \g__enumext_widest_label_tl
43 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§10.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§10.30) which determines the internal values for `\leftmargin` and `\itemindent`.

```

44 \cs_set_protected:Npn \__enumext_tmp:n #1
45 {
46   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
47   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
48   \dim_new:c { \l__enumext_leftmargin_#1_dim }
49   \dim_new:c { \l__enumext_itemindent_#1_dim }
50 }
51 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

```

Internal variables used by `columns` key §10.18).

```

52 \cs_set_protected:Npn \__enumext_tmp:n #1
53 {
54   \skip_new:c { \l__enumext_multicols_above_#1_skip }
55   \skip_new:c { \l__enumext_multicols_below_#1_skip }
56 }
57 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

Internal variables used by `\miniright` command (§10.19.4) and the keys `miniright`, `miniright*`, `mini-env` and `mini-sep` (§10.17, §10.19).

```
58 \int_new:N \g__enumext_minipage_stat_int
59 \skip_new:N \l__enumext_minipage_left_skip
60 \skip_new:N \l__enumext_minipage_right_skip
61 \skip_new:N \l__enumext_minipage_after_skip
62 \skip_new:N \g__enumext_minipage_right_skip
63 \skip_new:N \g__enumext_minipage_after_skip
64 \cs_set_protected:Npn \__enumext_tmp:n #1
65 {
66   \dim_new:c { \l__enumext_minipage_left_#1_dim }
67   \bool_new:c { \l__enumext_minipage_active_#1_bool }
68 }
69 \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 (§10.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§10.10). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```
70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \bool_new:c { \l__enumext_wrap_label_#1_bool }
73   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
74   \int_new:c { \l__enumext_start_#1_int }
75   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
76   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
77   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
78   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
79   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
80 }
81 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

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

The variable `\l__enumext_store_name_tl` sets the name for the storage in `⟨sequence⟩` and `⟨prop list⟩`, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the `check-ans` key (§10.21).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.28.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§10.34.1) for the `keyanspic` environment.

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

```
82 \bool_new:N \l__enumext_store_active_bool
83 \tl_new:N \l__enumext_store_name_tl
84 \tl_new:N \g__enumext_store_name_tl
85 \tl_new:N \l__enumext_store_anskey_arg_tl
86 \int_new:N \l__enumext_store_columns_join_int
87 \tl_new:N \l__enumext_store_keyans_label_tl
88 \tl_new:N \l__enumext_store_keyans_item_opt_tl
89 \tl_new:N \l__enumext_keyans_item_opt_tl
90 \tl_new:N \l__enumext_keyans_tmpa_tl
91 \tl_new:N \l__enumext_keyans_tmpb_tl
92 \dim_new:N \l__enumext_keyans_tmpa_dim
```

(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` (§10.39).

```
93 \tl_new:N \l__enumext_setkey_tmpa_tl
94 \tl_new:N \l__enumext_setkey_tmpb_tl
95 \int_new:N \l__enumext_setkey_tmpa_int
96 \seq_new:N \l__enumext_setkey_tmpa_seq
97 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

```
\l__enumext_store_opt_X_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_columns_X_bool
\l__enumext_store_columns_X_int
\l__enumext_store_columns_sep_X_bool
\l__enumext_store_columns_sep_X_dim
\l__enumext_store_upper_level_X_bool
```

Internal variables used by [$\langle key = val \rangle$] in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.38) and the keys `columns*` and `columns-sep*`.

```
98 \cs_set_protected:Npn \l__enumext_tmp:n #1
99 {
100   \tl_new:c { \l__enumext_store_opt_#1_tl }
101   \tl_new:c { \l__enumext_print_keyans_#1_tl }
102   \bool_new:c { \l__enumext_store_columns_#1_bool }
103   \int_new:c { \l__enumext_store_columns_#1_int }
104   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
105   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
106   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
107 }
108 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

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

```
\l__enumext_show_answer_bool
\l__enumext_show_position_bool
\l__enumext_mark_ref_sym_tl
\l__enumext_mark_answer_sym_tl
\l__enumext_mark_position_str
```

Internal variables for “storage system” mechanism used by `\anskey` (§10.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.23).

```
109 \bool_new:N \l__enumext_show_answer_bool
110 \bool_new:N \l__enumext_show_position_bool
111 \tl_new:N \l__enumext_mark_ref_sym_tl
112 \tl_new:N \l__enumext_mark_answer_sym_tl
113 \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 (§10.34.2).

```
114 \seq_new:N \l__enumext_keyans_pic_body_seq
115 \dim_new:N \l__enumext_keyans_pic_width_dim
116 \int_new:N \l__enumext_keyans_pic_above_int
117 \int_new:N \l__enumext_keyans_pic_below_int
118 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

```
\l__enumext_store_ans_bool
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int
```

Internal variables used by “check answer” mechanism (§10.22) controlled by the `check-ans` and `no-store` keys.

```
119 \bool_new:N \l__enumext_store_ans_bool
120 \bool_new:N \l__enumext_check_ans_bool
121 \bool_new:N \g__enumext_check_ans_show_bool
122 \bool_new:N \g__enumext_check_ans_show_h_bool
123 \tl_new:N \g__enumext_check_ans_item_tl
124 \int_new:N \g__enumext_count_item_anskey_int
125 \int_new:N \g__enumext_count_item_number_int
126 \int_new:N \g__enumext_standar_star_env_int
127 \int_new:N \g__enumext_starred_star_env_int
128 \int_new:N \g__enumext_starred_keyans_star_env_int
129 \int_new:N \g__enumext_standar_keyans_star_env_int
130 \int_new:N \g__enumext_standar_keyans_pic_star_env_int
```

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

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

```
131 \bool_new:N \l__enumext_hyperref_bool
132 \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.

```
133 \tl_new:N \l__enumext_newlabel_arg_one_tl
134 \tl_new:N \l__enumext_newlabel_arg_two_tl
```

```

135 \tl_new:N \l__enumext_store_write_aux_file_tl
136 \cs_set_protected:Npn \__enumext_tmp:n #1
137 {
138   \tl_new:c { l__enumext_label_copy_#1_tl }
139 }
140 \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`

Internal variables used for redefinition of `\footnote`.

`\g__enumext_footnote_arg_seq`
`\g__enumext_footnote_int_seq`

```

141 \int_new:N \g__enumext_footnote_int
142 \seq_new:N \g__enumext_footnote_arg_seq
143 \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`.)

`\c__enumext_counter_style_tl`
`\l__enumext_ref_key_arg_tl`
`\l__enumext_ref_aux_tl`
`\l__enumext_the_counter_X_tl`
`\l__enumext_counter_style_for_ref_X_tl`

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

```

144 \tl_const:Nn \c__enumext_counter_style_tl
145 { { arabic } { roman } { Roman } { alph } { Alph } }
146 \tl_new:N \l__enumext_ref_key_arg_tl
147 \tl_new:N \l__enumext_ref_aux_tl
148 \cs_set_protected:Npn \__enumext_tmp:n #1
149 {
150   \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
151   \tl_new:c { l__enumext_the_counter_#1_tl }
152   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
153 }
154 \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.)

`\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.

```

155 \cs_set_protected:Npn \__enumext_tmp:n #1
156 {
157   \bool_new:c { l__enumext_item_starred_#1_bool }
158   \int_new:c { l__enumext_item_column_pos_#1_int }
159   \int_new:c { g__enumext_item_count_all_#1_int }
160   \int_new:c { l__enumext_joined_item_#1_int }
161   \int_new:c { l__enumext_joined_item_aux_#1_int }
162   \int_new:c { l__enumext_tmpa_#1_int }
163   \box_new:c { l__enumext_item_text_#1_box }
164   \dim_new:c { l__enumext_joined_width_#1_dim }
165   \dim_new:c { l__enumext_item_width_#1_dim }
166   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
167   \str_new:c { l__enumext_align_label_#1_str }
168   \bool_new:c { g__enumext_minipage_active_#1_bool }
169   \tl_new:c { g__enumext_miniright_code_#1_tl }
170   \bool_new:c { g__enumext_minipage_center_#1_bool }
171   \dim_new:c { g__enumext_minipage_right_#1_dim }
172   \skip_new:c { g__enumext_minipage_right_#1_skip }
173 }
174 \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`.

```

175 \clist_const:Nn \c__enumext_all_envs_clist
176 {
177   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
178   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
179 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 Some utility functions

`__enumext_at_begin_document:n`

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

```
180 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
181 {
182   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
183 }
```

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

`__enumext_after_env:nn`

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

```
184 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
185 {
186   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
187 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
188 \cs_new:Nn \__enumext_level:
189 {
190   \int_to_roman:n { \l__enumext_level_int }
191 }
```

(End of definition for `__enumext_level:.`)

`__enumext_if_is_int:nT`

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

`__enumext_if_is_int:nF`

```
192 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
193 {
194   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
195   { \prg_return_true: }
196   { \prg_return_false: }
197 }
```

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

`__enumext_show_length:nnn`

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

```
198 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
199 {
200   * ~ #2
201   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
202   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
203 }
```

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
204 \cs_set_protected:Nn \__enumext_zero_count_level:
205 {
206   \cs_set_protected:Npn \__enumext_tmp:n ##1
207   {
208     \int_gzero:c { g__enumext_count_level_##1_int }
209   }
210   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
211 }
```

(End of definition for `__enumext_zero_count_level:.`)

`__enumext_current_env:`

The function `__enumext_current_env:` will set the global variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` with which we will distinguish whether the environments `enumext` and `enumext*` are nested in each other.

```
212 \cs_new_protected:Nn \__enumext_current_env:
213 {
214   \str_case:en { \@currenvir }
215   {
216     {enumext}
```

```

217         {
218             \bool_lazy_and:nnT
219             { \bool_not_p:n { \g__enumext_standar_bool } }
220             { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
221             {
222                 \bool_gset_true:N \g__enumext_standar_bool
223                 \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
224                 \typeout{working-on-enumext}
225             }
226         }
227     {enumext*}
228     {
229         \bool_lazy_and:nnT
230         { \bool_not_p:n { \g__enumext_starred_bool } }
231         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
232         {
233             \bool_gset_true:N \g__enumext_starred_bool
234             \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
235             \typeout{working-on-enumext*}
236         }
237     }
238 }
239 }

```

(End of definition for `__enumext_current_env:`.)

10.6 Copying list and minipage environments

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

```

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

```

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

```

\__enumext_start_list:nn \__enumext_stop_list: \__enumext_item_std:w
240 \__enumext_at_begin_document:n
241 {
242     \cs_new_eq:NN \__enumext_start_list:nn \list
243     \cs_new_eq:NN \__enumext_stop_list: \endlist
244     \cs_new_eq:NN \__enumext_item_std:w \item
245 }

```

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

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

```

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

```

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

```

\__enumext_minipage:w \__enumext_endminipage:
246 \__enumext_at_begin_document:n
247 {
248     \cs_new_eq:NN \__enumext_minipage:w \minipage
249     \cs_new_eq:NN \__enumext_endminipage: \endminipage
250 }

```

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

10.7 Compatibility with hyperref and footnotehyper

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

```

251 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
252 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

`__enumext_after_hyperref:` The function `__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hy
__enumext_hypertarget:nn hyperref_bool to “true” if the package is loaded. At this point we will use the public macro
__enumext_phantomsection: \IfHyperBoolean to determine if the hyperfootnotes=true key is present, if so, we set the state
of the boolean variable __enumext_footnotes_key_bool to “true”.`

```

253 \cs_new_protected:Nn \__enumext_after_hyperref:
254 {
255   \IfPackageLoadedTF { hyperref }
256   {
257     \msg_info:nnn { enumext } { package-load } { hyperref }
258     \bool_set_true:N \l__enumext_hyperref_bool
259     \IfHyperBoolean{hyperfootnotes}
260     {
261       \typeout{hyperfootnotes=true}
262       \bool_set_true:N \l__enumext_footnotes_key_bool
263     }
264     { \typeout{hyperfootnotes=false} }
265   }
266   { }

```

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`.

```

267 \bool_if:NT \l__enumext_footnotes_key_bool
268 {
269   \IfPackageLoadedTF { footnotehyper }
270   {
271     \msg_info:nnn { enumext } { package-load } { footnotehyper }
272   }
273   {
274     \typeout{No ~ footnotehyper ~ load}
275     \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
276     \bool_set_false:N \l__enumext_footnotes_key_bool
277   }
278 }

```

The functions `__enumext_hypertarget:nn` and `__enumext_phantomsection:` correspond to the internal copies of `\hypertarget` and `\phantomsection`. If the boolean variable `\l__enumext_hy
hyperref_bool is false the functions __enumext_hypertarget:nn and __enumext_phantomsection: will be disabled.`

```

279 \bool_if:NTF \l__enumext_hyperref_bool
280 {
281   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
282   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
283 }
284 {
285   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
286   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
287 }
288 }

```

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

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

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

```

289 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
290 {
291   \protected@write \@auxout { }
292   {
293     \token_to_str:N \newlabel {#1}
294     {
295       {#2}
296       \bool_if:NT \l__enumext_hyperref_bool
297       { { \thepage } {#2} {#1} }
298       { }
299     }

```

```

300     }
301     \__enumext_hypertarget:nn {#1} { }
302     \__enumext_phantomsection:
303 }

```

(End of definition for __enumext_newlabel:nn.)

10.8 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

The arguments taken by the function are:

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

```

304 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
305 {
306     \cs_if_exist:cTF { c@ #2 }
307     { \msg_fatal:nnn { enumext } { counters } { #2 } }
308     {
309         \tl_set:Nn #1 { #2 }
310         \newcounter { #2 }
311     }
312 }

```

(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      313 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     314 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    315 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     316 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXvii    317 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXviii   318 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
            319 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
            320 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

10.9 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```

321 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
322 {
323     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
324     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
325 }
326 \__enumext_register_counter_style:Nn \arabic { 0 }
327 \__enumext_register_counter_style:Nn \Alph { M }
328 \__enumext_register_counter_style:Nn \alph { m }
329 \__enumext_register_counter_style:Nn \Roman { VIII }
330 \__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.

```

331 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
332 {
333     \hbox_set:Nn \__enumext_label_width_by_box {#2}
334     \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
335 }
336 \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 create 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.

```

337 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
338 {
339   \tl_clear_new:N #1
340   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
341   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
342   \tl_map_inline:Nn \g__enumext_counter_styles_tl
343     {
344       \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
345       \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
346         { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
347     }
348   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
349     { \tl_use:N \g__enumext_widest_label_tl }
350   \tl_set_eq:cN { the #2 } #1
351 }
352 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

10.10 Setting keys associated with label

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

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

```

353 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
354 {
355   \keys_define:nn { enumext / #1 }
356   {
357     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
358     font      .value_required:n = true,
359     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
360     labelsep  .initial:n   = {0.3333em},
361     labelsep  .value_required:n = true,
362     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
363     labelwidth .value_required:n = true,
364     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
365     wrap-label .initial:n   = {##1},
366     wrap-label .value_required:n = true,
367     wrap-label* .code:n = {
368       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
369       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
370     },
371     wrap-label* .value_required:n = true,
372   }
373 }
374 \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.

```

375 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
376 {
377   \keys_define:nn { enumext / #1 }
378   {
379     align .choice:,
380     align / left .code:n =
381       {
382         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
383         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
384       },
385     align / right .code:n =
386       {
387         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
388         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
389       }
390   }
391 }

```

```

389         },
390         align / center .code:n =
391         {
392             \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
393             \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
394         },
395         align .initial:n = left,
396         align .value_required:n = true,
397     }
398 }
399 \clist_map_inline:nn
400 {
401     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
402 }
403 { \__enumext_tmp:nn #1 }

```

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

```

404 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
405 {
406     \keys_define:nn { enumext / #1 }
407     {
408         align .choice:,
409         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
410         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
411         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
412         align .initial:n = left,
413         align .value_required:n = true,
414     }
415 }
416 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

10.11 Setting label and ref keys

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

```

417 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
418 {
419     \tl_map_inline:Nn \c__enumext_counter_style_tl
420     {
421         \regex_replace_once:nnN { \c{##1}\* }
422         { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
423     }
424 }

```

(End of definition for `__enumext_regex_label_ref_key:`.)

`__enumext_set_label_ref:n` The `__enumext_set_label_ref:n` function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable `\l__enumext_the_counter_X_tl` according to the command created for *each counter*, apply the `regex` function `__enumext_regex_label_ref_key:` and then renew the command and save it in the variable `\l__enumext_counter_style_for_ref_X_tl`.

```

425 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
426 {
427     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
428     \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
429     \__enumext_regex_label_ref_key:
430     \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
431     \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
432     {
433         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
434         { \exp_not:V \l__enumext_ref_key_arg_tl }
435     }
436 }

```

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

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

```

437 \cs_new_protected:Nn \__enumext_use_key_ref:
438 {
439   \tl_if_empty:cF { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
440   {
441     \tl_use:c { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
442   }
443 }

```

(End of definition for `__enumext_use_key_ref:`.)

For `enumext*` and `keyans*` environments the situation is a bit different since `hyperref` interferes here (I am not clear why), so we will define a new function to execute the task.

To handle that we will look at the nesting level of the starred environments, later I will run the constraint functions to make everything OK.

`__enumext_set_label_ref_h:n` The `__enumext_set_label_ref_h:n` function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable `\l__enumext_the_counter_X_tl` according to the command created for *each counter*, apply the `regex` function `__enumext_regex_label_ref_key:` and then renew the command and save it in the variable `\l__enumext_counter_style_for_ref_X_tl`.

```

444 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
445 {
446   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
447   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
448   {
449     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
450     \__enumext_regex_label_ref_key:
451     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
452     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
453     {
454       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
455       { \exp_not:V \l__enumext_ref_key_arg_tl }
456     }
457   }
458   {
459     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
460     \__enumext_regex_label_ref_key:
461     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
462     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
463     {
464       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
465       { \exp_not:V \l__enumext_ref_key_arg_tl }
466     }
467   }
468 }

```

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

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

```

469 \cs_new_protected:Nn \__enumext_use_key_ref_h:
470 {
471   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
472   {
473     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
474     {
475       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
476     }
477   }
478   {
479     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
480     {
481       \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
482     }
483   }
484 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

label Here we set the default $\langle labels \rangle$ of the four levels of `enumext` environment, along with the default value
ref for `labelwidth` key.

```

\l__enumext_label_i_tl 485 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 486 {
\l__enumext_label_iii_tl 487   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 488   {
489     label .code:n = {
490       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
491       { l__enumext_counter_#2_tl } {##1}
492       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
493       \l__enumext_current_widest_dim
494     },
495     label .initial:n = #3,
496     label .value_required:n = true,
497     ref .code:n = \__enumext_set_label_ref:n {##1},
498     ref .value_required:n = true,
499   }
500 }
501 \__enumext_tmp:nnn { level-1 } { i } { \arabic*.}
502 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
503 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
504 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

10.11.2 Define and set label key for enumext* and keyans* environments

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

```

\l__enumext_label_vii_tl 505 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 506 {
507   \keys_define:nn { enumext / #1 }
508   {
509     label .code:n = {
510       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
511       { l__enumext_counter_#2_tl } {##1}
512       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
513       \l__enumext_current_widest_dim
514     },
515     label .initial:n = #3,
516     label .value_required:n = true,
517     ref .code:n = \__enumext_set_label_ref_h:n {##1},
518     ref .value_required:n = true,
519   }
520 }
521 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
522 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*) }

```

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

10.11.3 Define and set label key for keyans and keyanspic environment

label Here we set the default $\langle label \rangle$ for `keyans` and `keyanspic` environment, along with the default value for
`labelwidth`. The `keyanspic` environment use the same $\langle label \rangle$ as the `keyans` environment.

\l__enumext_label_v_tl Define and set `label` key for `keyans` environment.

```

\l__enumext_label_vi_tl 523 \keys_define:nn { enumext / keyans }
524 {
525   label .code:n = {
526     \__enumext_label_style:cvn { l__enumext_label_v_tl }
527     { l__enumext_counter_v_tl } {#1}
528     \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
529     \l__enumext_current_widest_dim
530     \__enumext_label_style:cvn { l__enumext_label_vi_tl }
531     { l__enumext_counter_vi_tl } {#1}
532     \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
533     \l__enumext_current_widest_dim
534   },
535   label .initial:n = (\Alph*),
536   label .value_required:n = true,
537 }

```

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

10.12 Setting start and widest keys

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

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

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

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

```
538 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
539 {
540   \__enumext_if_is_int:nTF { #3 }
541   {
542     \int_set:Nn #2 {#3}
543   }
544   {
545     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
546     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
547     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
548     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
549   }
550 }
551 \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.

```
552 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
553 {
554   \__enumext_if_is_int:nTF {#4}
555   {
556     \setcounter{enumX#1} { #4 }
557   }
558   {
559     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
560     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
561     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
562     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
563   }
564   \__enumext_label_width_by_box:cv
565   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
566 }
567 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

```
start
widest
\l__enumext_start_X_int
```

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

```
568 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
569 {
570   \keys_define:nn { enumext / #1 }
571   {
572     start .code:n = {
573       \__enumext_start_from:ccn
574       { \l__enumext_label_#2_tl }
575       { \l__enumext_start_#2_int } {##1}
576     },
577     start .initial:n = 1,
578     widest .code:n = {
579       \__enumext_widest_from:nccn {#2}
580       { \l__enumext_label_#2_tl }
581       { \l__enumext_labelwidth_#2_dim } {##1}
582     },

```

```

583         widest .value_required:n = true,
584         start .value_required:n = true,
585     }
586 }
587 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.13 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
588 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
589 {
590     \keys_define:nn { enumext / #1 }
591     {
592         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
593         topsep .initial:n = {#3},
594         topsep .value_required:n = true,
595         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
596         partopsep .initial:n = {#4},
597         partopsep .value_required:n = true,
598         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
599         parsep .initial:n = {#5},
600         parsep .value_required:n = true,
601         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
602         itemsep .initial:n = {#6},
603         itemsep .value_required:n = true,
604         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
605         noitemsep .value_forbidden:n = true,
606         nosepp .meta:n = {
607             itemsep = 0pt, parsep= 0pt,
608             topsep = 0pt, partopsep = 0pt,
609         },
610         nosepp .value_forbidden:n = true,
611     }
612 }

```

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

```

613 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
614 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
615 { 4.0pt plus 2.0pt minus 1.0pt }
616 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
617 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
618 { 2.0pt plus 1.0pt minus 1.0pt }
619 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
620 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
621 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
622 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
623 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
624 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
625 { 2.0pt plus 1.0pt minus 1.0pt }
626 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
627 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
628 { 4.0pt plus 2.0pt minus 1.0pt }
629 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
630 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
631 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

10.14 Setting keys for horizontal spaces

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

```

listparindent
list-offset
list-indent
632 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
633 {
634     \keys_define:nn { enumext / #1 }
635     {
636         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
637         itemindent .value_required:n = true,
638         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },

```

```

639     rightmargin .value_required:n = true,
640     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
641     listparindent .value_required:n = true,
642     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
643     list-offset .value_required:n = true,
644     list-indent .code:n =
645         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
646         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {#1},
647     list-indent .value_required:n = true,
648 }
649 }
650 \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.

```

651 \cs_set_protected:Npn \__enumext_tmp:n #1
652 {
653     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
654 }
655 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

10.14.1 Functions for setting the fake *itemindent*

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

```

656 \cs_set_protected:Nn \__enumext_fake_item:
657 {
658     \dim_compare:nNnT
659     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
660     >
661     { \c_zero_dim }
662     {
663         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
664         {
665             \exp_not:N \mode_leave_vertical:
666             \exp_not:n { \skip_horizontal:n
667                 { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
668             \ignorespaces
669         }
670     }
671 }
672 \cs_set_protected:Nn \__enumext_keyans_fake_item:
673 {
674     \dim_compare:nNnT
675     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
676     {
677         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
678         {
679             \exp_not:N \mode_leave_vertical:
680             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
681         }
682     }
683 }
684 \cs_set_protected:Nn \__enumext_fake_item_vii:
685 {
686     \dim_compare:nNnT
687     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
688     {
689         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
690         {
691             \exp_not:N \mode_leave_vertical:
692             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
693         }
694     }
695 }
696 \cs_set_protected:Nn \__enumext_fake_item_viii:
697 {
698     \dim_compare:nNnT

```

```

699     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
700     {
701         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
702         {
703             \exp_not:N \mode_leave_vertical:
704             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
705         }
706     }
707 }

```

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

10.15 Setting show-length key

`show-length` Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to “true” and calls the function `__enumext_show_length:nnn` which prints all the values of the “vertical” and “horizontal” parameters calculated and used.

```

708 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
709 {
710     \keys_define:nn { enumext / #1 }
711     {
712         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
713         show-length .initial:n = false,
714     }
715 }
716 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

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

```

before* 717 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    718 {
first    719     \keys_define:nn { enumext / #1 }
          720     {
          721         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
          722         before .value_required:n = true,
          723         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
          724         before* .value_required:n = true,
          725         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
          726         after .value_required:n = true,
          727         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
          728         first .value_required:n = true,
          729     }
          730 }
          731 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16.1 Functions for before, after and first keys in enumext

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

```

732 \cs_new_protected:Nn \__enumext_before_args_exec:
733 {
734     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
735 }

```

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

```

736 \cs_new_protected:Nn \__enumext_before_keys_exec:
737 {
738     \tl_use:c { \l__enumext_before_no_starred_key_ \__enumext_level: _tl }
739 }

```

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

```

740 \cs_new_protected:Nn \__enumext_after_stop_list:
741 {

```

```

742     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
743   }

```

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

```

744 \cs_new_protected:Nn \__enumext_after_args_exec:
745 {
746   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
747 }

```

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

10.16.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
748 \cs_new_protected:Nn \__enumext_before_args_exec_v:
749 {
750   \tl_use:N \l__enumext_before_starred_key_v_tl
751 }

```

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

```

752 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
753 {
754   \tl_use:N \l__enumext_before_no_starred_key_v_tl
755 }

```

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

```

756 \cs_new_protected:Nn \__enumext_after_stop_list_v:
757 {
758   \tl_use:N \l__enumext_after_stop_list_v_tl
759 }

```

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`.

```

760 \cs_new_protected:Nn \__enumext_after_args_exec_v:
761 {
762   \tl_use:N \l__enumext_after_list_args_v_tl
763 }

```

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

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

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

```

\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
764 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
765 {
766   \tl_use:N \l__enumext_before_starred_key_vii_tl
767 }
768 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
769 {
770   \tl_use:N \l__enumext_before_starred_key_viii_tl
771 }

```

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

```

772 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
773 {
774   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
775 }
776 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
777 {
778   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
779 }

```


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

```

780 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
781 {
782   \tl_use:N \__enumext_after_stop_list_vii_tl
783 }
784 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
785 {
786   \tl_use:N \__enumext_after_stop_list_viii_tl
787 }

```

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`.

```

788 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
789 {
790   \tl_use:N \__enumext_after_list_args_vii_tl
791 }
792 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
793 {
794   \tl_use:N \__enumext_after_list_args_viii_tl
795 }

```

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

10.17 Setting keys for multicols and minipage

`mini-env` The default value of the `columns-sep` key is handled by the state of the boolean variable `\l__enumext_columns_sep_X_bool` which is handled in the internal definition of the `enumext` and `keyans` environments.

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

`columns-sep`

`columns`

```

796 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
797 {
798   \keys_define:nn { enumext / #1 }
799   {
800     mini-env   .dim_set:c = { \l__enumext_minipage_right_#2_dim },
801     mini-env   .value_required:n = true,
802     mini-sep   .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
803     mini-sep   .initial:n = 0.3333em,
804     mini-sep   .value_required:n = true,
805     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
806     columns-sep .value_required:n = true,
807     columns    .int_set:c = { \l__enumext_columns_#2_int },
808     columns    .initial:n = 1,
809     columns    .value_required:n = true,
810   }
811 }
812 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

For `enumext*` and `keyans*` environments the situation is a bit different, the default value for `columns` key are `2` and the command `\miniright` is not available, so we will add the keys `miniright` and `miniright*` to implement support for `minipage`.

```

813 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
814 {
815   \keys_define:nn { enumext / #1 }
816   {
817     columns    .initial:n = 2,
818     miniright  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
819     miniright  .value_required:n = true,
820     miniright* .code:n = {
821       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
822       \keys_set:nn { enumext / #1 } { miniright = {##1} }
823     },
824     miniright* .value_required:n = true,
825   }
826 }
827 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for 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 are “stretched”, possibly due to the use of `\raggedcolumns` and this affects the lower space when closing the environment, which is “smaller” than expected. My attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

10.18.1 Adjustment of vertical spaces for 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*.

```

828 \cs_new_protected:Nn \__enumext_multi_set_vskip:
829 {
830   \skip_set:cn { \__enumext_multicol_above_ \__enumext_level: } _skip }
831   {
832     \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip }
833   }
834   \skip_set:cn { \__enumext_multicol_below_ \__enumext_level: } _skip }
835   {
836     \skip_use:c { \__enumext_topsep_ \__enumext_level: } _skip }
837   }
838   \__enumext_add_pre_parsep:
839 }
```

(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*.

```

840 \cs_new_protected:Nn \__enumext_add_pre_parsep:
841 {
842   \int_case:nn { \__enumext_level_int }
843   {
844     { 2 }{
845       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
846       {
847         \skip_add:Nn \__enumext_multicol_above_ii_skip { \__enumext_parsep_i_skip }
848       }
849     }
850     { 3 }{
851       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip }
852       {
853         \skip_add:Nn \__enumext_multicol_above_iii_skip { \__enumext_parsep_ii_skip }
854       }
855     }
856     { 4 }{
857       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip }
858       {
859         \skip_add:Nn \__enumext_multicol_above_iv_skip { \__enumext_parsep_iii_skip }
860       }
861     }
862   }
```

```

862     }
863 }

```

(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)*.

```

864 \cs_new_protected:Nn \__enumext_multi_addvspace:
865 {
866   \__enumext_multi_set_vskip:
867   \mode_if_vertical:T
868   {
869     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
870     {
871       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
872     }
873     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
874     {
875       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
876     }
877   }
878   \par\nopagebreak
879   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
880 }

```

(End of definition for `__enumext_multi_addvspace:`)

10.18.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

881 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
882 {
883   \skip_set:Nn \__enumext_multicols_above_v_skip
884   {
885     \__enumext_topsep_v_skip
886   }
887   \skip_set:Nn \__enumext_multicols_below_v_skip
888   {
889     \__enumext_topsep_v_skip
890   }
891 }
892 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
893 {
894   \__enumext_keyans_multi_set_vskip:
895   \mode_if_vertical:T
896   {
897     \skip_add:Nn \__enumext_multicols_above_v_skip
898     {
899       \skip_use:N \__enumext_partopsep_v_skip
900     }
901     \skip_add:Nn \__enumext_multicols_below_v_skip
902     {
903       \skip_use:N \__enumext_partopsep_v_skip
904     }
905   }
906   \par\nopagebreak
907   \addvspace{ \__enumext_multicols_above_v_skip }
908 }

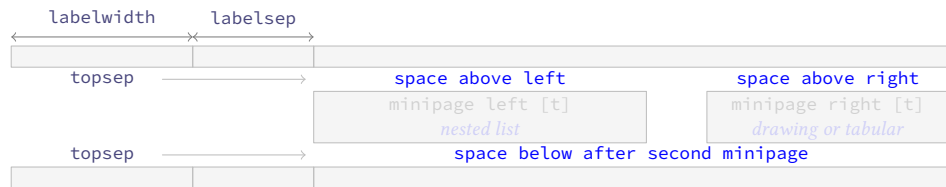
```

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

10.19 Adjustment of vertical spaces for minipage

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

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

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

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

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

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

```

909 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
910 {
911     \__enumext_minipage:w [ t ] { #1 }
912     \legacy_if_gset_false:n { @minipage }
913     \vspace { 0pt }
914 }
915 { \__enumext_endminipage: }
```

(End of definition for `__enumext_mini_env*`.)

10.19.1 Adjustment of vertical spaces for minipage in enumext

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

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

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

```

916 \cs_new_protected:Nn \__enumext_mini_set_vskip:
917 {
918     \int_compare:nNnTF
919     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
920     {
```

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

```

921     \skip_if_eq:nnTF
922     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
923     {
924         \skip_set:Nn \l__enumext_minipage_left_skip
925         {
926             -0.150\box_dp:N \strutbox
927         }
928         \skip_set:Nn \l__enumext_minipage_right_skip
929         {
930             0.695\box_dp:N \strutbox
931         }
932         \skip_set:Nn \l__enumext_minipage_after_skip
933         {
934             \box_dp:N \strutbox
935         }
936         \__enumext_zero_parsep:
937     }
```

```

938     {
939         \skip_set:Nn \l__enumext_minipage_left_skip
940         {
941             \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
942         }
943         \skip_set:Nn \l__enumext_minipage_right_skip
944         {
945             0.695\box_dp:N \strutbox
946         }
947         \skip_set:Nn \l__enumext_minipage_after_skip
948         {
949             1.85\box_dp:N \strutbox
950             + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
951         }
952     }
953 }
954 {

```

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.

```

955 \skip_if_eq:nnTF
956 { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
957 {
958     \skip_set:Nn \l__enumext_minipage_left_skip
959     {
960         0.5\box_dp:N \strutbox
961         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
962     }
963     \skip_set:Nn \l__enumext_minipage_right_skip
964     {
965         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
966     }
967     \skip_set:Nn \l__enumext_minipage_after_skip
968     {
969         1.6\box_dp:N \strutbox
970     }
971 }
972 {
973     \skip_set:Nn \l__enumext_minipage_left_skip
974     {
975         0.5875\box_dp:N \strutbox
976         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
977     }
978     \skip_set:Nn \l__enumext_minipage_right_skip
979     {
980         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
981         + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
982     }
983     \skip_set:Nn \l__enumext_minipage_after_skip
984     {
985         0.325\box_dp:N \strutbox
986         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
987     }
988 }
989 }
990 }

```

(End of definition for `__enumext_mini_set_vskip:`)

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

```

991 \cs_new_protected:Nn \__enumext_zero_parsep:
992 {
993     \int_case:nn { \l__enumext_level_int }
994     {
995         { 2 }{
996             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
997             {
998                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }

```

```

999         }
1000     }
1001     { 3 }{
1002         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1003         {
1004             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1005         }
1006     }
1007     { 4 }{
1008         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1009         {
1010             \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1011         }
1012     }
1013 }
1014 }

```

(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*.

```

1015 \cs_new_protected:Nn \__enumext_mini_addvspace:
1016 {
1017     \__enumext_mini_set_vskip:
1018     \mode_if_vertical:T
1019     {
1020         \skip_add:Nn \l__enumext_minipage_left_skip
1021         {
1022             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1023         }
1024         \skip_add:Nn \l__enumext_minipage_after_skip
1025         {
1026             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1027         }
1028     }
1029     \par\nopagebreak
1030     \addvspace { \l__enumext_minipage_left_skip }
1031 }

```

(End of definition for `__enumext_mini_addvspace:`.)

10.19.2 Adjustment of vertical spaces for minipage in keyans

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

```

1032 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1033 {
1034     \skip_zero_new:N \l__enumext_minipage_after_skip
1035     \skip_zero_new:N \l__enumext_minipage_left_skip
1036     \skip_zero_new:N \l__enumext_minipage_right_skip
1037     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1038     {
1039         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1040         {
1041             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1042             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1043             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1044             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1045             {
1046                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1047             }
1048         }
1049     }
1050     \skip_set:Nn \l__enumext_minipage_left_skip
1051     {
1052         \skip_use:N \l__enumext_topsep_v_skip
1053     }
1054     \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1055         {
1056             0.705\box_dp:N \strutbox
1057         }
1058         \skip_set:Nn \l__enumext_minipage_after_skip
1059         {
1060             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1061         }
1062     }
1063 }
1064 {
1065     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1066     {
1067         \skip_set:Nn \l__enumext_minipage_left_skip
1068         {
1069             0.5\box_dp:N \strutbox
1070             + \l__enumext_partopsep_v_skip
1071         }
1072         \skip_set:Nn \l__enumext_minipage_right_skip
1073         {
1074             \l__enumext_partopsep_v_skip
1075         }
1076         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1077     }
1078     {
1079         \skip_set:Nn \l__enumext_minipage_left_skip
1080         {
1081             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1082         }
1083         \skip_set:Nn \l__enumext_minipage_right_skip
1084         {
1085             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1086         }
1087         \skip_set:Nn \l__enumext_minipage_after_skip
1088         {
1089             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1090         }
1091     }
1092 }
1093 }

```

(End of definition for `\l__enumext_keyans_mini_set_vskip:`)

`\l__enumext_keyans_mini_addvspace:`

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

```

1094 \cs_new_protected:Nn \l__enumext_keyans_mini_addvspace:
1095 {
1096     \l__enumext_keyans_mini_set_vskip:
1097     \mode_if_vertical:T
1098     {
1099         \skip_add:Nn \l__enumext_minipage_left_skip
1100         {
1101             \l__enumext_partopsep_v_skip
1102         }
1103         \skip_add:Nn \l__enumext_minipage_after_skip
1104         {
1105             \l__enumext_partopsep_v_skip
1106         }
1107     }
1108     \par\nopagebreak
1109     \addvspace { \l__enumext_minipage_left_skip }
1110 }

```

(End of definition for `\l__enumext_keyans_mini_addvspace:`)

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

`\l__enumext_mini_set_vskip_vii:`

`\l__enumext_mini_set_vskip_viii:`

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


```

1111 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1112 {
1113   \skip_zero_new:N \l__enumext_minipage_left_skip
1114   \skip_gzero_new:N \g__enumext_minipage_right_skip
1115   \skip_gzero_new:N \g__enumext_minipage_after_skip
1116   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1117   {
1118     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1119     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1120   }
1121   {
1122     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1123     \skip_gset:Nn \g__enumext_minipage_right_skip
1124       {
1125         \l__enumext_topsep_vii_skip
1126       }
1127     \skip_gset:Nn \g__enumext_minipage_after_skip
1128       {
1129         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1130       }
1131   }
1132 }
1133 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1134 {
1135   \skip_zero_new:N \l__enumext_minipage_after_skip
1136   \skip_zero_new:N \l__enumext_minipage_left_skip
1137   \skip_zero_new:N \l__enumext_minipage_right_skip
1138   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1139   {
1140     \skip_set:Nn \l__enumext_minipage_left_skip
1141       {
1142         0.5\box_dp:N \strutbox
1143       }
1144     \skip_set:Nn \l__enumext_minipage_right_skip
1145       {
1146         \l__enumext_partopsep_viii_skip
1147       }
1148     \skip_set:Nn \l__enumext_minipage_after_skip
1149       {
1150         1.6\box_dp:N \strutbox
1151       }
1152   }
1153   {
1154     \skip_set:Nn \l__enumext_minipage_left_skip
1155       {
1156         0.5875\box_dp:N \strutbox
1157       }
1158     \skip_set:Nn \l__enumext_minipage_right_skip
1159       {
1160         \l__enumext_topsep_viii_skip
1161       }
1162     \skip_set:Nn \l__enumext_minipage_after_skip
1163       {
1164         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1165       }
1166   }
1167 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext_mini_env*` environment on the *left side* when the `miniright` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether T_EX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1168 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1169 {
1170   \__enumext_mini_set_vskip_vii:
1171   \par\nopagebreak
1172   \addvspace { \l__enumext_minipage_left_skip }
1173 }

```

```

1174 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1175 {
1176   \__enumext_mini_set_vskip_viii:
1177   \par\nopagebreak
1178   \addvspace { \__enumext_minipage_left_skip }
1179 }

```

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

10.19.4 The command \miniright

The command `\miniright` will close the `__enumext_mini_env*` environment on the “left side”, open the `__enumext_mini_env*` environment on the “right side” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “right side” environment. The *starred version* ‘*’ inhibits the use of `\centering` command i.e. the usual L^AT_EX 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.

```

1180 \NewDocumentCommand \miniright { s }
1181 {
1182   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1183   {
1184     \msg_error:nnn { enumext } { wrong-miniright-place }
1185   }
1186   \int_compare:nNt { \__enumext_level_int } = { 0 }
1187   {
1188     \msg_error:nnn { enumext } { wrong-miniright-place }
1189   }
1190   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1191   {
1192     \__enumext_keyans_mini_right_cmd:n {#1}
1193   }
1194   { \__enumext_mini_right_cmd:n {#1} }
1195 }

```

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

`__enumext_mini_right_cmd:n`

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

```

1196 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1197 {
1198   \dim_compare:nNtF
1199   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1200   {
1201     \__enumext_multicols_stop:
1202     \end{__enumext_mini_env*}
1203     \hfill
1204     \begin{__enumext_mini_env*}
1205     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1206     \par\addvspace { \__enumext_minipage_right_skip }
1207     \bool_if:nF {#1}
1208     {
1209       \centering
1210     }
1211     \int_gzero:N \g__enumext_minipage_stat_int
1212   }
1213   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1214 }

```

(End of definition for __enumext_mini_right_cmd:n.)

_enumext_keyans_mini_right_cmd:n

The function _enumext_keyans_mini_right_cmd:n takes as argument the *starred version* ‘*’ of the `\\mini\\right` command in the `keyans` environment. The implementation of this function is the same as that of the _enumext_mini_right_cmd:n function of the `enumext` environment.

```

1215 \\cs_new_protected:Npn \\_enumext_keyans_mini_right_cmd:n #1
1216 {
1217   \\dim_compare:nNnTF { \\_enumext_minipage_right_v_dim } > { \\c_zero_dim }
1218   {
1219     \\_enumext_keyans_multicols_stop:
1220     \\end{\\_enumext_mini_env*}
1221     \\hfill
1222     \\begin{\\_enumext_mini_env*}{ \\_enumext_minipage_right_v_dim }
1223     \\par\\addvspace { \\_enumext_minipage_right_skip }
1224     \\bool_if:nF {#1}
1225     {
1226       \\centering
1227     }
1228     \\int_gzero:N \\g__enumext_minipage_stat_int
1229   }
1230   { \\msg_error:nnn { enumext } { wrong-mini\\right-use } }
1231 }

```

(End of definition for _enumext_keyans_mini_right_cmd:n.)

10.20 Setting above and below keys

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

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

```

1232 \\cs_set_protected:Npn \\_enumext_tmp:nn #1 #2
1233 {
1234   \\keys_define:nn { enumext / #1 }
1235   {
1236     above .skip_set:c = { \\_enumext_vspace_above_#2_skip },
1237     above .value_required:n = true,
1238     above* .code:n = \\bool_set_true:c { \\_enumext_vspace_a_star_#2_bool }
1239     \\keys_set:nn { enumext / #1 } { above = {##1} },
1240     above* .value_required:n = true,
1241     below .skip_set:c = { \\_enumext_vspace_below_#2_skip },
1242     below .value_required:n = true,
1243     below* .code:n = \\bool_set_true:c { \\_enumext_vspace_b_star_#2_bool }
1244     \\keys_set:nn { enumext / #1 } { below = {##1} },
1245     below* .value_required:n = true,
1246   }
1247 }
1248 \\clist_map_inline:Nn \\c__enumext_all_envs_clist { \\_enumext_tmp:nn #1 }

```

(End of definition for above and others.)

10.20.1 Functions for above and below keys in enumext

_enumext_vspace_above:

The function _enumext_vspace_above: apply the *vertical space above* the `enumext` environment set by the `above*` and `above` keys.

```

1249 \\cs_new_protected:Nn \\_enumext_vspace_above:
1250 {
1251   \\skip_if_eq:nnF
1252   { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } } { \\c_zero_skip }
1253   {
1254     \\bool_if:cTF { \\_enumext_vspace_a_star_ \\_enumext_level: _bool }
1255     {
1256       \\vspace*{ \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1257     }
1258     {
1259       \\vspace { \\skip_use:c { \\_enumext_vspace_above_ \\_enumext_level: _skip } }
1260     }
1261   }
1262 }

```

(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.

```

1263 \cs_new_protected:Nn \__enumext_vspace_below:
1264 {
1265   \skip_if_eq:nnF
1266   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1267   {
1268     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1269     {
1270       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1271     }
1272     {
1273       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1274     }
1275   }
1276 }

```

(End of definition for `__enumext_vspace_below`.)

10.20.2 Functions for above and below keys in keyans

`__enumext_vspace_above_v`: The function `__enumext_vspace_above_v`: apply the *vertical space above* the `keyans` environment set by the `above` and `above*` keys.

```

1277 \cs_new_protected:Nn \__enumext_vspace_above_v:
1278 {
1279   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1280   {
1281     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1282     {
1283       \vspace*{ \l__enumext_vspace_above_v_skip }
1284     }
1285     { \vspace { \l__enumext_vspace_above_v_skip } }
1286   }
1287 }

```

(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.

```

1288 \cs_new_protected:Nn \__enumext_vspace_below_v:
1289 {
1290   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1291   {
1292     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1293     {
1294       \vspace*{ \l__enumext_vspace_below_v_skip }
1295     }
1296     { \vspace { \l__enumext_vspace_below_v_skip } }
1297   }
1298 }

```

(End of definition for `__enumext_vspace_below_v`.)

10.20.3 Functions for above and below keys in enumext* keyans*

`__enumext_vspace_above_vii`: The functions `__enumext_vspace_above_vii`: and `__enumext_vspace_above_viii`: apply the *vertical space above* the `enumext*` and `keyans*` environments set by the `above` and `above*` keys.

`__enumext_vspace_above_viii`:

```

1299 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1300 {
1301   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1302   {
1303     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1304     {
1305       \vspace*{ \l__enumext_vspace_above_vii_skip }
1306     }
1307     { \vspace { \l__enumext_vspace_above_vii_skip } }
1308   }
1309 }
1310 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1311 {
1312   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1313   {

```

```

1314         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1315         {
1316             \vspace*{ \l__enumext_vspace_above_viii_skip }
1317         }
1318         { \vspace { \l__enumext_vspace_above_viii_skip } }
1319     }
1320 }

```

(End of definition for \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii:.)

```

\__enumext_vspace_below_vii:
\__enumext_vspace_below_viii:

```

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

```

1321 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1322 {
1323     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1324     {
1325         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1326         {
1327             \vspace*{ \l__enumext_vspace_below_vii_skip }
1328         }
1329         { \vspace { \l__enumext_vspace_below_vii_skip } }
1330     }
1331 }
1332 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1333 {
1334     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1335     {
1336         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1337         {
1338             \vspace*{ \l__enumext_vspace_below_viii_skip }
1339         }
1340         { \vspace { \l__enumext_vspace_below_viii_skip } }
1341     }
1342 }

```

(End of definition for \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii:.)

10.21 Setting save-ans and resume keys

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

```

save-ans
resume
resume*
series

```

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

```

1343 \keys_define:nn { enumext / level-1 }
1344 {
1345     save-ans .code:n = \__enumext_storing_set:n {#1},
1346     save-ans .value_required:n = true,
1347     resume* .code:n = \__enumext_resume_starred:,
1348     resume* .value_forbidden:n = true,
1349     resume .code:n = \__enumext_resume_counter_series:n {#1},
1350     series .str_set:N = \l__enumext_series_str,
1351     series .value_required:n = true,
1352 }
1353 \keys_define:nn { enumext / enumext* }
1354 {
1355     save-ans .code:n = \__enumext_storing_set:n {#1},
1356     save-ans .value_required:n = true,
1357     resume* .code:n = \__enumext_resume_starred_vii:,
1358     resume* .value_forbidden:n = true,
1359     resume .code:n = \__enumext_resume_counter_series_vii:n {#1},
1360     series .str_set:N = \l__enumext_series_str,
1361     series .value_required:n = true,
1362 }

```

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

```

\__enumext_resume_counter:n
\__enumext_resume_only_counter:
\__enumext_resume_counter_vii:

```

The functions \l__enumext_resume_only_counter: and \l__enumext_resume_only_counter_vii: used by `resume` key in `enumext` and `enumext*`. If `save-ans` key present then set the start value from integer created by \l__enumext_storing_set:n.

```

1363 \cs_new_protected:Nn \__enumext_resume_counter:

```

```

1364 {
1365     \bool_if:NT \l__enumext_store_active_bool
1366     {
1367         \int_gset:Nn \g__enumext_resume_int
1368         {
1369             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1370         }
1371     }
1372     \int_gincr:N \g__enumext_resume_int
1373     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1374 }
1375 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1376 {
1377     \bool_if:NT \l__enumext_store_active_bool
1378     {
1379         \int_gset:Nn \g__enumext_resume_vii_int
1380         {
1381             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1382         }
1383     }
1384     \int_gincr:N \g__enumext_resume_vii_int
1385     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1386 }

```

(End of definition for __enumext_resume_counter:n, __enumext_resume_only_counter:, and __enumext_resume_counter_vii:.)

```

__enumext_resume_counter_series:n
\__enumext_filter_series:n
    __enumext_filter_series_key:n
    __enumext_filter_series_pair:nn
    __enumext_parse_series_resume:n
1387 \cs_new_protected:Nn \__enumext_resume_starred:
1388 {
1389     \tl_if_empty:NF \g__enumext_series_standar_default_tl
1390     {
1391         \__enumext_resume_counter:
1392         \keys_set:nV { enumext / level-1 } \g__enumext_series_standar_default_tl
1393     }
1394 }
1395 \cs_new_protected:Npn \__enumext_resume_counter_series:n #1
1396 {
1397     \tl_if_empty:nTF {#1}
1398     {
1399         \__enumext_resume_counter:
1400     }
1401     {
1402         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1403         {
1404             \keys_set:nv { enumext / level-1 }
1405             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1406         }
1407         { \msg_error:nnn { enumext } { unknown-series } {#1} }
1408     }
1409 }
1410 \cs_new_protected:Nn \__enumext_resume_starred_vii:
1411 {
1412     \tl_if_empty:NF \g__enumext_series_starred_default_tl
1413     {
1414         \__enumext_resume_counter_vii:
1415         \keys_set:nV { enumext / enumext* } \g__enumext_series_starred_default_tl
1416     }
1417 }
1418 \cs_new_protected:Npn \__enumext_resume_counter_series_vii:n #1
1419 {
1420     \tl_if_empty:nTF {#1}
1421     {
1422         \__enumext_resume_counter_vii:
1423     }
1424     {
1425         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1426         {
1427             \keys_set:nv { enumext / enumext* }
1428             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1429         }

```

```

1430         { \msg_error:nnn { enumext } { unknown-series } {#1} }
1431     }
1432 }
1433
1434 \cs_new:Npn \__enumext_filter_series:n #1
1435 {
1436     \use:e
1437     {
1438         \keyval_parse:NNn
1439         \__enumext_filter_series_key:n
1440         \__enumext_filter_series_pair:nn {#1}
1441     }
1442 }
1443 \cs_new:Npn \__enumext_filter_series_key:n #1
1444 {
1445     \str_case:nnF {#1}
1446     {
1447         { resume } {}
1448         { resume* } {}
1449     }
1450     { , { \exp_not:n {#1} } }
1451 }
1452 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1453 {
1454     \str_case:nnF {#1}
1455     {
1456         { series } {}
1457         { resume } {}
1458         { columns* } {}
1459         { columns-sep* } {}
1460     }
1461     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1462 }
1463 \cs_new_protected:Npn \__enumext_parse_series_resume:n #1
1464 {
1465     \str_if_empty:NTF \l__enumext_series_str
1466     {
1467         \__enumext_resume_series_default:n {#1}
1468     }
1469     {
1470         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1471         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1472         { \__enumext_filter_series:n {#1} }
1473     }
1474 }
1475 \cs_new_protected:Npn \__enumext_resume_series_default:n #1
1476 {
1477     % Only if enumext not nested in enumext*
1478     \bool_lazy_all:nT
1479     {
1480         { \bool_if_p:N \g__enumext_standar_bool }
1481         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1482         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1483     }
1484     {
1485         \typeout{[[ON-LEVEL-ONE-ENUMEXT]]}
1486         \tl_gclear:N \g__enumext_series_standar_default_tl
1487         \tl_gset:Ne \g__enumext_series_standar_default_tl { \__enumext_filter_series:n {#1} }
1488     }
1489     % Only if enumext* not nested in enumext.
1490     \bool_lazy_all:nT
1491     {
1492         { \bool_if_p:N \g__enumext_starred_bool }
1493         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1494         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1495     }
1496     {
1497         \typeout{[[ON-LEVEL-ONE-ENUMEXT*]]}
1498         \tl_gclear:N \g__enumext_series_starred_default_tl
1499         \tl_gset:Ne \g__enumext_series_starred_default_tl { \__enumext_filter_series:n {#1} }
1500     }

```



```
1501 }
```

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

```
\__enumext_storing_set:n
```

The function `__enumext_storing_set:n` executed by the `save-ans` key sets the parameters for the operation of `\anskey`, `keyans` and `keyanspic`. The variable `\l__enumext_store_name_tl` will have the “store name” with which the *(sequence)* and *(prop list)* will be created, if it does not exist it will create it globally.

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

```
1502 \cs_new_protected:Npn \__enumext_storing_set:n #1
1503 {
1504   \tl_set:Nx \l__enumext_store_name_tl {#1}
1505   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1506   {
1507     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1508   }
1509   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1510   {
1511     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1512   }
1513   \bool_set_true:N \l__enumext_store_active_bool
1514   \bool_set_true:N \l__enumext_store_ans_bool
1515   \int_if_exist:cF { g__enumext_resume_#1_int }
1516   {
1517     \int_new:c { g__enumext_resume_#1_int }
1518   }
1519 }
```

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

10.22 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_count_item_number_int` must match the integer variable `g__enumext_count_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.

10.22.1 Setting check-ans key

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

```
no-store 1520 \cs_set_protected:Npn \__enumext_tmp:n #1
1521 {
1522   \keys_define:nn { enumext / #1 }
1523   {
1524     check-ans .bool_set:N = \__enumext_check_ans_bool,
1525     check-ans .initial:n = false,
1526     no-store .code:n = {
1527       \bool_set_false:N \__enumext_store_ans_bool
1528       \bool_set_false:N \__enumext_check_ans_bool
1529     },
1530     no-store .value_forbidden:n = true,
1531   }
1532 }
1533 \clist_map_inline:nn
1534 {
1535   level-1, level-2, level-3, level-4, enumext*
1536 }
1537 { \__enumext_tmp:n {#1} }
```

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

10.22.2 Set-up check answer mechanism

`__enumext_check_ans_set:` The function `__enumext_check_ans_set:` will adjust the value of the variable `\g__enumext_count_item_number_int` by decrementing its value by one each time you open a nested level `enumext` environment.

```
1538 \cs_new_protected:Nn \__enumext_check_ans_set:
1539 {
1540   \int_case:nn { \__enumext_level_int }
1541   {
1542     { 1 }{
1543       \bool_lazy_all:nT
1544       {
1545         { \bool_if_p:N \g__enumext_starred_bool }
1546         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
1547       }
1548       {
1549         \int_gdecr:N \g__enumext_count_item_number_int
1550         \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEEESTED}
1551       }
1552     }
1553     { 2 }{
1554       \int_gdecr:N \g__enumext_count_item_number_int
1555     }
1556     { 3 }{
1557       \int_gdecr:N \g__enumext_count_item_number_int
1558     }
1559     { 4 }{
1560       \int_gdecr:N \g__enumext_count_item_number_int
1561     }
1562   }
1563   \int_case:nn { \__enumext_level_h_int }
1564   {
1565     { 1 }{
1566       \bool_if:NT \g__enumext_standar_bool
1567       {
1568         \int_gdecr:N \g__enumext_count_item_number_int
1569         \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEEESTED}
1570       }
1571     }
1572   }
1573 }
```

(End of definition for `__enumext_check_ans_set:`)

`__enumext_check_ans_exec:` The function `__enumext_check_ans_exec:` will count the number of times the `\item` and `\item*` commands appears per level within the `enumext` environment. The boolean variable `\l__enumext_store_ans_bool` controlled by the `no-store` key will increment the integer variable of the level counter by `1` to preserve the equality that we will use in the final comparison of the process.

```

1574 \cs_new_protected:Nn \__enumext_check_ans_exec:
1575 {
1576   \bool_if:NT \l__enumext_check_ans_bool
1577   {
1578     \__enumext_check_ans_set:
1579   }
1580 }

```

(End of definition for `__enumext_check_ans_exec:`.)

`__enumext_check_ans_show:` The function `__enumext_check_ans_show:` compares all executions of `\item` and `\item*` with the executions of `\anskey`. After the function is executed, we set the integer variables to zero.

```

1581 \cs_new_protected:Nn \__enumext_check_ans_show:
1582 {
1583   \int_compare:nNnTF
1584     { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1585     {
1586       \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1587     }
1588     {
1589       \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1590     }
1591   \int_gzero:N \g__enumext_count_item_number_int
1592   \int_gzero:N \g__enumext_count_item_anskey_int
1593 }

```

(End of definition for `__enumext_check_ans_show:`.)

10.23 Keys and functions associated with storage

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

```

1594 \cs_set_protected:Npn \__enumext_tmp:n #1
1595 {
1596   \keys_define:nn { enumext / #1 }
1597   {
1598     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1599     wrap-ans .initial:n = \fbox{##1},
1600     wrap-ans .value_required:n = true,
1601     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1602     wrap-opt .initial:n = [{##1}],
1603     wrap-opt .value_required:n = true,
1604     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
1605     save-sep .initial:n = {, ~ },
1606     save-sep .value_required:n = true,
1607     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
1608     mark-ans .initial:n = \textasteriskcentered,
1609     mark-ans .value_required:n = true,
1610     mark-pos .choice:,
1611     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1612     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1613     mark-pos .initial:n = right,
1614     mark-pos .value_required:n = true,
1615     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1616     show-ans .initial:n = false,
1617     show-ans .value_required:n = true,
1618     show-pos .bool_set:N = \l__enumext_show_position_bool,
1619     show-pos .initial:n = false,
1620     show-pos .value_required:n = true,
1621     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
1622     mark-ref .initial:n = \textasteriskcentered,
1623     mark-ref .value_required:n = true,
1624     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1625     save-ref .initial:n = false,
1626     save-ref .value_required:n = true,
1627   }
1628 }
1629 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

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

```

1630 \cs_set_protected:Npn \__enumext_tmp:n #1
1631 {
1632   \keys_define:nn { enumext / #1 }
1633   {
1634     mark-pos .choice:,
1635     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1636     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1637     mark-pos .initial:n = right,
1638     mark-pos .value_required:n = true,
1639     show-ans .bool_set:N = \l__enumext_show_answer_bool,
1640     show-ans .initial:n = false,
1641     show-ans .value_required:n = true,
1642     show-pos .bool_set:N = \l__enumext_show_position_bool,
1643     show-pos .initial:n = false,
1644     show-pos .value_required:n = true,
1645   }
1646 }
1647 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

columns* For the `enumext` and `enumext*` environments we will only add the keys `columns*` and `columns-sep*`.
columns-sep* The values set by these keys will be passed as optional arguments to the “inner levels” of the `enumext` and `enumext*` environments via the `__enumext_store_level_open:` function used by the “storage system” to preserve the structure and then used by the `\printkeyans` command.

```

1648 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1649 {
1650   \keys_define:nn { enumext / #1 }
1651   {
1652     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1653     \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1654     \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1655     {
1656       columns = \exp_not:v { l__enumext_store_columns_#2_int },
1657     },
1658     columns* .value_required:n = true,
1659     columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1660     \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1661     \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1662     {
1663       columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1664     },
1665     columns-sep* .value_required:n = true,
1666   }
1667 }
1668 \clist_map_inline:nn
1669 {
1670   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1671 }
1672 { \__enumext_tmp:nn #1 }

```

(End of definition for `columns*` and `columns-sep*`.)

10.23.1 Function for storing content in prop list

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

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

```

1673 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1674 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1675 {
1676   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1677   {
1678     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1679   }
1680   { #1 }
1681 }

```

```
1682 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

10.23.2 Function for storing content in sequence

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 an 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.

```
1683 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1684 {
1685   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
1686 }
1687 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

10.23.3 Functions for storing the list structure in the sequence

The memorization structure of the list is handled by the functions `__enumext_store_level_open:` and `__enumext_store_level_close:` which are executed per level within the `enumext` environment. As this structure will be stored in the sequence set by the `save-ans` key, we will not be able to modify it locally, so it is better to take only two copies of the values set by the `columns` and `columns-sep` keys if they are present when changing levels within the `enumext` environment when executing `\anskey`. We will store these values in the variable `__enumext_store_columns_X_tl` if they are different from `0` and `0pt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```
1688 \cs_new_protected:Nn \__enumext_store_level_open:
1689 {
1690   \bool_if:NT \__enumext_store_ans_bool
1691   {
1692     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1693     {
1694       \__enumext_store_addto_seq:n
1695       {
1696         \item \begin{enumext}
1697       }
1698     }
1699     {
1700       \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1701       {
1702         \item \begin{enumext} [
1703       }
1704       \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1705       {
1706         ]
1707       }
1708       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1709     }
1710   }
1711 }
1712 \cs_new_protected:Nn \__enumext_store_level_close:
1713 {
1714   \bool_if:NT \__enumext_store_ans_bool
1715   {
1716     \__enumext_store_addto_seq:n { \end{enumext} }
1717   }
1718 }
```

(End of definition for `__enumext_store_level_open:` and `__enumext_store_level_close:`.)

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.

```
1719 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1720 {
1721   \bool_if:NT \__enumext_store_ans_bool
1722   {
1723     \tl_if_empty:NTF \__enumext_store_opt_vii_tl
```

```

1724         {
1725             \__enumext_store_addto_seq:n
1726             {
1727                 \item \mode_leave_vertical:
1728                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1729                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1730             }
1731         }
1732     {
1733         \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1734         {
1735             \item \mode_leave_vertical:
1736             \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1737             \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1738         }
1739         \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1740         {
1741             ]
1742         }
1743         \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1744     }
1745 }
1746 }
1747 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1748 {
1749     \bool_if:NT \l__enumext_store_ans_bool
1750     {
1751         \__enumext_store_addto_seq:n { \end{enumext*} }
1752     }
1753 }

```

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

10.23.4 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

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

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```

1754 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1755 {
1756     \mode_leave_vertical:
1757     \skip_horizontal:n { -\dim_use:N #2 }
1758     \makebox[\opt][ r ]
1759     {
1760         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1761         {
1762             \tl_use:N \l__enumext_mark_answer_sym_tl
1763         }
1764     }
1765     \skip_horizontal:n { \dim_use:N #2 }
1766 }
1767 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

10.24 The command \anskey and internal label and ref

Since we will be “storing content” in a list environment within *sequences* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over \item when storing. The \anskey command will cover this point and give it very similar behaviour to that of \item in the enumext and enumext* environments.

\anskey We want the command to be executed as follows: \anskey(<number>)*[<key = val>]{<content>} so first we’ll add the keys item-sym*, item-pos* and store-brk.

```

1768 \keys_define:nn { enumext / anskey }
1769 {
1770     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1771     item-sym* .value_required:n = true,
1772     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,

```

```

1773     item-pos* .value_required:n = true,
1774     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1775     store-brk .default:n = true,
1776     store-brk .value_forbidden:n = true,
1777 }

```

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

```

1778 \NewDocumentCommand \anskey { d() s o +m }
1779 {
1780   \bool_if:NF \l__enumext_store_active_bool
1781   {
1782     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1783   }
1784   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1785   {
1786     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1787   }
1788   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1789   {
1790     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1791   }
1792   \group_begin:
1793     \bool_if:NT \l__enumext_store_ans_bool
1794     {
1795       \bool_if:NT \l__enumext_check_ans_bool
1796       {
1797         \int_gincr:N \g__enumext_count_item_anskey_int
1798       }
1799       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1800     }
1801   \group_end:
1802 }

```

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

`__enumext_store_anskey_code:nnnn`

The internal function `__enumext_store_anskey_code:nnnn` first we pass the command *argument* to the *prop list*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `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.

```

1803 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1804 {
1805   \__enumext_store_addto_prop:n {#4}
1806   \bool_if:NT \l__enumext_store_ref_key_bool
1807   {
1808     \__enumext_store_internal_ref:
1809   }
1810   \__enumext_store_anskey_show_left:n { #4 }

```

Now we start processing the optional arguments passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “store” in the *sequence*. First we clear the variable `\l__enumext_store_anskey_arg_tl` and process `[key = val]`, if the `store-brk` key is present and the command is running under `enumext` (not in the starred version) we will add `\columnbreak` and then `\item`.

```

1811   \tl_clear:N \l__enumext_store_anskey_arg_tl
1812   \tl_if_novalue:nF {#3}
1813   {
1814     \keys_set:nn { enumext / anskey } {#3}
1815   }
1816   \bool_lazy_and:nnT
1817   { \bool_if_p:N \l__enumext_store_columns_break_bool }
1818   { \bool_not_p:n { \l__enumext_starred_bool } }
1819   {
1820     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1821   }
1822   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```


Now we will check the ($\langle number \rangle$) argument and add it to `\l__enumext_store_anskey_arg_tl` if the command is running under `enumext*` (starred version).

```

1823 \tl_if_novalue:nF {#1}
1824 {
1825   \int_set:Nn \l__enumext_store_columns_join_int {#1}
1826   \bool_if:NT \l__enumext_starred_bool
1827   {
1828     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1829     {
1830       ( \exp_not:V \l__enumext_store_columns_join_int )
1831     }
1832   }
1833 }

```

And now we will review the starred argument `*` together with the keys `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_arg_tl`.

```

1834 \bool_if:nTF {#2}
1835 {
1836   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1837   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1838   {
1839     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1840     {
1841       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1842     }
1843   }
1844   \dim_compare:nT
1845   {
1846     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1847   }
1848   {
1849     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1850     {
1851       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1852     }
1853   }
1854   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1855 }
1856 {
1857   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1858 }

```

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

```

1859 \bool_lazy_and:nnT
1860 { \bool_if_p:N \l__enumext_store_ref_key_bool }
1861 { \bool_if_p:N \l__enumext_hyperref_bool }
1862 {
1863   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1864   {
1865     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1866     { \exp_not:V \l__enumext_mark_ref_sym_tl }
1867   }
1868 }
1869 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1870 }

```

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

`\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 ($\langle labels \rangle$), 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`.

```

1871 \cs_new_protected:Nn \l__enumext_store_internal_ref:
1872 {
1873   \cs_set_protected:Npn \l__enumext_tmp:n ##1
1874   {
1875     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1876     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1877     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }

```

```

1878     \tl_reverse:c { l__enumext_label_copy_##1_tl }
1879   }
1880   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1881   \cs_set:Npn \__enumext_tmp:n ##1
1882     { . \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.

```

1883   \bool_lazy_all:nT
1884   {
1885     { \bool_if_p:N \g__enumext_starred_bool }
1886     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1887   }
1888   {
1889     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1890     { \tl_use:N \l__enumext_label_copy_vii_tl }
1891   }
1892   \bool_lazy_all:nT
1893   {
1894     { \bool_if_p:N \l__enumext_standar_bool }
1895     { \bool_if_p:N \g__enumext_starred_bool }
1896     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1897   }
1898   {
1899     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1900     {
1901       \tl_use:N \l__enumext_label_copy_vii_tl
1902       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1903     }
1904   }

```

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.

```

1905   \bool_lazy_all:nT
1906   {
1907     { \bool_if_p:N \l__enumext_standar_bool }
1908     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1909     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1910     { \bool_not_p:n { \l__enumext_starred_bool } }
1911   }
1912   {
1913     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1914     {
1915       \tl_use:N \l__enumext_label_copy_i_tl
1916       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1917     }
1918   }
1919   \cs_set:Npn \__enumext_tmp:n ##1
1920     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1921   \bool_lazy_all:nT
1922   {
1923     { \bool_if_p:N \l__enumext_standar_bool }
1924     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1925     { \bool_not_p:n { \g__enumext_starred_bool } }
1926     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1927   }
1928   {
1929     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1930     {
1931       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1932       . \tl_use:N \l__enumext_label_copy_vii_tl
1933     }
1934   }

```

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

```

1935   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1936   {
1937     \l__enumext_store_name_tl \c_colon_str
1938     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1939   }

```

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

```

1940 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1941 {
1942   \__enumext_newlabel:n
1943   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1944   { \l__enumext_newlabel_arg_two_tl }
1945 }
1946 \l__enumext_store_write_aux_file_tl
1947 }

```

(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.

```

1948 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1949 {
1950   \par
1951   \bool_if:NT \l__enumext_starred_bool
1952   {
1953     \cs_set:Nn \__enumext_level: { vii }
1954   }
1955   \__enumext_print_keyans_box:cc
1956   { \l__enumext_labelwidth_ \__enumext_level: _dim }
1957   { \l__enumext_labelsep_ \__enumext_level: _dim }
1958   \__enumext_anskey_wrapper:n { #1 }
1959 }

```

(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.

```

1960 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1961 {
1962   \bool_if:NT \l__enumext_show_answer_bool
1963   {
1964     \__enumext_store_anskey_show_wrap:n { #1 }
1965   }
1966   \bool_if:NT \l__enumext_show_position_bool
1967   {
1968     \tl_set:Ne \l__enumext_mark_answer_sym_tl
1969     {
1970       \group_begin:
1971       \exp_not:N \normalfont
1972       \exp_not:N \footnotesize [ \int_eval:n
1973       {
1974         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1975       }
1976       ]
1977       \group_end:
1978     }
1979     \__enumext_store_anskey_show_wrap:n { #1 }
1980   }
1981 }

```

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

10.25 Common functions for keyans, keyans* and keyanspic

10.25.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

1982 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1983 {

```

```

1984 \tl_clear:N \l__enumext_store_keyans_label_tl
1985 \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1986 {
1987   \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1988 }
1989 {
1990   \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1991 }
1992 \tl_if_novalue:nF { #1 }
1993 {
1994   % Set save-sep
1995   \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
1996   {
1997     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
1998   }
1999   \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2000 }
2001 \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2002 }

```

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

10.25.2 The save-ref key for keyans, keyans* and keyanspic

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

```

2003 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2004 {
2005   \bool_if:NT \l__enumext_store_ref_key_bool
2006   {
2007     \cs_set_protected:Npn \__enumext_tmp:n ##1
2008     {
2009       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2010       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2011       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2012       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2013     }
2014     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2015     \__enumext_keyans_store_ref_aux_i:
2016   }
2017 }

```

The auxiliary function `__enumext_keyans_store_ref_aux_i:` set the variable `\l__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`.

```

2018 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2019 {
2020   \bool_if:NT \g__enumext_starred_bool
2021   {
2022     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2023   }
2024   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2025   {
2026     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2027     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2028   }
2029   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2030   {
2031     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2032     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2033   }
2034   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2035   {
2036     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2037     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2038   }

```

```

2039 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2040 {
2041     \l__enumext_store_name_tl \c_colon_str
2042     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2043 }
2044 \__enumext_keyans_store_ref_aux_ii:
2045 }

```

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.

```

2046 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2047 {
2048     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2049     {
2050         \__enumext_newlabel:nn
2051         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2052         { \l__enumext_newlabel_arg_two_tl }
2053     }
2054     \l__enumext_store_write_aux_file_tl
2055 }

```

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

10.25.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

The function `__enumext_keyans_addto_seq:n` will pass the contents of the current *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.

```

2056 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2057 {
2058     \tl_clear:N \l__enumext_store_keyans_label_tl
2059     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2060     {
2061         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2062     }
2063     {
2064         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2065     }
2066     \tl_if_novalue:nF { #1 }
2067     {
2068         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2069         {
2070             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2071         }
2072         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2073     }
2074     \__enumext_keyans_addto_seq_link:
2075 }

```

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_keyans_check_ans:nn` and increment the value of the integer variable `\g__enumext_count_item_anskey_int` handled by the `check-ans` key.

```

2076 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2077 {
2078     \bool_lazy_and:nnT
2079     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2080     { \bool_if_p:N \l__enumext_hyperref_bool }
2081     {
2082         \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2083         {
2084             \hfill \exp_not:N \hyperlink
2085             {
2086                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2087             }
2088             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2089         }
2090     }

```

```

2090     }
2091     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2092     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2093     \bool_if:NT \l__enumext_check_ans_bool
2094     {
2095         \int_gincr:N \g__enumext_count_item_anskey_int
2096     }
2097 }

```

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

10.25.4 Check for starred commands

__enumext_keyans_check_ans:nn

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

```

2098 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2099 {
2100     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2101     {
2102         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2103     }
2104     { \tl_gclear:N \g__enumext_check_ans_item_tl }
2105 }

```

(End of definition for __enumext_keyans_check_ans:nn.)

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

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

__enumext_keyans_show_left:n
 __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_ \l__enumext_store_name_tl _prop** for **show-pos** key.

```

2106 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2107 {
2108     \tl_if_novalue:nF { #1 }
2109     {
2110         \tl_set:Nx \l__enumext_keyans_item_opt_tl { #1 }
2111     }
2112     \bool_if:NT \l__enumext_show_answer_bool
2113     {
2114         \__enumext_keyans_show_ans:
2115     }
2116     \bool_if:NT \l__enumext_show_position_bool
2117     {
2118         \__enumext_keyans_show_pos:
2119     }
2120 }
2121 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2122 {
2123     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2124     {
2125         \bool_lazy_or:nnT
2126         { \bool_if_p:N \l__enumext_show_answer_bool }
2127         { \bool_if_p:N \l__enumext_show_position_bool }
2128         {
2129             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2130         }
2131     }
2132 }
2133 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2134 {
2135     \tl_put_left:Nn \l__enumext_label_v_tl
2136     {
2137         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2138     }
2139 }
2140 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2141 {
2142     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2143     {

```

```

2144     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2145     {
2146         \group_begin:
2147         \exp_not:N \normalfont
2148         \exp_not:N \footnotesize [ \int_eval:n
2149         {
2150             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2151         }
2152         ]
2153         \group_end:
2154     }
2155 }
2156 {
2157     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2158     {
2159         \group_begin:
2160         \exp_not:N \normalfont
2161         \exp_not:N \footnotesize [ \int_eval:n
2162         {
2163             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2164         }
2165         ]
2166         \group_end:
2167     }
2168 }
2169 \tl_put_left:Nn \l__enumext_label_v_tl
2170 {
2171     \__enumext_print_keyans_box:NN
2172     \l__enumext_labelwidth_i_dim
2173     \l__enumext_labelsep_i_dim
2174 }
2175 }

```

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

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

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

```

item-sym* Define and set item-sym* and item-pos* keys for enumext and enumext*.
item-pos*
2176 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2177 {
2178     \keys_define:nn { enumext / #1 }
2179     {
2180         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2181         item-sym* .value_required:n = true,
2182         item-sym* .initial:n = { $\star$ },
2183         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2184         item-pos* .value_required:n = true,
2185     }
2186 }
2187 \clist_map_inline:nn
2188 {
2189     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2190 }
2191 { \__enumext_tmp:nn #1 }

```

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

10.27 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](#).

```

2192 \cs_new_protected:Nn \__enumext_footnotetext:nn
2193 {
2194     \footnotetext[#1]{#2}
2195 }
2196 \cs_new_protected:Nn \__enumext_renew_footnote:
2197 {
2198     \seq_gclear:N \g__enumext_footnote_arg_seq

```



```

2199 \seq_gclear:N \g__enumext_footnote_int_seq
2200 \RenewDocumentCommand \footnote { o +m }
2201 {
2202   \tl_if_novalue:nTF {##1}
2203   {
2204     \stepcounter{footnote}
2205     \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2206   }
2207   {
2208     \int_gset:Nn \g__enumext_footnote_int { ##1 }
2209   }
2210   \footnotemark [ \g__enumext_footnote_int ]
2211   \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2212   \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2213 }
2214 }
2215 \cs_new_protected:Nn \__enumext_print_footnote:
2216 {
2217   \seq_if_empty:NF \g__enumext_footnote_int_seq
2218   {
2219     \seq_map_pairwise_function:NNN
2220     \g__enumext_footnote_int_seq
2221     \g__enumext_footnote_arg_seq
2222     \__enumext_footnotetext:nn
2223   }
2224 }

```

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

10.28 Redefining `\item` command

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makelabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

10.28.1 The `\item` command in `enumext`

`__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_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `__enumext_item_std:w`.

Otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the optional argument.

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `__enumext_make_label:` (§10.29).

```

2225 \cs_new_protected:Npn \__enumext_default_item:n #1
2226 {
2227   \tl_if_novalue:nTF {##1}
2228   {
2229     \bool_if:NT \l__enumext_check_ans_bool
2230     {
2231       \int_gincr:N \g__enumext_count_item_number_int
2232     }
2233     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2234     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2235   }
2236   {
2237     \bool_set_eq:cc
2238     { l__enumext_wrap_label_ \__enumext_level: _bool }
2239     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2240     \__enumext_item_std:w [##1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2241   }
2242 }

```

(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_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:` (§10.29).

```

2243 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2244 {
2245   \tl_if_novalue:nF {#1}
2246   {
2247     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2248   }
2249   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2250   \tl_if_novalue:nTF {#2}
2251   {
2252     \dim_set_eq:cc
2253     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2254     { \__enumext_labelsep_ \__enumext_level: _dim }
2255   }
2256   {
2257     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2258   }
2259   \bool_if:NT \l__enumext_check_ans_bool
2260   {
2261     \int_gincr:N \g__enumext_count_item_number_int
2262   }
2263   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2264   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2265 }

```

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

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

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

```

2266 \cs_new_protected:Nn \__enumext_redefine_item:
2267 {
2268   \RenewDocumentCommand \item { s o o }
2269   {
2270     \bool_if:nTF {##1}
2271     {
2272       \__enumext_starred_item:nn {##2} {##3}
2273     }
2274     { \__enumext_default_item:n {##2} }
2275   }
2276 }

```

(End of definition for `__enumext_redefine_item:`.)

10.28.2 The `\item` command in keyans

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

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

```

2277 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2278 {
2279   \tl_if_novalue:nTF { #1 }
2280   {
2281     \bool_set_true:N \__enumext_wrap_label_v_bool
2282     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```

```

2283     }
2284     {
2285         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2286         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2287     }
2288 }

```

(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>`.

```

2289 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2290 {
2291     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2292     \__enumext_keyans_show_left:n { #1 }
2293     \bool_set_true:N \l__enumext_wrap_label_v_bool
2294     \__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>`.

```

2295     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2296     \__enumext_keyans_addto_prop:n { #1 }
2297     \__enumext_keyans_store_ref:
2298     \__enumext_keyans_addto_seq:n { #1 }
2299 }

```

(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 (§10.31).

```

2300 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2301 {
2302     \RenewDocumentCommand \item { s o }
2303     {
2304         \bool_if:nTF {##1}
2305         {
2306             \peek_remove_spaces:n
2307             {
2308                 \__enumext_keyans_starred_item:n {##2}
2309             }
2310         }
2311         {
2312             \__enumext_keyans_default_item:n {##2}
2313         }
2314     }
2315 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 11.)

10.29 Redefining `\makeLabel` command

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

10.29.1 Redefining `\makeLabel` for `enumext`

`__enumext_item_starred:`

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

```

2316 \cs_new_protected:Nn \__enumext_item_starred:
2317 {
2318     \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2319     {
2320         \mode_leave_vertical:
2321         \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2322         \makebox[ opt ][ r ]{ \g__enumext_item_symbol_tl }

```

```

2323     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2324   }
2325 }

```

(End of definition for `__enumext_item_starred:`)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makeLabel` for the `enumext` environment.

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

```

2326 \cs_new_protected:Nn \__enumext_make_label:
2327 {
2328   \RenewDocumentCommand \makeLabel { m }
2329   {
2330     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2331     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2332     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2333     {
2334       \__enumext_item_starred:
2335       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2336     }
2337     { ##1 }
2338     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2339     \tl_gclear:N \g__enumext_item_symbol_tl
2340   }
2341 }

```

(End of definition for `__enumext_make_label:`)

10.29.2 Redefining `\makeLabel` for `keyans`

`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment.

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

```

2342 \cs_new_protected:Nn \__enumext_keyans_make_label:
2343 {
2344   \RenewDocumentCommand \makeLabel { m }
2345   {
2346     \tl_use:N \l__enumext_label_fill_left_v_tl
2347     \tl_use:N \l__enumext_label_font_style_v_tl
2348     \bool_if:NTF \l__enumext_wrap_label_v_bool
2349     {
2350       \__enumext_wrapper_label_v:n { ##1 }
2351     }
2352     { ##1 }
2353     \tl_use:N \l__enumext_label_fill_right_v_tl
2354   }
2355 }

```

(End of definition for `__enumext_keyans_make_label:`)

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

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

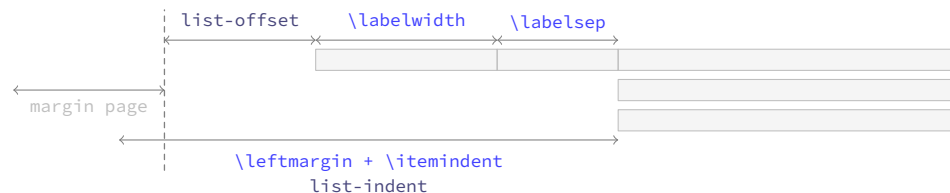


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

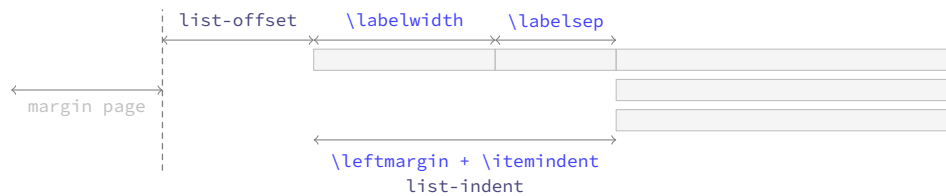
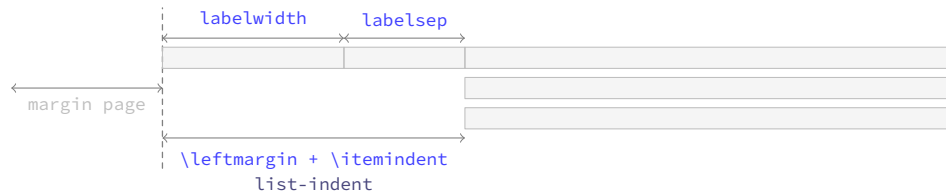
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

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

Figure 10: Representation of horizontal lengths concept in list in `enumext`.Figure 11: Default horizontal lengths in `enumext`.

```

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

```

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

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

```

2356 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
2357 {
2358   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2359   {
2360     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2361     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2362   }
2363   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2364   {
2365     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2366     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2367   }

```

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

```

2368   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2369   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2370   {
2371     \dim_set:Nn #6 { #1 + #2 - #4 }
2372     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2373   }
2374   {
2375     \dim_compare:nNnT { #4 } = { #1 + #2 }
2376     { \dim_set:Nn #6 { \c_zero_dim } }
2377     \dim_compare:nNnT { #4 } < { #1 + #2 }
2378     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2379     \dim_compare:nNnT { #4 } > { #1 + #2 }
2380     {
2381       \dim_set:Nn #6 { -#1 - #2 + #4 }
2382       \dim_set:Nn #6 { #6*-1 }
2383     }
2384     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2385   }
2386 }
2387 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNN { ccccccc }

```

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

10.31 Setting second argument of the lists

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

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments.

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

```

2388 \cs_set_protected:Npn \__enumext_tmp:n #1
2389 {
2390   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2391   {
2392     \__enumext_calc_hspace:cccccc
2393     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2394     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2395     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2396     { l__enumext_leftmargin_tmp_#1_bool }
2397   \clist_map_inline:nn
2398     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2399     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2400   \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2401     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2402   \usecounter { enumX#1 }
2403   \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2404   \str_if_eq:nnTF {#1} { v }
2405   {
2406     \__enumext_keyans_redefine_item:
2407     \__enumext_keyans_make_label:
2408     \__enumext_keyans_fake_item:
2409     \bool_if:cT { l__enumext_show_length_#1_bool }
2410     {
2411       \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2412     }
2413   }
2414   {
2415     \__enumext_redefine_item:
2416     \__enumext_make_label:
2417     \__enumext_use_key_ref:
2418     \__enumext_fake_item:
2419     \bool_if:cT { l__enumext_show_length_#1_bool }
2420     {
2421       \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_int }
2422     }
2423   }
2424 }
2425 }
2426 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

2427 \cs_set_protected:Npn \__enumext_tmp:n #1
2428 {
2429   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2430   {
2431     \__enumext_calc_hspace:cccccc
2432     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2433     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2434     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2435     { l__enumext_leftmargin_tmp_#1_bool }
2436   \clist_map_inline:nn
2437     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2438     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2439   \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2440     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }

```

```

2441 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2442 \skip_zero:N \partopsep
2443 \usecounter { enumX#1 }
2444 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2445 \__enumext_use_key_ref_h:
2446 \str_if_eq:nnTF {#1} { vii }
2447 {
2448   \__enumext_fake_item_vii:
2449   \bool_if:cT { \__enumext_show_length_vii_bool }
2450   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2451 }
2452 {
2453   \__enumext_fake_item_viii:
2454   \bool_if:cT { \__enumext_show_length_#1_bool }
2455   { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2456 }
2457 }
2458 }
2459 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

10.32 The environment enumext

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

```

2460 \NewDocumentEnvironment{enumext}{0} { }
2461 {
2462   \__enumext_current_env:
2463   \__enumext_safe_exec:
2464   \__enumext_parse_keys:n {#1}
2465   \__enumext_before_list:
2466   \__enumext_start_store_level:
2467   \__enumext_start_list:nn
2468   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2469   {
2470     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2471     \__enumext_before_keys_exec:
2472   }
2473   \__enumext_after_args_exec:
2474 }
2475 {
2476   \__enumext_stop_list:
2477   \__enumext_stop_store_level:
2478   \__enumext_after_list:
2479 }

```

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

__enumext_safe_exec: First check the maximum nesting level for the **enumext** environment and set the state of the booleans vars **__enumext_standar_bool** and **\g__enumext_standar_bool** to “true”, the latter only if the environment is NOT nested in the **enumext*** environment.

```

2480 \cs_new_protected:Nn \__enumext_safe_exec:
2481 {
2482   \int_incr:N \__enumext_level_int
2483   \int_compare:nNtT { \__enumext_level_int } > { 4 }
2484   { \msg_fatal:nn { enumext } { list-too-deep } }
2485   \bool_set_true:N \__enumext_standar_bool
2486 }

```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n Parse [*key = val*] by levels in **enumext**. If the variable **__enumext_store_active_bool** is true it will call the function **__enumext_parse_store_keys:n** and reprocess the *keys* to pass them to the storage sequence.

```

2487 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2488 {
2489   \tl_if_novalue:nF {#1}
2490   {
2491     \str_clear:N \__enumext_series_str
2492     \int_compare:nNtTF { \__enumext_level_int } = { 1 }
2493     {

```



```

2494         \keys_set:nn { enumext / level-1 } {#1}
2495         \__enumext_parse_series_resume:n {#1}
2496     }
2497     {
2498         \exp_args:Ne \keys_set:nn
2499         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2500     }
2501     \bool_if:NT \l__enumext_store_active_bool
2502     {
2503         \__enumext_parse_store_keys:n {#1}
2504     }
2505 }
2506 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_parse_store_keys:n

The function __enumext_parse_store_keys:n searches for the values of the `columns` and `columns-sep` keys in the optional arguments per-level in `enumext` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\l__enumext_store_opt_X_tl` which is used by the function `__enumext_store_level_open:`.

```

2507 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2508 {
2509     \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2510     {
2511         \regex_match:nnT { \b columns\b } {#1}
2512         {
2513             \int_set_eq:cc
2514             { l__enumext_store_columns_ \__enumext_level: _int }
2515             { l__enumext_columns_ \__enumext_level: _int }
2516             \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2517             {
2518                 columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2519             }
2520         }
2521     }
2522     \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2523     {
2524         \regex_match:nnT { \b columns-sep\b } {#1}
2525         {
2526             \dim_set_eq:cc
2527             { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2528             { l__enumext_columns_sep_ \__enumext_level: _dim }
2529             \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2530             {
2531                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2532             }
2533         }
2534     }
2535 }

```

(End of definition for __enumext_parse_store_keys:n.)

__enumext_start_store_level:

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

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

```

2536 \cs_new_protected:Nn \__enumext_start_store_level:
2537 {
2538     \bool_lazy_all:nT
2539     {
2540         { \bool_if_p:N \l__enumext_store_active_bool }
2541         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2542         { \bool_not_p:n { \g__enumext_starred_bool } }
2543     }
2544     {
2545         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2546         {
2547             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2548             \__enumext_store_level_open:
2549         }
2550     }

```

```

2551 \bool_lazy_all:nT
2552 {
2553   { \bool_if_p:N \l__enumext_store_active_bool }
2554   { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2555   { \bool_if_p:N \g__enumext_starred_bool }
2556 }
2557 {
2558   \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2559   {
2560     \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2561     \__enumext_store_level_open:
2562   }
2563 }
2564 }
2565 \cs_new_protected:Nn \__enumext_stop_store_level:
2566 {
2567   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2568   {
2569     \__enumext_store_level_close:
2570   }
2571 }

```

(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.

```

2572 \cs_new_protected:Nn \__enumext_before_list:
2573 {
2574   \__enumext_vspace_above:
2575   \__enumext_before_args_exec:

```

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

```

2576 \__enumext_check_ans_exec:

```

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.

```

2577 \dim_compare:nNnT
2578 { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2579 {
2580   \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2581   {
2582     \linewidth
2583     - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2584     - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2585   }

```

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*.

```

2586 \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
2587 \int_gincr:N \g__enumext_minipage_stat_int
2588 \__enumext_mini_addvspace:
2589 \nointerlineskip\noindent
2590 \begin{\__enumext_mini_env*}
2591   { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
2592 }
2593 \__enumext_multicols_start:
2594 }

```

(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.

```

2595 \cs_new_protected:Nn \__enumext_multicols_start:
2596 {
2597   \int_compare:nNt
2598     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2599   {
2600     \dim_compare:nNt
2601       { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2602     {
2603       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
2604       {
2605         ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
2606           + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
2607         ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
2608         - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
2609       }
2610     }
2611     \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
2612     \skip_zero:N \multicolsep
2613     \int_compare:nNt { \__enumext_level_int } > { 1 }
2614     {
2615       \dim_zero:N \columnseprule
2616     }

```

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.

```

2617   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2618   {
2619     \__enumext_multi_addvspace:
2620   }
2621   \raggedcolumns
2622   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2623 }
2624 }

```

(End of definition for `__enumext_multicols_start:`)

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

```

2625 \cs_new_protected:Nn \__enumext_multicols_stop:
2626 {
2627   \int_compare:nNt
2628     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2629   {
2630     \end{multicols}
2631     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2632     {
2633       \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2634     }
2635   }

```

If the `check-ans` key is active, we set the boolean variable `\g__enumext_check_ans_show_bool` to true and copy the stored name to the variable `\g__enumext_store_name_tl`. These variables will be used by the function `__enumext_after_env:n` to display the result of the internal check answer mechanism in the terminal.

```

2636   \bool_lazy_and:nnT
2637     { \bool_if_p:N \__enumext_check_ans_bool }
2638     { \bool_not_p:n { \g__enumext_starred_bool } }
2639   {
2640     \bool_gset_true:N \g__enumext_check_ans_show_bool
2641     \tl_gset:NV \g__enumext_store_name_tl \__enumext_store_name_tl
2642   }
2643 }

```

(End of definition for `__enumext_multicols_stop:`)

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

```

2644 \cs_new_protected:Nn \__enumext_after_list:
2645 {
2646   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
2647   {
2648     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2649     {
2650       \msg_warning:nn { enumext } { missing-miniright }
2651       \miniright
2652     }
2653     \int_gzero:N \g__enumext_minipage_stat_int
2654     \end{\__enumext_mini_env*}
2655     \par\addvspace { \l__enumext_minipage_after_skip }
2656   }
2657   { \__enumext_multicols_stop: }

```

Now apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

2658   \__enumext_after_stop_list:
2659   \__enumext_vspace_below:

```

Finally save the *current value* of the counter in `\g__enumext_resume_int` for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

2660   \bool_set_false:N \l__enumext_standar_bool
2661   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2662   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2663   {
2664     \int_gset_eq:cN
2665     { g__enumext_resume_ \l__enumext_store_name_tl _int }
2666     { \value{enumXi} }
2667   }
2668 }

```

(End of definition for `__enumext_after_list:`)

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

```

2669 \__enumext_after_env:nn {enumext}
2670 {
2671   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2672   {
2673     \bool_if:NT \g__enumext_check_ans_show_bool
2674     {
2675       \__enumext_check_ans_show:
2676     }
2677     \bool_gset_false:N \g__enumext_standar_bool
2678     \bool_gset_false:N \g__enumext_check_ans_show_bool
2679     \tl_gclear:N \g__enumext_store_name_tl
2680   }
2681 }

```

10.33 The environment keyans

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “multiple choice questions”.

`keyans` Now we define the environment `keyans` also based on lists.

```

2682 \NewDocumentEnvironment{keyans}{ 0{ } }
2683 {
2684   \__enumext_keyans_safe_exec:
2685   \__enumext_keyans_parse_keys:n {#1}
2686   \__enumext_before_list_v:
2687   \__enumext_start_list:nn
2688   { \tl_use:N \l__enumext_label_v_tl }
2689   {
2690     \__enumext_list_arg_two_v:

```

```

2691     \__enumext_before_keys_exec_v:
2692   }
2693   \__enumext_after_args_exec_v:
2694 }
2695 {
2696   \__enumext_keyans_check_ans:nn { item }{ keyans }
2697   \__enumext_stop_list:
2698   \__enumext_after_list_v:
2699 }

```

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

`__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.

```

2700 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2701 {
2702   \bool_if:NF \l__enumext_store_active_bool
2703   {
2704     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2705   }
2706   \int_incr:N \l__enumext_keyans_level_int
2707   \bool_set_true:N \l__enumext_keyans_env_bool
2708   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2709   \bool_set_false:N \l__enumext_store_active_bool
2710   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2711   {
2712     \msg_error:nn { enumext } { keyans-nested }
2713   }
2714   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2715   {
2716     \msg_error:nn { enumext } { keyans-wrong-level }
2717   }
2718 }

```

(End of definition for `__enumext_keyans_safe_exec:`.)

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

```

2719 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2720 {
2721   \keys_set:nn { enumext / keyans } { #1 }
2722 }

```

(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.

```

2723 \cs_new_protected:Nn \__enumext_before_list_v:
2724 {
2725   \__enumext_vspace_above_v:
2726   \__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.

```

2727   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2728   {
2729     \dim_set:Nn \l__enumext_minipage_left_v_dim
2730     {
2731       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2732     }

```

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

```

2733   \bool_set_true:N \l__enumext_minipage_active_v_bool

```

```

2734         \int_gincr:N \g__enumext_minipage_stat_int
2735         \__enumext_keyans_mini_addvspace:
2736         \nointerlineskip\noindent
2737         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2738     }

```

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

```

2739     \__enumext_keyans_multicols_start:
2740 }

```

(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.

```

2741 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2742 {
2743     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2744     {

```

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

```

2745         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2746         {
2747             \dim_set:Nn \l__enumext_columns_sep_v_dim
2748             {
2749                 (
2750                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2751                 ) / \l__enumext_columns_v_int
2752                 - \l__enumext_listoffset_v_dim
2753             }
2754         }
2755         \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).

```

2756         \skip_zero:N \multicolsep
2757         \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.

```

2758         \bool_if:NF \l__enumext_minipage_active_v_bool
2759         {
2760             \__enumext_keyans_multi_addvspace:
2761         }
2762         \raggedcolumns
2763         \begin{multicols}{ \l__enumext_columns_v_int }
2764     }
2765 }

```

(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.

```

2766 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2767 {
2768     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2769     {
2770         \end{multicols}
2771         \bool_if:NF \l__enumext_minipage_active_v_bool
2772         {
2773             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2774         }
2775     }
2776 }

```

(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.

```

2777 \cs_new_protected:Nn \__enumext_after_list_v:
2778 {
2779   \bool_if:NTF \l__enumext_minipage_active_v_bool
2780   {
2781     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2782     {
2783       \msg_warning:nn { enumext } { missing-miniright }
2784       \miniright
2785     }
2786     \int_gzero:N \g__enumext_minipage_stat_int
2787     \end{\__enumext_mini_env*}
2788     \par\addvspace{ \l__enumext_minipage_after_skip }
2789   }
2790   { \__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.

```

2791   \bool_set_false:N \l__enumext_keyans_env_bool
2792   \__enumext_after_stop_list_v:
2793   \__enumext_vspace_below_v:
2794 }

```

(End of definition for `__enumext_after_list_v:`)

10.34 The environment `keyanspic` and `\anspic`

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

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

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

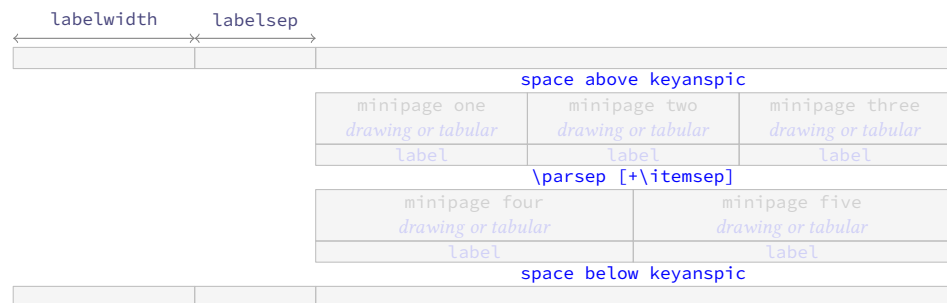


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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?](#).

10.34.1 The command `\anspic`

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

```

2795 \NewDocumentCommand \anspic { s o +m }
2796 {

```

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

```

2797   \bool_if:NF \l__enumext_store_active_bool
2798   {
2799     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2800   }
2801   \int_compare:nNt { \l__enumext_level_int } > { 1 }
2802   {
2803     \msg_error:nn { enumext } { keyanspic-wrong-level }
2804   }

```



```

2805 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2806 {
2807   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2808 }

```

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.

```

2809 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
2810 {
2811   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2812 }
2813 }

```

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

`__enumext_keyans_anspic_code:nnn`

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

```

2814 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2815 {
2816   \stepcounter { enumXvi }
2817   #3 \\\
2818   \bool_if:nT { #1 }
2819   {
2820     \__enumext_keyans_addto_prop:n { #2 }
2821     \__enumext_keyans_store_ref:
2822     \__enumext_keyans_addto_seq:n { #2 }
2823     \bool_lazy_or:nnT
2824     { \bool_if_p:N \__enumext_show_answer_bool }
2825     { \bool_if_p:N \__enumext_show_position_bool }
2826     {
2827       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
2828       \__enumext_keyans_show_left:n { #2 }
2829       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
2830     }
2831   }
2832   \tl_use:N \__enumext_label_font_style_v_tl
2833   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2834 }

```

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

10.34.2 The environment `keyanspic`

`keyanspic`

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

```

2835 \NewDocumentEnvironment{keyanspic}{ o }
2836 {
2837   \__enumext_keyans_pic_safe_exec:
2838   \__enumext_start_list:nn
2839   { }
2840   {
2841     \__enumext_keyans_pic_arg_two:
2842   }

```

We apply the “adjusted” vertical spacing above the environment

```

2843 \vspace { \__enumext_keyans_pic_above_skip }
2844 }

```

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

```

2845 {
2846   \tl_if_novalue:nTF { #1 }
2847   {
2848     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
2849   }
2850   { \__enumext_keyans_pic_do:n { #1 } }
2851   \__enumext_stop_list:
2852   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2853   \setcounter { enumXvi } { 0 }
2854   \vspace { \__enumext_topsep_v_skip }

```

```

2855     %\bool_set_false:N \l__enumext_store_active_bool
2856   }

```

(End of definition for `keyanspic`. This function is documented on page 11.)

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

```

2857 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2858 {
2859   \int_incr:N \l__enumext_keyans_pic_level_int
2860   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2861   {
2862     \msg_error:nn { enumext } { keyanspic-nested }
2863   }
2864 }

```

(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`.

```

2865 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2866 {
2867   \dim_compare:nNnT { #1 } < { 0pt }
2868   { \skip_set:Nn #1 { -#1 } }
2869 }

```

(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.

```

2870 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2871 {

```

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.

```

2872   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2873   \__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.

```

2874   \skip_add:Nn \parsep { \itemsep }
2875   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2876   \dim_zero:N \labelwidth
2877   \dim_zero:N \listparindent
2878   \dim_zero:N \labelsep
2879   \skip_zero:N \partopsep
2880   \skip_zero:N \itemsep

```

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

```

2881   \__enumext_keyans_pic_skip_abs:N \parsep
2882   \skip_set:Nn \l__enumext_keyans_pic_above_skip
2883   {
2884     \box_dp:N \strutbox
2885     + \l__enumext_topsep_v_skip
2886     - \parsep
2887   }
2888   \__enumext_item_std:w \scan_stop:
2889 }

```

(End of definition for `__enumext_keyans_pic_arg_two:.`)

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

```

2890 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2891 {
2892   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2893 }
2894 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

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

```

2895 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2896 {
2897   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2898   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2899   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2900   \int_step_inline:nnn
2901     { \l__enumext_keyans_pic_above_int + 1 }
2902     { \l__enumext_keyans_pic_below_int }
2903     {
2904       \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
2905       \centering
2906       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2907       \__enumext_endminipage:
2908     }
2909   \par
2910 }

```

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

10.35 The `enumext*` environment

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

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content passed to `\item`, we will also add the optional argument ($\langle number \rangle$) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an optional first argument ($\langle number \rangle$).

10.35.1 Functions for item box width

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

```

2911 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2912 {
2913   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2914   {
2915     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2916     {
2917       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2918       / \l__enumext_columns_vii_int
2919     }
2920   }
2921   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2922   \dim_set:Nn \l__enumext_item_width_vii_dim
2923   {
2924     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2925     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2926     - \l__enumext_labelsep_vii_dim
2927   }
2928   \dim_zero_new:N \itemwidth
2929 }

```

(End of definition for `__enumext_starred_columns_set_vii:`.)

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

```

2930 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2931 {
2932   \int_set:Nn \l__enumext_joined_item_vii_int { #1 }
2933   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2934   {
2935     \msg_warning:nnee { enumext } { item-joined }
2936     { \int_use:N \l__enumext_joined_item_vii_int }

```

```

2937         { \int_use:N \l__enumext_columns_vii_int }
2938     \int_set:Nn \l__enumext_joined_item_vii_int
2939     {
2940         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2941     }
2942 }
2943 \int_compare:nNnT
2944 { \l__enumext_joined_item_vii_int }
2945 >
2946 { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2947 {
2948     \msg_warning:nnee { enumext } { item-joined-columns }
2949     { \int_use:N \l__enumext_joined_item_vii_int }
2950     {
2951         \int_eval:n
2952         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2953     }
2954     \int_set:Nn \l__enumext_joined_item_vii_int
2955     {
2956         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2957     }
2958 }

```

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

```

2959     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2960     {
2961         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2962         \int_decr:N \l__enumext_joined_item_aux_vii_int
2963         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2964         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2965         \dim_set:Nn \l__enumext_joined_width_vii_dim
2966         {
2967             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2968             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2969                 + \l__enumext_columns_sep_vii_dim
2970                 ) * \l__enumext_joined_item_aux_vii_int
2971         }
2972         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2973     }
2974     {
2975         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2976         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2977     }
2978 }

```

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

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

```

2979 \cs_new_protected:Nn \__enumext_start_mini_vii:
2980 {
2981     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2982     {
2983         \dim_set:Nn \l__enumext_minipage_left_vii_dim
2984         {
2985             \linewidth
2986             - \l__enumext_minipage_right_vii_dim
2987             - \l__enumext_minipage_hsep_vii_dim
2988         }
2989         \bool_set_true:N \l__enumext_minipage_active_vii_bool
2990         \dim_gset_eq:NN
2991         \g__enumext_minipage_right_vii_dim
2992         \l__enumext_minipage_right_vii_dim
2993         \__enumext_mini_addvspace_vii:
2994         \nointerlineskip\noindent
2995         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2996     }
2997 }

```

(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”.

```

2998 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2999 {
3000     \bool_if:NT \l__enumext_minipage_active_vii_bool
3001     {
3002         \end{\__enumext_mini_env*}
3003         \hfill
3004         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3005     }
3006 }
```

Finally we execute code passed to the `miniright` key stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “right side”.

```

3007 \__enumext_after_env:nn {enumext*}
3008 {
3009     \bool_if:NT \g__enumext_minipage_active_vii_bool
3010     {
3011         \begin{\__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3012         \par\addvspace { \g__enumext_minipage_right_skip }
3013         \bool_if:NF \g__enumext_minipage_center_vii_bool
3014         {
3015             \centering
3016         }
3017         \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3018         \end{\__enumext_mini_env*}
3019         \par\addvspace{ \g__enumext_minipage_after_skip }
3020     }
3021     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3022     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3023     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3024     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3025 }
```

(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.

```

3026 \NewDocumentEnvironment{enumext*}{ o }
3027 {
3028     \__enumext_current_env:
3029     \__enumext_safe_exec_vii:
3030     \__enumext_parse_keys_vii:n {#1}
3031     \__enumext_before_list_vii:
3032     \__enumext_start_store_level_vii:
3033     \__enumext_start_list:nn { }
3034     {
3035         \__enumext_list_arg_two_vii:
3036         \__enumext_before_keys_exec_vii:
3037     }
3038     \__enumext_starred_columns_set_vii:
3039     \item[] \scan_stop:
3040     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3041     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3042 }
3043 {
3044     \__enumext_stop_item_tmp_vii:
3045     \__enumext_remove_extra_parsep_vii:
3046     \__enumext_stop_list:
3047     \__enumext_stop_store_level_vii:
3048     \__enumext_after_list_vii:
3049 }
```

(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`.

```

3050 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3051 {
3052   \int_incr:N \l__enumext_level_h_int
3053   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3054   {
3055     \msg_error:nn { enumext } { nested }
3056   }
3057   \bool_set_true:N \l__enumext_starred_bool
3058 }

```

(End of definition for `__enumext_safe_exec_vii:`)

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

```

3059 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3060 {
3061   \tl_if_novalue:nF {#1}
3062   {
3063     \str_clear:N \l__enumext_series_str
3064     \keys_set:nn { enumext / enumext* } {#1}
3065     \__enumext_parse_series_resume:n {#1}
3066     \bool_if:NT \l__enumext_store_active_bool
3067     {
3068       \__enumext_parse_store_keys_vii:n {#1}
3069     }
3070   }
3071 }

```

(End of definition for `__enumext_parse_keys_vii:n`)

`__enumext_parse_store_keys_vii:n` The function `__enumext_parse_store_keys_vii:n` searches for the values of the `columns` and `columns-sep` keys in the optional argument in `enumext*` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\l__enumext_store_opt_vii_tl` which is used by the function `__enumext_store_level_open_vii:`.

```

3072 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3073 {
3074   \bool_if:NF \l__enumext_store_columns_vii_bool
3075   {
3076     \regex_match:nnT { \b columns\b } {#1}
3077     {
3078       \int_set_eq:NN
3079       \l__enumext_store_columns_vii_int
3080       \l__enumext_columns_vii_int
3081       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3082       {
3083         columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3084       }
3085     }
3086   }
3087   \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3088   {
3089     \regex_match:nnT { \b columns-sep\b } {#1}
3090     {
3091       \dim_set_eq:NN
3092       \l__enumext_store_columns_sep_vii_dim
3093       \l__enumext_columns_sep_vii_dim
3094       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3095       {
3096         columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3097       }
3098     }
3099   }
3100 }

```

(End of definition for `__enumext_parse_store_keys_vii:n`)

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

```

3101 \cs_new_protected:Nn \__enumext_before_list_vii:
3102 {
3103     \__enumext_vspace_above_vii:
3104     \__enumext_check_ans_exec: % need by chek-ans
3105     \__enumext_before_args_exec_vii:
3106     \__enumext_start_mini_vii:
3107 }

```

(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.

```

3108 \cs_new_protected:Nn \__enumext_after_list_vii:
3109 {
3110     \__enumext_stop_mini_vii:
3111     \__enumext_after_stop_list_vii:
3112     \__enumext_vspace_below_vii:
3113     \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
3114     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
3115     {
3116         \int_gset_eq:cN
3117         { g__enumext_resume_ \l__enumext_store_name_tl _int }
3118         { \value{enumXvii} }
3119     }
3120     \bool_lazy_and:nnT
3121     { \bool_if_p:N \g__enumext_starred_bool }
3122     { \bool_if_p:N \l__enumext_check_ans_bool }
3123     {
3124         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
3125         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
3126     }
3127     %\bool_gset_false:N \g__enumext_starred_bool
3128     \bool_set_false:N \l__enumext_starred_bool
3129 }

```

(End of definition for `__enumext_after_list_vii:`.)

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

```

3130 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3131 {
3132     \bool_if:NT \l__enumext_store_active_bool
3133     {
3134         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3135         {
3136             \__enumext_store_level_open_vii:
3137         }
3138     }
3139 }
3140 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3141 {
3142     \bool_if:NT \l__enumext_store_active_bool
3143     {
3144         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3145         {
3146             \__enumext_store_level_close_vii:
3147         }
3148     }
3149 }

```

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

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

`__enumext_start_item_tmp_vii:`

First we will call the function `__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_vii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```
3150 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3151 {
3152   \__enumext_stop_item_tmp_vii:
3153   \int_incr:N \l__enumext_item_column_pos_vii_int
3154   \int_gincr:N \g__enumext_item_count_all_vii_int
3155   \__enumext_item_peek_args_vii:
3156 }
```

(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).

```
3157 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3158 {
3159   \peek_meaning:NTF (
3160     { \__enumext_joined_item_vii:w }
3161     { \__enumext_joined_item_vii:w (1) }
3162 }
```

(End of definition for `__enumext_item_peek_args_vii:`)

`__enumext_joined_item_vii:w`

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

```
3163 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3164 {
3165   \__enumext_starred_joined_item_vii:n {#1}
3166   \peek_meaning_remove:NTF *
3167     { \__enumext_starred_item_vii:w }
3168     { \__enumext_standard_item_vii:w }
3169 }
```

(End of definition for `__enumext_joined_item_vii:w`)

`__enumext_standard_item_vii:w`

The function `__enumext_standard_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_vii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_vii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[<custom>]` by means of the function `__enumext_start_item_vii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_vii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_vii:w [__enumext_label_vii_tl]`.

```
3170 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3171 {
3172   \bool_set_false:N \l__enumext_item_starred_vii_bool
3173   \peek_meaning:NTF [
3174     {
3175       \bool_set_eq:NN
3176       \l__enumext_wrap_label_vii_bool
3177       \l__enumext_wrap_label_opt_vii_bool
3178       \__enumext_start_item_vii:w
3179     }
3180     {
3181       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3182       \legacy_if_set_true:n { @noitemarg }
3183       \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
3184     }
3185 }
```

(End of definition for `__enumext_standard_item_vii:w`)

```

\__enumext_starred_item_vii:w
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii:w

```

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

```

3186 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3187 {
3188   \bool_set_true:N \__enumext_item_starred_vii_bool
3189   \bool_set_true:N \__enumext_wrap_label_vii_bool
3190   \peek_meaning:NTF [
3191     { \__enumext_starred_item_vii_aux_i:w }
3192     { \__enumext_starred_item_vii_aux_ii:w }
3193   }
3194   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3195   {
3196     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3197     \__enumext_starred_item_vii_aux_ii:w
3198   }
3199   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3200   {
3201     \peek_meaning:NTF [
3202       { \__enumext_starred_item_vii_aux_iii:w }
3203       {
3204         \dim_set_eq:NN
3205           \l__enumext_item_symbol_sep_vii_dim
3206           \l__enumext_labelsep_vii_dim
3207         \legacy_if_set_true:n { @noitemarg }
3208         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3209       }
3210     }
3211     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3212     {
3213       \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3214       \legacy_if_set_true:n { @noitemarg }
3215       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3216     }

```

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

10.35.3 Real definition of `\item` in `enumext*`

```

\__enumext_start_item_vii:w

```

The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment.

The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to the value of `__enumext_stop_item_vii:` which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

3217 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3218 {
3219   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3220   \legacy_if:nT { @noitemarg }
3221   {
3222     \legacy_if_set_false:n { @noitemarg }
3223     \legacy_if:nT { @nmbrrlist }
3224     {
3225       \bool_if:NT \l__enumext_hyperref_bool
3226       {
3227         \legacy_if_set_true:n { @hyper@item }
3228       }
3229       \refstepcounter{enumXvii}
3230       \bool_if:NT \l__enumext_check_ans_bool
3231       {
3232         \int_gincr:N \g__enumext_count_item_number_int
3233       }
3234     }
3235   }

```

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.

```

3236 \group_begin:
3237 \lrbox{ \l__enumext_item_text_vii_box }
3238 \bool_if:NF \l__enumext_footnotes_key_bool
3239 {
3240   \__enumext_renew_footnote:
3241 }
3242 \bool_if:NT \l__enumext_item_starred_vii_bool
3243 {
3244   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3245   {
3246     \tl_gset_eq:NN
3247       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3248   }
3249   \mode_leave_vertical:
3250   \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3251   \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3252   \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3253   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3254 }
3255 \group_begin:
3256 \tl_use:N \l__enumext_label_font_style_vii_tl
3257 \bool_if:NTF \l__enumext_wrap_label_vii_bool
3258 {
3259   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3260     { \__enumext_wrapper_label_vii:n {#1} }
3261 }
3262 {
3263   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3264 }
3265 \group_end:
3266 \skip_horizontal:N \l__enumext_labelsep_vii_dim
3267 \tl_use:N \l__enumext_after_list_args_vii_tl
3268 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3269   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3270   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3271   \tl_use:N \l__enumext_fake_item_indent_vii_tl
3272 }

```

(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.

```

3273 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3274 {
3275   \__enumext_endminipage:
3276   \endlrbox
3277   \group_end:
3278   \box_set_wd:Nn \l__enumext_item_text_vii_box
3279   {
3280     \l__enumext_joined_width_vii_dim
3281     + \l__enumext_labelwidth_vii_dim
3282     + \l__enumext_labelsep_vii_dim
3283   }
3284   \int_set:Nn \hbadness { 10000 }
3285   \box_use:N \l__enumext_item_text_vii_box
3286   \bool_if:NF \l__enumext_footnotes_key_bool
3287   {
3288     \__enumext_print_footnote:
3289   }
3290   \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3291   {
3292     \par\noindent
3293     \int_zero:N \l__enumext_item_column_pos_vii_int
3294   }
3295   { \hspace{ \l__enumext_columns_sep_vii_dim } }
3296 }

```

(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.

```

3297 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3298 {
3299   \int_compare:nNnT
3300     {
3301       \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3302     }
3303     =
3304     { \c_zero_int }
3305     {
3306       \par
3307       \vspace{ -\l__enumext_itemsep_vii_skip }
3308       \int_gzero:N \g__enumext_item_count_all_vii_int
3309     }
3310 }

```

(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`.

```

3311 \__enumext_after_env:nn {enumext*}
3312 {
3313   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3314   {
3315     \bool_if:NT \g__enumext_check_ans_show_h_bool
3316     {
3317       \__enumext_check_ans_show:
3318     }
3319     \bool_gset_false:N \g__enumext_starred_bool
3320     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3321     \tl_gclear:N \g__enumext_store_name_tl
3322   }
3323 }

```

10.36 The `keyans*` environment

10.36.1 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.

```

3324 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3325 {
3326   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3327   {
3328     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3329     {
3330       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3331       / \l__enumext_columns_viii_int
3332     }
3333   }
3334   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3335   \dim_set:Nn \l__enumext_item_width_viii_dim
3336   {
3337     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3338     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3339     - \l__enumext_labelsep_viii_dim
3340   }
3341   \dim_zero_new:N \itemwidth
3342 }

```

(End of definition for `__enumext_starred_columns_set_viii:`.)

`__enumext_starred_joined_item_viii:n`

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

```

3343 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3344 {
3345   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3346   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3347   {
3348     \msg_warning:nnee { enumext } { item-joined }
3349     { \int_use:N \l__enumext_joined_item_viii_int }

```

```

3350         { \int_use:N \l__enumext_columns_viii_int }
3351     \int_set:Nn \l__enumext_joined_item_viii_int
3352     {
3353         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3354     }
3355 }
3356 \int_compare:nNnT
3357 { \l__enumext_joined_item_viii_int }
3358 >
3359 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3360 {
3361     \msg_warning:nnee { enumext } { item-joined-columns }
3362     { \int_use:N \l__enumext_joined_item_viii_int }
3363     {
3364         \int_eval:n
3365         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3366     }
3367     \int_set:Nn \l__enumext_joined_item_viii_int
3368     {
3369         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3370     }
3371 }
3372 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3373 {
3374     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3375     \int_decr:N \l__enumext_joined_item_aux_viii_int
3376     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3377     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3378     \dim_set:Nn \l__enumext_joined_width_viii_dim
3379     {
3380         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3381         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3382           + \l__enumext_columns_sep_viii_dim
3383         ) * \l__enumext_joined_item_aux_viii_int
3384     }
3385     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3386 }
3387 {
3388     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3389     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3390 }
3391 }

```

(End of definition for `__enumext_starred_joined_item_viii:n`)

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

```

3392 \cs_new_protected:Nn \__enumext_start_mini_viii:
3393 {
3394     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3395     {
3396         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3397         {
3398             \linewidth
3399             - \l__enumext_minipage_right_viii_dim
3400             - \l__enumext_minipage_hsep_viii_dim
3401         }
3402         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3403         \dim_gset_eq:NN
3404             \g__enumext_minipage_right_viii_dim
3405             \l__enumext_minipage_right_viii_dim
3406         \__enumext_mini_addvspace_viii:
3407         \nointerlineskip\noindent
3408         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3409     }
3410 }
3411 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3412 {
3413     \bool_if:NT \l__enumext_minipage_active_viii_bool
3414     {
3415         \end{__enumext_mini_env*}
3416         \hfill

```

```

3417         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3418     }
3419 }
3420 \__enumext_after_env:nn {keyans*}
3421 {
3422     \bool_if:NT \g__enumext_minipage_active_viii_bool
3423     {
3424         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3425         \par\addvspace { \g__enumext_minipage_right_skip }
3426         \bool_if:NF \g__enumext_minipage_center_viii_bool
3427         {
3428             \centering
3429         }
3430         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3431         \end{__enumext_mini_env*}
3432         \par\addvspace{ \g__enumext_minipage_after_skip }
3433     }
3434     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3435     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3436     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3437     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3438 }

```

(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.

```

3439 \NewDocumentEnvironment{keyans*}{ o }
3440 {
3441     \__enumext_safe_exec_viii:
3442     \__enumext_parse_keys_viii:n {#1}
3443     \__enumext_before_list_viii:
3444     \__enumext_start_list:nn { }
3445     {
3446         \__enumext_list_arg_two_viii:
3447         \__enumext_before_keys_exec_viii:
3448     }
3449     \__enumext_starred_columns_set_viii:
3450     \item[] \scan_stop:
3451     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3452     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3453 }
3454 {
3455     \__enumext_stop_item_tmp_viii:
3456     \__enumext_remove_extra_parsep_viii:
3457     \__enumext_keyans_check_ans:nn { item }{ keyans* }
3458     \__enumext_stop_list:
3459     \__enumext_after_list_viii:
3460 }

```

(End of definition for keyans*. This function is documented on page 10.)

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

```

3461 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3462 {
3463     \int_incr:N \l__enumext_keyans_level_h_int
3464     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3465     {
3466         \msg_error:nn { enumext } { nested }
3467     }
3468     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3469     \bool_set_false:N \l__enumext_store_active_bool
3470     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3471     {
3472         \msg_error:nn { enumext } { keyans-wrong-level }
3473     }
3474 }

```

(End of definition for __enumext_safe_exec_viii:.)

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

```

3475 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3476 {
3477   \tl_if_novalue:nF {#1}
3478   {
3479     \keys_set:nn { enumext / keyans* } {#1}
3480   }
3481 }

```

(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*.

```

3482 \cs_new_protected:Nn \__enumext_before_list_viii:
3483 {
3484   \__enumext_vspace_above_viii:
3485   \__enumext_before_args_exec_viii:
3486   \__enumext_start_mini_viii:
3487 }

```

(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.

```

3488 \cs_new_protected:Nn \__enumext_after_list_viii:
3489 {
3490   \__enumext_stop_mini_viii:
3491   \__enumext_after_stop_list_viii:
3492   \__enumext_vspace_below_viii:
3493 }

```

(End of definition for `__enumext_after_list_viii:`.)

10.36.2 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`.

```

3494 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3495 {
3496   \__enumext_stop_item_tmp_viii:
3497   \int_incr:N \l__enumext_item_column_pos_viii_int
3498   \int_gincr:N \g__enumext_item_count_all_viii_int
3499   \__enumext_item_peek_args_viii:
3500 }

```

(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).

```

3501 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3502 {
3503   \peek_meaning:NTF (
3504     { \__enumext_joined_item_viii:w }
3505     { \__enumext_joined_item_viii:w (1) }
3506   }

```

(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_standard_item_viii:w`.

```

3507 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3508 {
3509     \__enumext_starred_joined_item_viii:n {#1}
3510     \peek_meaning_remove:NTF *
3511     { \__enumext_starred_item_viii:w }
3512     { \__enumext_standard_item_viii:w }
3513 }

```

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

`__enumext_standard_item_viii:w` The function `__enumext_standard_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]`.

```

3514 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3515 {
3516     \bool_set_false:N \l__enumext_item_starred_viii_bool
3517     \peek_meaning:NTF [
3518     {
3519         \bool_set_eq:NN
3520         \l__enumext_wrap_label_viii_bool
3521         \l__enumext_wrap_label_opt_viii_bool
3522         \__enumext_start_item_viii:w
3523     }
3524     {
3525         \bool_set_true:N \l__enumext_wrap_label_viii_bool
3526         \legacy_if_set_true:n { @noitemarg }
3527         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3528     }
3529 }

```

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

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

```

\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
3530 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3531 {
3532     \bool_set_true:N \l__enumext_item_starred_viii_bool
3533     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3534     \peek_meaning:NTF [
3535     { \__enumext_starred_item_viii_aux_i:w }
3536     { \__enumext_starred_item_viii_aux_ii:w }
3537 }

```

The optional argument will be captured in the variables `\l__enumext_keyans_tmpa_tl` and `\l__enumext_keyans_tmpb_tl` which we will use later for the implementation of the `show-ans` and `show-pos` keys together with the stored in `⟨sequence⟩` and `⟨prop list⟩`.

```

3538 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3539 {
3540     \tl_clear:N \l__enumext_store_keyans_label_tl
3541     \tl_if_no_value:nF { #1 }
3542     {
3543         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3544         {
3545             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
3546             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3547         }
3548         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3549     }
3550     \__enumext_starred_item_viii_aux_ii:w
3551 }
3552 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w

```



```

3553 {
3554   \legacy_if_set_true:n { @noitemarg }
3555   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
3556 }

```

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

`__enumext_starred_item_exec:`

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

```

3557 \cs_new_protected:Nn \__enumext_starred_item_exec:
3558 {
3559   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \__enumext_label_viii_tl }
3560   \__enumext_store_addto_prop:V \__enumext_store_keyans_label_tl
3561   \__enumext_keyans_store_ref:
3562   \tl_put_left:Ne \__enumext_store_keyans_label_tl { \item }
3563   \__enumext_keyans_addto_seq_link:
3564   \bool_if:NT \__enumext_show_answer_bool
3565   {
3566     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3567   }
3568   \bool_if:NT \__enumext_show_position_bool
3569   {
3570     \tl_set:Ne \__enumext_mark_answer_sym_tl
3571     {
3572       \group_begin:
3573       \exp_not:N \normalfont
3574       \exp_not:N \footnotesize [ \int_eval:n
3575       {
3576         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3577       }
3578       ]
3579       \group_end:
3580     }
3581     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
3582   }
3583 }

```

(End of definition for `__enumext_starred_item_exec:`.)

Real definition of `\item in keyans*`

`__enumext_start_item_viii:w`

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

```

3584 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3585 {
3586   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3587   \legacy_if:nT { @noitemarg }
3588   {
3589     \legacy_if_set_false:n { @noitemarg }
3590     \legacy_if:nT { @nmbrlist }
3591     {
3592       \bool_if:NT \__enumext_hyperref_bool
3593       {
3594         \legacy_if_set_true:n { @hyper@item }
3595       }
3596       \refstepcounter{enumXviii}
3597     }
3598   }

```

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

```

3599   \group_begin:
3600   \lrbox{ \__enumext_item_text_viii_box }
3601   \bool_if:NF \__enumext_footnotes_key_bool
3602   {
3603     \__enumext_renew_footnote:
3604   }
3605   \bool_if:NT \__enumext_item_starred_viii_bool
3606   {
3607     \__enumext_starred_item_exec:
3608   }

```

```

3609     \group_begin:
3610     \tl_use:N \l__enumext_label_font_style_viii_tl
3611     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3612     {
3613         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3614         { \l__enumext_wrapper_label_viii:n {#1} }
3615     }
3616     {
3617         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3618     }
3619 \group_end:
3620 \skip_horizontal:N \l__enumext_labelsep_viii_dim
3621 \tl_use:N \l__enumext_after_list_args_viii_tl
3622 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3623 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3624 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3625 \bool_if:NT \l__enumext_item_starred_viii_bool
3626 {
3627     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3628     \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent.
3629 }
3630 {
3631     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3632 }
3633 }

```

(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.

```

3634 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3635 {
3636     \__enumext_endminipage:
3637     \endlrbox
3638     \group_end:
3639     \box_set_wd:Nn \l__enumext_item_text_viii_box
3640     {
3641         \l__enumext_joined_width_viii_dim
3642         + \l__enumext_labelwidth_viii_dim
3643         + \l__enumext_labelsep_viii_dim
3644     }
3645     \int_set:Nn \hbadness { 10000 }
3646     \box_use:N \l__enumext_item_text_viii_box
3647     \bool_if:NF \l__enumext_footnotes_key_bool
3648     {
3649         \__enumext_print_footnote:
3650     }
3651     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3652     {
3653         \par\noindent
3654         \int_zero:N \l__enumext_item_column_pos_viii_int
3655     }
3656     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3657 }

```

(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.

```

3658 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3659 {
3660     \int_compare:nNnT
3661     {
3662         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3663     }
3664     =
3665     { \c_zero_int }
3666     {
3667         \par
3668         \vspace{ -\l__enumext_itemsep_viii_skip }

```

```

3669         \int_gzero:N \g__enumext_item_count_all_viii_int
3670     }
3671 }

```

(End of definition for `__enumext_remove_extra_parsep_viii::`)

10.37 The command `\getkeyans`

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

```

3672 \NewDocumentCommand \getkeyans { m }
3673 {
3674     \exp_args:Ne \__enumext_getkeyans_aux:n
3675     { \tl_to_str:e { \text_expand:n {#1} } }
3676 }

```

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

`__enumext_getkeyans_aux:n` The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the $\langle \text{argument} \rangle$ using “:”. If “:” is omitted it will return an error.

```

3677 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3678 {
3679     \str_if_in:nnTF {#1} { : }
3680     {
3681         \use:e
3682         {
3683             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3684             { {##1} {##2} }
3685         }
3686         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3687     }
3688     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3689 }

```

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

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

```

3690 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3691 {
3692     \prop_if_exist:cF { g__enumext_#1_prop }
3693     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3694     \group_begin:
3695     \prop_item:cn { g__enumext_#1_prop }{#2}
3696     \group_end:
3697 }

```

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

10.38 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the $\langle \text{sequence} \rangle$ defined by the `save-ans` key. The first thing we will do is to define a set of $\langle \text{keys} \rangle$ with which we will control the options of the different nesting levels for the `enumext` and `enumext*` environment by storing the values of these in the token list variables `\l__enumext_print_keyans_X_tl`.

```

3698 \keys_define:nn { keyanskey / print }
3699 {
3700     level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3701                     {
3702                         \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3703                     },
3704     level-1 .initial:n = { label=\arabic*, nosep, columns=2, first=\small, font=\small },
3705     level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3706                     {
3707                         \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3708                     },
3709     level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3710     level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3711                     {
3712                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3713                     },

```

```

3714   level-3 .initial:n = { nosep, label=\roman*, first=\small, font=\small },
3715   level-4 .code:n    = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3716                       {
3717                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3718                       },
3719   level-4 .initial:n = { nosep, label=\Alph*, first=\small, font=\small },
3720   level-* .code:n    = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3721                       {
3722                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3723                       },
3724   level-* .initial:n = { label=\arabic*, nosep, columns=2, first=\small, font=\small },
3725 }

```

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

```

3726 \NewDocumentCommand \printkeyans { s O{ } m }
3727 {
3728   \group_begin:
3729   \tl_use:N \l__enumext_print_keyans_i_tl
3730   \tl_use:N \l__enumext_print_keyans_ii_tl
3731   \tl_use:N \l__enumext_print_keyans_iii_tl
3732   \tl_use:N \l__enumext_print_keyans_iv_tl
3733   \tl_use:N \l__enumext_print_keyans_vii_tl
3734   \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3735   \group_end:
3736 }

```

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

`__enumext_printkeyans:nnn`

The internal function `__enumext_printkeyans:nnn` will check for the existence of the *(sequence)*, if it does not exist it will return an error message, then it will fetch the content specified by the first argument mapping the *(sequence)*.

#1: starred
#2: key-val
#3: seq-name

```

3737 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3738 {
3739   \seq_if_exist:cTF { g__enumext_#3_seq }
3740   {
3741     \seq_if_empty:cF { g__enumext_#3_seq }
3742     {
3743       %%\seq_show:c { g__enumext_#3_seq }
3744       \bool_if:nTF {#1}
3745       {
3746         \begin{enumext*}[#2]
3747         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3748         \end{enumext*}
3749       }
3750       {
3751         \begin{enumext}[#2]
3752         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3753         \end{enumext}
3754       }
3755     }
3756   }
3757   {
3758     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3759   }
3760 }

```

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

10.39 The command `\setenumext`

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

```

3761 \keys_define:nn { enumext / meta-families }
3762 {
3763   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3764   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
3765   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
3766   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
3767   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },

```

```

3768     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3769     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3770     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3771     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3772     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3773     print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3774     print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3775     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3776 }

```

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

```

3777 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3778 {
3779     level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3780     keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3781 }

```

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

```

3782 \NewDocumentCommand \setenumext { o +m }
3783 {
3784     \tl_if_novalue:nTF {#1}
3785     {
3786         \seq_map_inline:Nn \c__enumext_all_families_seq
3787     }
3788     {
3789         \seq_clear:N \l__enumext_setkey_tmpa_seq
3790         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3791         \int_set:Nn \l__enumext_setkey_tmpa_int
3792         {
3793             \seq_count:N \l__enumext_setkey_tmpb_seq
3794         }
3795         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3796         {
3797             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3798             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3799             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3800             {
3801                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3802             }
3803         }
3804         {
3805             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3806         }
3807         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3808         { \seq_map_inline:Nn \c__enumext_all_families_seq }
3809         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3810     }
3811     {
3812         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3813     }
3814 }

```

(End of definition for `\setenumext`. This function is documented on page 5.)

`__enumext_set_parse:n`
`__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```

3815 \cs_new_protected:Npn \__enumext_set_parse:n #1
3816 {
3817     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3818     \int_step_inline:nnn { 0 } { 4 } % <- max level
3819     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3820     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3821     {
3822         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3823         { \tl_trim_spaces:n {#1} }
3824     }
3825     { \__enumext_set_error:nn {#1} { } }
3826 }
3827 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3828 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

10.40 Messages

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

```

3829 \msg_new:nnn { enumext } { package-load }
3830 {
3831   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3832 }

3833 \msg_new:nnn { enumext } { package-not-load }
3834 {
3835   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3836 }

3837 \msg_new:nnn { enumext } { package-load-foot }
3838 {
3839   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3840 }
```

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

```

3841 \msg_new:nnn { enumext } { counters }
3842 {
3843   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3844   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3845 }
```

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

```

3846 \msg_new:nnn { enumext } { invalid-key }
3847 {
3848   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3849 }
3850 \msg_new:nnn { enumext } { unknown-key-family }
3851 {
3852   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3853 }
```

Messages used in length calculation.

```

3854 \msg_new:nnn { enumext } { width-negative }
3855 {
3856   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3857   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
3858 }
3859 \msg_new:nnn { enumext } { width-zero }
3860 {
3861   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3862   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
3863 }
```

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

```

3864 \msg_new:nnn { enumext } { list-lengths }
3865 {
3866   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3867   \__enumext_show_length:nnn { dim } { labelsep } { #1}
3868   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
3869   \__enumext_show_length:nnn { dim } { itemindent } { #1}
3870   \__enumext_show_length:nnn { dim } { leftmargin } { #1}
3871   \__enumext_show_length:nnn { dim } { rightmargin } { #1}
3872   \__enumext_show_length:nnn { dim } { listparindent } { #1}
3873   \__enumext_show_length:nnn { skip } { topsep } { #1}
3874   \__enumext_show_length:nnn { skip } { parsep } { #1}
3875   \__enumext_show_length:nnn { skip } { partopsep } { #1}
3876   \__enumext_show_length:nnn { skip } { itemsep } { #1}
3877   ****
3878 }
```

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

```

3879 \msg_new:nnn { enumext } { list-lengths-not-nested }
3880 {
3881   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3882   \__enumext_show_length:nnn { dim } { labelsep } { #1}
3883   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
3884   \__enumext_show_length:nnn { dim } { itemindent } { #1}
3885   \__enumext_show_length:nnn { dim } { leftmargin } { #1}
3886   \__enumext_show_length:nnn { dim } { rightmargin } { #1}
3887   \__enumext_show_length:nnn { dim } { listparindent } { #1}
```

```

3888     \__enumext_show_length:nnn { skip } { topsep } {#1}
3889     \__enumext_show_length:nnn { skip } { parsep } {#1}
3890     \__enumext_show_length:nnn { skip } { partopsep } {#1}
3891     \__enumext_show_length:nnn { skip } { itemsep } {#1}
3892     *****
3893 }

```

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

```

3894 \msg_new:nnn { enumext } { items-same-answer }
3895 {
3896     *****~Checking~answers~on~'#1'~OK~*****\\
3897     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3898     *****
3899     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3900 }
3901 \msg_new:nnn { enumext } { item-different-answer }
3902 {
3903     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3904     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3905 }

```

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

```

3906 \msg_new:nnn { enumext } { missing-starred }
3907 {
3908     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3909 }

```

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

```

3910 \msg_new:nnn { enumext } { list-too-deep }
3911 {
3912     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
3913     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3914 }

```

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

```

3915 \msg_new:nnn { enumext } { anskey-wrong-place }
3916 {
3917     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3918     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3919 }
3920 \msg_new:nnn { enumext } { anspic-wrong-place }
3921 {
3922     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3923     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3924 }
3925 \msg_new:nnn { enumext } { command-wrong-place }
3926 {
3927     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3928     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3929 }

```

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

```

3930 \msg_new:nnn { enumext } { keyans-nested }
3931 {
3932     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3933 }
3934 \msg_new:nnn { enumext } { keyans-wrong-level }
3935 {
3936     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context::~ \\
3937     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3938 }
3939 \msg_new:nnn { enumext } { wrong-place }
3940 {
3941     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context::~ \\
3942     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3943 }
3944 \msg_new:nnn { enumext } { keyanspic-nested }
3945 {
3946     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context::~.
3947 }
3948 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3949 {
3950     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context::~ \\

```

```

3951     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3952 }

```

Messages used by `\getkeyans` command.

```

3953 \msg_new:nnn { enumext } { undefined-storage-anskey }
3954 {
3955     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3956 }

```

Messages used by `\miniright` command.

```

3957 \msg_new:nnn { enumext } { missing-miniright }
3958 {
3959     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
3960     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3961 }
3962 \msg_new:nnn { enumext } { wrong-miniright-place }
3963 {
3964     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3965     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3966 }
3967 \msg_new:nnn { enumext } { wrong-miniright-use }
3968 {
3969     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3970     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3971 }

```

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

```

3972 \msg_new:nnn { enumext } { nested }
3973 {
3974     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3975 }
3976 \msg_new:nnn { enumext } { item-joined }
3977 {
3978     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3979 }
3980 \msg_new:nnn { enumext } { item-joined-columns }
3981 {
3982     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3983 }

```

10.41 Finish package

Finish package implementation.

```

3984 \file_input_stop:
3985 </package>

```


enumXviii 23, 30
 enumXvii 23, 30, 90
 enumXvi 23, 30
 enumXv 23, 30

cs commands:

\cs_generate_variant:Nn 336, 352, 551, 567, 1673,
 1682, 1687, 1767, 2387, 2894
 \cs_if_exist:NTF 306
 \cs_new:Nn 188
 \cs_new:Npn 198, 1434, 1443, 1452
 \cs_new_eq:NN 242, 243, 244, 248, 249, 281, 282, 285,
 286
 \cs_new_protected:Nn . 212, 253, 417, 437, 469, 732,
 736, 740, 744, 748, 752, 756, 760, 764, 768, 772, 776,
 780, 784, 788, 792, 828, 840, 864, 881, 892, 916, 991,
 1015, 1032, 1094, 1111, 1133, 1168, 1174, 1249, 1263,
 1277, 1288, 1299, 1310, 1321, 1332, 1363, 1375, 1387,
 1410, 1538, 1574, 1581, 1688, 1712, 1719, 1747, 1754,
 1871, 2003, 2018, 2046, 2076, 2121, 2133, 2140, 2192,
 2196, 2215, 2266, 2300, 2316, 2326, 2342, 2480, 2536,
 2565, 2572, 2595, 2625, 2644, 2700, 2723, 2741, 2766,
 2777, 2814, 2857, 2870, 2890, 2895, 2911, 2979, 2998,
 3050, 3101, 3108, 3130, 3140, 3157, 3297, 3324, 3392,
 3411, 3461, 3482, 3488, 3501, 3557, 3658
 \cs_new_protected:Npn 180, 184, 289, 304, 321, 331,
 337, 425, 444, 538, 552, 1196, 1215, 1395, 1418, 1463,
 1475, 1502, 1674, 1683, 1803, 1948, 1960, 1982, 2056,
 2098, 2106, 2225, 2243, 2277, 2289, 2356, 2390, 2429,
 2487, 2507, 2719, 2865, 2930, 3059, 3072, 3163, 3170,
 3186, 3194, 3199, 3211, 3343, 3475, 3507, 3514, 3530,
 3538, 3552, 3677, 3690, 3737, 3815, 3827
 \cs_new_protected_nopar:Nn . . . 3150, 3273, 3494,
 3634
 \cs_new_protected_nopar:Npn 3217, 3584
 \cs_set:Nn 1953
 \cs_set:Npn 1881, 1919, 3683
 \cs_set_eq:NN . . 3040, 3041, 3219, 3451, 3452, 3586
 \cs_set_protected:Nn 204, 656, 672, 684, 696
 \cs_set_protected:Npn . 30, 44, 52, 64, 70, 98, 136,
 148, 155, 206, 353, 375, 404, 485, 505, 568, 588, 632,
 651, 708, 717, 796, 813, 1232, 1520, 1594, 1630, 1648,
 1873, 2007, 2176, 2388, 2427
 \cs_to_str:N 323, 346

D

\d 194
 \DeclareDocumentEnvironment 909

dim commands:

\dim_abs:n 2361, 2366
 \dim_add:Nn 2875
 \dim_compare:nNnTF . 658, 674, 686, 698, 1198, 1217,
 2358, 2363, 2369, 2375, 2377, 2379, 2577, 2600, 2727,
 2745, 2867, 2913, 2981, 3326, 3394
 \dim_compare:nTF 1844
 \dim_gset_eq:NN 2990, 3403
 \dim_gzero:N 3024, 3437
 \dim_new:N 40, 47, 48, 49, 66, 92, 105, 115, 164, 165, 171
 \dim_set:Nn . . 334, 646, 1660, 2257, 2361, 2366, 2368,
 2371, 2372, 2376, 2378, 2381, 2382, 2384, 2580, 2603,
 2729, 2747, 2897, 2915, 2922, 2965, 2983, 3213, 3328,
 3335, 3378, 3396
 \dim_set_eq:NN 492, 512, 528, 532, 2252, 2399, 2438,
 2526, 2611, 2755, 2972, 2975, 2976, 3091, 3204, 3385,
 3388, 3389
 \dim_use:N 659, 667, 1199, 1205, 1757, 1760, 1765, 2321,
 2323, 2578, 2583, 2584, 2591, 2601, 2605, 2606, 2608

\dim_zero:N 2615, 2757, 2876, 2877, 2878
 \dim_zero_new:N 2928, 3341
 \c_zero_dim 661, 675, 687, 699, 1199, 1217, 1846, 2358,
 2363, 2369, 2376, 2578, 2601, 2727, 2745, 2913, 2981,
 3326, 3394

E

\end . . 1202, 1220, 1716, 1751, 2630, 2654, 2770, 2787, 3002,
 3018, 3415, 3431, 3748, 3753
 \endlist 28
 \endlist 243
 \endlrbox 3276, 3637
 \endminipage 28
 \endminipage 249
 enumext 5, 2460

enumext internal commands:

\g__enumext_ __enumext_store_name_tl
 _prop 66
 __enumext_add_pre_parsep: . . . 41, 838, 840, 840
 __enumext_after_args_exec: . 39, 732, 744, 2473
 __enumext_after_args_exec_v: . 39, 40, 748, 760,
 2693
 __enumext_after_args_exec_vii: . . . 764, 788
 __enumext_after_args_exec_viii: 792
 __enumext_after_env:n 77
 __enumext_after_env:nn . . 78, 92, 184, 184, 2669,
 3007, 3311, 3420
 __enumext_after_hyperref: . . . 29, 251, 253, 253
 __enumext_after_list: 78, 88, 95, 2478, 2644, 2644
 \l__enumext_after_list_args_v_tl 762
 \l__enumext_after_list_args_vii_tl 790, 3267
 \l__enumext_after_list_args_viii_tl 794, 3621
 __enumext_after_list_v: . . 81, 2698, 2777, 2777
 __enumext_after_list_vii: . . . 3048, 3108, 3108
 __enumext_after_list_viii: . . 3459, 3488, 3488
 __enumext_after_star_env:nn 86
 __enumext_after_stop_list: . . . 38, 40, 732, 740,
 2658
 __enumext_after_stop_list_v: 39, 748, 756, 2792
 \l__enumext_after_stop_list_v_tl 758
 __enumext_after_stop_list_vii: 764, 780, 3111
 \l__enumext_after_stop_list_vii_tl . . . 782
 __enumext_after_stop_list_viii: . 784, 3491
 \l__enumext_after_stop_list_viii_tl . . . 786
 \l__enumext_align_label_vii_str . . 3259, 3263
 \l__enumext_align_label_viii_str . 3613, 3617
 \l__enumext_align_label_X_str 155
 \c__enumext_all_envs_clist . . 175, 374, 587, 650,
 716, 731, 812, 1248
 \c__enumext_all_families_seq . . 101, 3777, 3786,
 3808
 __enumext_anskey_wrapper:n 1598, 1958
 __enumext_at_begin_document:n . . 28, 180, 180,
 240, 246
 __enumext_before_args_exec: 38, 732, 732, 2575
 __enumext_before_args_exec_v: . . 39, 748, 748,
 2726
 __enumext_before_args_exec_vii: . . 764, 764,
 3105
 __enumext_before_args_exec_viii: 768, 3485
 __enumext_before_keys_exec: 38, 732, 736, 2471
 __enumext_before_keys_exec_v: . . 39, 748, 752,
 2691
 __enumext_before_keys_exec_vii 764
 __enumext_before_keys_exec_vii: 39, 772, 3036

```

\__enumext_before_keys_exec_viii: .. 39, 776,
    3447
\__enumext_before_list: ... 76, 2465, 2572, 2572
\__enumext_before_list_v: . 79, 2686, 2723, 2723
\__enumext_before_list_vii: 88, 3031, 3101, 3101
\__enumext_before_list_viii: .. 95, 3443, 3482,
    3482
\l__enumext_before_no_starred_key_v_tl 754
\l__enumext_before_no_starred_key_vii_-
    tl ..... 774
\l__enumext_before_no_starred_key_viii_-
    tl ..... 778
\l__enumext_before_starred_key_v_tl ... 750
\l__enumext_before_starred_key_vii_tl . 766
\l__enumext_before_starred_key_viii_tl 770
\__enumext_calc_hspace:NNNNNNN 71, 2356, 2356,
    2387, 2392, 2431
\l__enumext_check_ans_bool 54, 68, 69, 119, 1524,
    1528, 1576, 1795, 2093, 2229, 2259, 2637, 3122, 3230
\__enumext_check_ans_exec: .. 55, 76, 1574, 1574,
    2576, 3104
\__enumext_check_ans_int:n ..... 54
\g__enumext_check_ans_item_tl .. 65, 119, 2092,
    2100, 2104
\__enumext_check_ans_set: . 55, 1538, 1538, 1578
\__enumext_check_ans_show: 56, 1581, 1581, 2675,
    3317
\g__enumext_check_ans_show_bool 77, 119, 2640,
    2673, 2678
\g__enumext_check_ans_show_h_bool 119, 3124,
    3315, 3320
\l__enumext_columns_sep_v_dim 2745, 2747, 2755
\l__enumext_columns_sep_vii_dim .. 2913, 2915,
    2924, 2969, 3093, 3295
\l__enumext_columns_sep_viii_dim . 3326, 3328,
    3337, 3382, 3656
\l__enumext_columns_v_int 1037, 2743, 2751, 2763,
    2768
\l__enumext_columns_vii_int .. 2918, 2921, 2925,
    2933, 2937, 2940, 2946, 2952, 2956, 3080, 3290, 3301
\l__enumext_columns_viii_int . 3331, 3334, 3338,
    3346, 3350, 3353, 3359, 3365, 3369, 3651, 3662
\g__enumext_count_item_anskey_int .. 65, 119,
    1584, 1592, 1797, 2095
\g__enumext_count_item_number_int 119, 1549,
    1554, 1557, 1560, 1568, 1584, 1591, 2231, 2261, 3232
\g__enumext_count_item_with_ans_int .... 60
\l__enumext_counter_i_tl ..... 30, 313
\l__enumext_counter_ii_tl ..... 30, 314
\l__enumext_counter_iii_tl ..... 30, 315
\l__enumext_counter_iv_tl ..... 30, 316
\l__enumext_counter_style_for_ref_vii_-
    tl ..... 452, 462, 473, 475
\l__enumext_counter_style_for_ref_viii_-
    tl ..... 479, 481
\l__enumext_counter_style_for_ref_X_tl 144
\c__enumext_counter_style_tl .... 32, 144, 419
\g__enumext_counter_styles_tl . 23, 31, 40, 324,
    342
\l__enumext_counter_v_tl ..... 30, 317
\l__enumext_counter_vi_tl ..... 30, 318
\l__enumext_counter_vii_tl ..... 30, 319, 449
\l__enumext_counter_viii_tl ..... 30, 320, 459
\__enumext_current_env: 27, 212, 212, 2462, 3028

\l__enumext_current_widest_dim 23, 40, 348, 493,
    513, 529, 533
\__enumext_default_item:n ... 2225, 2225, 2274
\__enumext_define_counters:Nn 23, 304, 304, 313,
    314, 315, 316, 317, 318, 319, 320
\__enumext_endminipage: . 28, 246, 249, 915, 2907,
    3275, 3636
\__enumext_fake_item: ..... 656, 656, 2418
\l__enumext_fake_item_indent_v_dim 675, 680
\l__enumext_fake_item_indent_v_tl 677, 2282,
    2286, 2294
\l__enumext_fake_item_indent_vii_dim 687, 692
\l__enumext_fake_item_indent_vii_tl 689, 3271
\l__enumext_fake_item_indent_viii_dim . 699,
    704, 3628
\l__enumext_fake_item_indent_viii_tl .. 701,
    3627, 3631
\l__enumext_fake_item_indent_X_tl ..... 70
\__enumext_fake_item_vii: .... 656, 684, 2448
\__enumext_fake_item_viii: .... 656, 696, 2453
\__enumext_filter_series:n ... 1387, 1434, 1472,
    1487, 1499
\__enumext_filter_series_key:n 1387, 1439, 1443
\__enumext_filter_series_pair:nn . 1387, 1440,
    1452
\g__enumext_footnote_arg_seq . 141, 2198, 2211,
    2221
\g__enumext_footnote_int . 141, 2205, 2208, 2210,
    2212
\g__enumext_footnote_int_seq . 141, 2199, 2212,
    2217, 2220
\__enumext_footnotes_key_bool ..... 29
\l__enumext_footnotes_key_bool 25, 29, 90, 131,
    262, 267, 276, 3238, 3286, 3601, 3647
\__enumext_footnotetext:nn ... 2192, 2192, 2222
\__enumext_getkeyans:nn ... 99, 3686, 3690, 3690
\__enumext_getkeyans_aux:n . 99, 3674, 3677, 3677
\l__enumext_hyperref_bool 25, 29, 131, 258, 279,
    296, 1861, 2080, 3225, 3592
\__enumext_hypertarget:nn 29, 253, 281, 285, 301
\__enumext_if_is_int:n ..... 192
\__enumext_if_is_int:nTF ..... 192, 540, 554
\l__enumext_item_column_pos_vii_int 89, 2940,
    2946, 2952, 2956, 2963, 3153, 3290, 3293
\l__enumext_item_column_pos_viii_int ... 95,
    3353, 3359, 3365, 3369, 3376, 3497, 3651, 3654
\l__enumext_item_column_pos_X_int ..... 155
\g__enumext_item_count_all_vii_int 89, 2964,
    3154, 3301, 3308
\g__enumext_item_count_all_viii_int 95, 3377,
    3498, 3662, 3669
\g__enumext_item_count_all_X_int ..... 155
\__enumext_item_peek_args_vii: 89, 3155, 3157,
    3157
\__enumext_item_peek_args_viii: 95, 3499, 3501,
    3501
\__enumext_item_starred: .. 70, 2316, 2316, 2334
\l__enumext_item_starred_vii_bool 3172, 3188,
    3242
\l__enumext_item_starred_viii_bool 3516, 3532,
    3605, 3625
\l__enumext_item_starred_X_bool ..... 155
\__enumext_item_std:w 28, 68-70, 83, 240, 244, 2234,
    2240, 2264, 2282, 2286, 2294, 2888

```

[\g__enumext_item_symbol_aux_vii_tl](#) 3196, 3244, 3247, 3251, 3253
[\g__enumext_item_symbol_aux_X_tl](#) 155
[\l__enumext_item_symbol_sep_vii_dim](#) . . 3205, 3213, 3250, 3252
[\g__enumext_item_symbol_tl](#) 23, 69, 35, 2249, 2322, 2339
[\l__enumext_item_symbol_vii_tl](#) 3247
[\l__enumext_item_text_vii_box](#) 3237, 3278, 3285
[\l__enumext_item_text_viii_box](#) 3600, 3639, 3646
[\l__enumext_item_text_X_box](#) 155
[\l__enumext_item_width_vii_dim](#) . . 2922, 2967, 2975, 2976
[\l__enumext_item_width_viii_dim](#) . . 3335, 3380, 3388, 3389
[\l__enumext_item_width_X_dim](#) 155
[\l__enumext_itemindent_X_dim](#) 44
[\l__enumext_itemsep_vii_skip](#) 3307
[\l__enumext_itemsep_viii_skip](#) 3668
[\l__enumext_joined_item_aux_vii_int](#) . . 2961, 2962, 2963, 2964, 2970
[\l__enumext_joined_item_aux_viii_int](#) . 3374, 3375, 3376, 3377, 3383
[\l__enumext_joined_item_aux_X_int](#) . . . 155
[__enumext_joined_item_vii:w](#) . . 89, 3160, 3161, 3163, 3163
[\l__enumext_joined_item_vii_int](#) . . 2932, 2933, 2936, 2938, 2944, 2949, 2954, 2959, 2961, 2967
[__enumext_joined_item_viii:w](#) 95, 96, 3504, 3505, 3507, 3507
[\l__enumext_joined_item_viii_int](#) . 3345, 3346, 3349, 3351, 3357, 3362, 3367, 3372, 3374, 3380
[\l__enumext_joined_item_X_int](#) 155
[\l__enumext_joined_width_vii_dim](#) . 2965, 2972, 2975, 3268, 3280
[\l__enumext_joined_width_viii_dim](#) 3378, 3385, 3388, 3622, 3641
[\l__enumext_joined_width_X_dim](#) 155
[__enumext_keyans_addto_prop:n](#) 63, 1982, 1982, 2296, 2820
[__enumext_keyans_addto_seq:n](#) . 65, 2056, 2056, 2298, 2822
[__enumext_keyans_addto_seq_link:](#) 2056, 2074, 2076, 3563
[__enumext_keyans_anspic_code:nnn](#) . 82, 2811, 2814, 2814
[__enumext_keyans_check_ans:nn](#) . . 65, 66, 2098, 2098, 2696, 2852, 3457
[__enumext_keyans_default_item:n](#) . . 69, 2277, 2277, 2312
[\l__enumext_keyans_env_bool](#) 20, 2541, 2554, 2707, 2791
[__enumext_keyans_fake_item:](#) . . 656, 672, 2408
[\l__enumext_keyans_item_opt_tl](#) 82, 2110, 2123, 2129, 3548
[\l__enumext_keyans_level_h_int](#) 20, 2034, 3463, 3464
[\l__enumext_keyans_level_int](#) . . 20, 1190, 1784, 2029, 2706, 2710, 2805
[__enumext_keyans_make_label:](#) 31, 71, 2342, 2342, 2407
[__enumext_keyans_mini_addvspace:](#) 46, 79, 1094, 1094, 2735
[__enumext_keyans_mini_right_cmd:n](#) 49, 1192, 1215, 1215
[__enumext_keyans_mini_set_vskip:](#) . 45, 1032, 1032, 1096
[__enumext_keyans_multi_addvspace:](#) . 80, 881, 892, 2760
[__enumext_keyans_multi_set_vskip:](#) . 42, 881, 881, 894
[__enumext_keyans_multicols_start:](#) 80, 2739, 2741, 2741
[__enumext_keyans_multicols_stop:](#) . 80, 1219, 2766, 2766, 2790
[__enumext_keyans_parse_keys:n](#) 2685, 2719, 2719
[\l__enumext_keyans_pic_above_int](#) . 114, 2898, 2899, 2901
[\l__enumext_keyans_pic_above_skip](#) . . 83, 114, 2843, 2882
[__enumext_keyans_pic_arg_two:](#) 83, 2841, 2870, 2870
[\l__enumext_keyans_pic_below_int](#) . 114, 2898, 2899, 2902
[\l__enumext_keyans_pic_body_seq](#) . . 82, 84, 114, 2809, 2848, 2906
[__enumext_keyans_pic_do:n](#) 83, 2848, 2850, 2890, 2890, 2894
[\l__enumext_keyans_pic_level_int](#) . . 20, 1182, 1788, 1985, 2024, 2059, 2142, 2859, 2860
[__enumext_keyans_pic_row:n](#) . 83, 84, 2892, 2895, 2895
[__enumext_keyans_pic_safe_exec:](#) . . 83, 2837, 2857, 2857
[__enumext_keyans_pic_skip_abs:N](#) . . 83, 2865, 2865, 2881
[\l__enumext_keyans_pic_width_dim](#) . 114, 2897, 2904
[__enumext_keyans_redefine_item:](#) . . 70, 2300, 2300, 2406
[__enumext_keyans_safe_exec:](#) . 2684, 2700, 2700
[__enumext_keyans_show_ans:](#) . . 2106, 2114, 2133
[__enumext_keyans_show_item_opt:](#) . 2106, 2121, 2294, 2833, 3628
[__enumext_keyans_show_left:n](#) . 70, 2106, 2106, 2292, 2828
[__enumext_keyans_show_pos:](#) . . 2106, 2118, 2140
[__enumext_keyans_starred_item:n](#) . . 70, 2289, 2289, 2308
[__enumext_keyans_store_ref:](#) . . 64, 2003, 2003, 2297, 2821, 3561
[__enumext_keyans_store_ref_aux_i:](#) 64, 2003, 2015, 2018
[__enumext_keyans_store_ref_aux_ii:](#) 65, 2003, 2044, 2046
[\l__enumext_keyans_tmpa_dim](#) 82
[\l__enumext_keyans_tmpa_tl](#) 24, 96, 82, 2291, 2295
[\l__enumext_keyans_tmpb_tl](#) 96, 82
[__enumext_keyans_wrapper_opt:n](#) . . 1601, 2129
[\l__enumext_label_copy_i_tl](#) . . 1915, 2022, 2027, 2032, 2037
[\l__enumext_label_copy_v_tl](#) 2032
[\l__enumext_label_copy_vi_tl](#) 2027
[\l__enumext_label_copy_vii_tl](#) 1890, 1901, 1932, 2022
[\l__enumext_label_copy_viii_tl](#) 2037
[\l__enumext_label_copy_X_tl](#) 133
[\l__enumext_label_fill_left_v_tl](#) 2346
[\l__enumext_label_fill_left_X_tl](#) 70

```

\l__enumext_label_fill_right_v_tl . . . . 2353
\l__enumext_label_fill_right_X_tl . . . . 70
\l__enumext_label_font_style_v_tl 2347, 2832
\l__enumext_label_font_style_vii_tl . . 3256
\l__enumext_label_font_style_viii_tl . . 3610
\l__enumext_label_i_tl . . . . . 485
\l__enumext_label_ii_tl . . . . . 485
\l__enumext_label_iii_tl . . . . . 485
\l__enumext_label_iv_tl . . . . . 485
\__enumext_label_style:Nnn 23, 31, 337, 337, 352,
490, 510, 526, 530
\l__enumext_label_v_tl . . 63, 65, 523, 1990, 2064,
2135, 2169, 2291, 2295, 2688, 2827, 2829
\l__enumext_label_vi_tl . 63, 65, 523, 1987, 2061,
2827, 2829, 2833
\l__enumext_label_vii_tl . 505, 3183, 3208, 3215
\l__enumext_label_viii_tl 505, 3527, 3555, 3559
\l__enumext_label_width_by_box . . 40, 333, 334
\__enumext_label_width_by_box:Nn 30, 331, 331,
336, 348, 564
\l__enumext_labelsep_i_dim . . . 2137, 2173, 3566,
3581
\l__enumext_labelsep_v_dim . . . . . 2750
\l__enumext_labelsep_vii_dim . 2917, 2926, 2968,
3206, 3266, 3282
\l__enumext_labelsep_viii_dim 3330, 3339, 3381,
3620, 3643
\l__enumext_labelwidth_i_dim . 2137, 2172, 3566,
3581
\l__enumext_labelwidth_v_dim . . . . . 2750
\l__enumext_labelwidth_vii_dim . . . 2917, 2925,
2968, 3259, 3263, 3281
\l__enumext_labelwidth_viii_dim . . 3330, 3338,
3381, 3613, 3617, 3642
\l__enumext_leftmargin_tmp_v_bool . 83, 2872
\l__enumext_leftmargin_tmp_X_bool . . . . 44
\l__enumext_leftmargin_tmp_X_dim . . . . 44
\l__enumext_leftmargin_X_dim . . . . . 44
\__enumext_level: 188, 188, 428, 430, 431, 439, 441,
659, 663, 667, 734, 738, 742, 746, 830, 832, 834, 836,
869, 871, 873, 875, 879, 919, 922, 941, 950, 956, 961,
965, 976, 980, 981, 986, 1022, 1026, 1199, 1205, 1252,
1254, 1256, 1259, 1266, 1268, 1270, 1273, 1692, 1700,
1704, 1708, 1953, 1956, 1957, 2233, 2234, 2238, 2239,
2240, 2247, 2249, 2253, 2254, 2257, 2263, 2264, 2318,
2321, 2323, 2330, 2331, 2332, 2335, 2338, 2468, 2470,
2509, 2514, 2515, 2516, 2518, 2522, 2527, 2528, 2529,
2531, 2547, 2560, 2567, 2578, 2580, 2583, 2584, 2586,
2591, 2598, 2601, 2603, 2605, 2606, 2607, 2608, 2611,
2617, 2622, 2628, 2631, 2633, 2646
\l__enumext_level_h_int . 20, 220, 447, 471, 1482,
1493, 1546, 1563, 1909, 1926, 3052, 3053
\l__enumext_level_int 20, 190, 231, 842, 993, 1186,
1481, 1494, 1540, 1886, 1896, 1902, 1908, 1916, 1924,
1931, 2421, 2482, 2483, 2492, 2499, 2545, 2558, 2613,
2671, 2714, 2801, 3134, 3144, 3313, 3470
\__enumext_list_arg_two_i: . . . . . 2388
\__enumext_list_arg_two_ii: . . . . . 2388
\__enumext_list_arg_two_iii: . . . . . 2388
\__enumext_list_arg_two_iv: . . . . . 2388
\__enumext_list_arg_two_v: . 70, 2388, 2690, 2873
\__enumext_list_arg_two_vii: . . . . 2427, 3035
\__enumext_list_arg_two_viii: . . . . 2427, 3446
\l__enumext_listoffset_v_dim . . . . . 2752
\l__enumext_listparindent_vii_dim . . . . 3269
\l__enumext_listparindent_viii_dim . . . 3623
\__enumext_make_label: 31, 68, 69, 71, 2326, 2326,
2416
\l__enumext_mark_answer_sym_tl . 59, 109, 1607,
1762, 1968, 2144, 2157, 3570
\l__enumext_mark_position_str 109, 1611, 1612,
1635, 1636, 1760
\l__enumext_mark_ref_sym_tl . . 109, 1621, 1866,
2088
\__enumext_mini_addvspace: . . 45, 76, 1015, 1015,
2588
\__enumext_mini_addvspace_vii: 47, 1168, 1168,
2993
\__enumext_mini_addvspace_viii: 47, 1168, 1174,
3406
__enumext_mini_env* . . . . . 909
\__enumext_mini_right_cmd:n . 48, 49, 1194, 1196,
1196
\__enumext_mini_set_vskip: . . 43, 916, 916, 1017
\__enumext_mini_set_vskip_vii: 46, 1111, 1111,
1170
\__enumext_mini_set_vskip_viii: 46, 1111, 1133,
1176
\__enumext_minipage:w 28, 246, 248, 911, 2904, 3268,
3622
\l__enumext_minipage_active_v_bool . . 79–81,
2733, 2758, 2771, 2779
\g__enumext_minipage_active_vii_bool . . . 86,
3004, 3009, 3021
\l__enumext_minipage_active_vii_bool . 2989,
3000
\g__enumext_minipage_active_viii_bool 3417,
3422, 3434
\l__enumext_minipage_active_viii_bool 3402,
3413
\g__enumext_minipage_active_X_bool . . . 155
\l__enumext_minipage_active_X_bool . . . . 58
\g__enumext_minipage_after_skip 58, 1115, 1127,
3019, 3432
\l__enumext_minipage_after_skip 43, 44, 78, 81,
58, 932, 947, 967, 983, 998, 1004, 1010, 1024, 1034,
1043, 1046, 1058, 1076, 1087, 1103, 1135, 1148, 1162,
2655, 2788
\g__enumext_minipage_center_vii_bool . 3013,
3022
\g__enumext_minipage_center_viii_bool 3426,
3435
\g__enumext_minipage_center_X_bool . . . 155
\l__enumext_minipage_hsep_v_dim . . . 79, 2731
\l__enumext_minipage_hsep_vii_dim . . . . 2987
\l__enumext_minipage_hsep_viii_dim . . . 3400
\l__enumext_minipage_left_skip 43, 79, 58, 924,
939, 958, 973, 1020, 1030, 1035, 1041, 1050, 1067,
1079, 1099, 1109, 1113, 1118, 1122, 1136, 1140, 1154,
1172, 1178
\l__enumext_minipage_left_v_dim 79, 2729, 2737
\l__enumext_minipage_left_vii_dim 2983, 2995
\l__enumext_minipage_left_viii_dim 3396, 3408
\l__enumext_minipage_left_X_dim . . . . . 58
\g__enumext_minipage_right_skip 58, 1114, 1119,
1123, 3012, 3425
\l__enumext_minipage_right_skip . . 43, 58, 928,
943, 963, 978, 1036, 1042, 1054, 1072, 1083, 1137,
1144, 1158, 1206, 1223

```


`\l__enumext_minipage_right_v_dim` .. 79, 1217, 1222, 2727, 2731
`\g__enumext_minipage_right_vii_dim` 85, 2991, 3011, 3024
`\l__enumext_minipage_right_vii_dim` 85, 2981, 2986, 2992
`\g__enumext_minipage_right_viii_dim` .. 3404, 3424, 3437
`\l__enumext_minipage_right_viii_dim` .. 3394, 3399, 3405
`\g__enumext_minipage_right_X_dim` 155
`\g__enumext_minipage_right_X_skip` 155
`\g__enumext_minipage_stat_int` . 76, 79, 58, 1211, 1228, 2587, 2648, 2653, 2734, 2781, 2786
`\g__enumext_miniright_code_vii_tl` . 86, 3017, 3023
`\g__enumext_miniright_code_viii_tl` 3430, 3436
`\g__enumext_miniright_code_X_tl` 155
`__enumext_multi_addvspace`: ... 42, 77, 864, 864, 2619
`__enumext_multi_set_vskip`: .. 41, 828, 828, 866
`\l__enumext_multicols_above_ii_skip` ... 847
`\l__enumext_multicols_above_iii_skip` .. 853
`\l__enumext_multicols_above_iv_skip` ... 859
`\l__enumext_multicols_above_v_skip` 883, 897, 907
`\l__enumext_multicols_above_X_skip` 52
`\l__enumext_multicols_below_v_skip` 887, 901, 2773
`\l__enumext_multicols_below_X_skip` 52
`__enumext_multicols_start`: . 76, 77, 2593, 2595, 2595
`__enumext_multicols_stop`: 77, 1201, 2625, 2625, 2657
`__enumext_newlabel:nn` 25, 29, 63, 289, 289, 1942, 2050
`\l__enumext_newlabel_arg_one_tl` 25, 29, 62, 64, 133, 1865, 1935, 1943, 2039, 2051, 2086
`\l__enumext_newlabel_arg_two_tl` 25, 29, 61, 133, 1889, 1899, 1913, 1929, 1944, 2026, 2031, 2036, 2052
`__enumext_parse_keys:n` 2464, 2487, 2487
`__enumext_parse_keys_vii:n` .. 3030, 3059, 3059
`__enumext_parse_keys_viii:n` . 3442, 3475, 3475
`__enumext_parse_series_resume:n` . 1387, 1463, 2495, 3065
`__enumext_parse_store_keys:n` 74, 75, 2503, 2507, 2507
`__enumext_parse_store_keys_vii:n` . 87, 3068, 3072, 3072
`\l__enumext_parsep_i_skip` . 845, 847, 996, 1044
`\l__enumext_parsep_ii_skip` 851, 853, 1002
`\l__enumext_parsep_iii_skip` ... 857, 859, 1008
`\l__enumext_parsep_vii_skip` 3270
`\l__enumext_parsep_viii_skip` 3624
`\l__enumext_partopsep_v_skip` .. 899, 903, 1070, 1074, 1081, 1085, 1101, 1105
`\l__enumext_partopsep_viii_skip` 1146
`__enumext_phantomsection`: 29, 253, 282, 286, 302
`__enumext_print_footnote`: ... 2192, 2215, 3288, 3649
`__enumext_print_keyans_box:NN` 59, 1754, 1754, 1767, 1955, 2137, 2171, 3566, 3581
`\l__enumext_print_keyans_i_tl` 3700, 3729
`\l__enumext_print_keyans_ii_tl` ... 3705, 3730
`\l__enumext_print_keyans_iii_tl` .. 3710, 3731
`\l__enumext_print_keyans_iv_tl` ... 3715, 3732
`\l__enumext_print_keyans_vii_tl` .. 3720, 3733
`\l__enumext_print_keyans_X_tl` 98
`__enumext_printkeyans:nnn` 100, 3734, 3737, 3737
`__enumext_redefine_item`: . 69, 2266, 2266, 2415
`\l__enumext_ref_aux_tl` 144, 428, 430, 433, 449, 451, 454, 459, 461, 464
`\l__enumext_ref_key_arg_tl` .. 144, 422, 427, 434, 446, 455, 465
`__enumext_regex_label_ref_key`: .. 32, 33, 417, 417, 429, 450, 460
`__enumext_register_counter_style:Nn` .. 321, 321, 326, 327, 328, 329, 330
`__enumext_remove_extra_parsep_vii`: .. 3045, 3297, 3297
`__enumext_remove_extra_parsep_viii`: . 3456, 3658, 3658
`__enumext_renew_footnote`: ... 2192, 2196, 3240, 3603
`\l__enumext_resume_bool` 23
`__enumext_resume_counter`: ... 1363, 1391, 1399
`__enumext_resume_counter:n` 1363
`__enumext_resume_counter_series:n` 1349, 1387, 1395
`__enumext_resume_counter_series_vii:n` 1359, 1418
`__enumext_resume_counter_vii`: ... 1363, 1375, 1414, 1422
`\g__enumext_resume_int` 23, 78, 35, 1367, 1372, 1373, 2661
`__enumext_resume_only_counter`: ... 51, 1363
`__enumext_resume_only_counter_vii`: 51
`__enumext_resume_series_default:n` 1467, 1475
`__enumext_resume_starred`: 1347, 1387
`__enumext_resume_starred_vii`: ... 1357, 1410
`\g__enumext_resume_vii_int` .. 88, 35, 1379, 1384, 1385, 3113
`__enumext_safe_exec`: 2463, 2480, 2480
`__enumext_safe_exec_vii`: ... 3029, 3050, 3050
`__enumext_safe_exec_viii`: ... 3441, 3461, 3461
`\g__enumext_series_standar_default_tl` .. 35, 1389, 1392, 1486, 1487
`\g__enumext_series_starred_default_tl` .. 35, 1412, 1415, 1498, 1499
`\l__enumext_series_str` .. 1350, 1360, 1465, 1470, 1471, 2491, 3063
`__enumext_set_error:nn` 3815, 3825, 3827
`__enumext_set_label_ref:n` ... 32, 425, 425, 497
`__enumext_set_label_ref_h:n` . 33, 444, 444, 517
`__enumext_set_parse:n` 3798, 3815, 3815
`\l__enumext_setkey_tmpa_int` ... 93, 3791, 3795
`\l__enumext_setkey_tmpa_seq` 93, 3789, 3799, 3805, 3807, 3809, 3822
`\l__enumext_setkey_tmpa_tl` 93, 3797, 3801
`\l__enumext_setkey_tmpb_seq` 93, 3790, 3793, 3797, 3798
`\l__enumext_setkey_tmpb_tl` 93, 3817, 3819, 3820
`\l__enumext_show_answer_bool` . 109, 1615, 1639, 1962, 2112, 2126, 2824, 3564
`__enumext_show_length:nnn` .. 38, 198, 198, 3867, 3868, 3869, 3870, 3871, 3872, 3873, 3874, 3875, 3876, 3882, 3883, 3884, 3885, 3886, 3887, 3888, 3889, 3890, 3891

```

\l__enumext_show_position_bool 109, 1618, 1642,
    1966, 2116, 2127, 2825, 3568
\g__enumext_standar_bool . 27, 20, 219, 222, 1480,
    1566, 2677
\l__enumext_standar_bool . 20, 1894, 1907, 1923,
    2485, 2660
\g__enumext_standar_keyans_pic_star_env-
    int ..... 130
\g__enumext_standar_keyans_star_env_int 129
\g__enumext_standar_star_env_int .. 126, 223
\__enumext_standard_item_vii:w 89, 3168, 3170,
    3170
\__enumext_standard_item_viii:w 96, 3512, 3514,
    3514
\g__enumext_starred_bool 27, 87, 88, 20, 230, 233,
    1492, 1545, 1885, 1895, 1925, 2020, 2542, 2555, 2638,
    3121, 3127, 3319
\l__enumext_starred_bool . 87, 88, 20, 1818, 1826,
    1910, 1951, 3057, 3128
\__enumext_starred_columns_set_vii: .. 2911,
    2911, 3038
\__enumext_starred_columns_set_viii: . 3324,
    3324, 3449
\__enumext_starred_item:nn ... 2243, 2243, 2272
\__enumext_starred_item_exec: . 97, 3557, 3557,
    3607
\__enumext_starred_item_vii:w 89, 90, 3167, 3186,
    3186
\__enumext_starred_item_vii_aux_i:w .. 3186,
    3191, 3194
\__enumext_starred_item_vii_aux_ii:w . 3186,
    3192, 3197, 3199
\__enumext_starred_item_vii_aux_iii:w 3186,
    3202, 3211
\__enumext_starred_item_viii:w 96, 3511, 3530,
    3530
\__enumext_starred_item_viii_aux_i:w . 3530,
    3535, 3538
\__enumext_starred_item_viii_aux_ii:w 3530,
    3536, 3550, 3552
\__enumext_starred_joined_item_vii:n . 84, 89,
    2930, 2930, 3165
\__enumext_starred_joined_item_viii:n 92, 96,
    3343, 3343, 3509
\g__enumext_starred_keyans_star_env_int 128
\g__enumext_starred_star_env_int .. 127, 234
\__enumext_start_from:NNn 35, 538, 538, 551, 573
\l__enumext_start_i_int ..... 1373
\__enumext_start_item_tmp_vii: 86, 3041, 3150,
    3150
\__enumext_start_item_tmp_viii: 94, 3452, 3494,
    3494
\__enumext_start_item_vii:w . 89, 90, 3178, 3183,
    3208, 3215, 3217, 3217
\__enumext_start_item_viii:w .. 96, 3522, 3527,
    3555, 3584, 3584
\__enumext_start_list:nn 28, 73, 83, 240, 242, 2467,
    2687, 2838, 3033, 3444
\__enumext_start_mini_vii: . 88, 2979, 2979, 3106
\__enumext_start_mini_viii: 95, 3392, 3392, 3486
\__enumext_start_store_level: . 75, 2466, 2536,
    2536
\__enumext_start_store_level_vii: . 88, 3032,
    3130, 3130
\l__enumext_start_vii_int ..... 1385
\l__enumext_start_X_int ..... 70, 568
\__enumext_stop_item_tmp_vii: . 86, 89, 90, 3040,
    3044, 3152, 3219
\__enumext_stop_item_tmp_viii: .. 94, 95, 3451,
    3455, 3496, 3586
\__enumext_stop_item_vii: 90, 91, 3219, 3273, 3273
\__enumext_stop_item_viii: . 98, 3586, 3634, 3634
\__enumext_stop_list: .. 28, 240, 243, 2476, 2697,
    2851, 3046, 3458
\__enumext_stop_mini_vii: 86, 88, 2998, 2998, 3110
\__enumext_stop_mini_viii: . 95, 3392, 3411, 3490
\__enumext_stop_store_level: .. 75, 2477, 2536,
    2565
\__enumext_stop_store_level_vii: .. 88, 3047,
    3130, 3140
\l__enumext_store_active_bool 24, 54, 74, 87, 82,
    1365, 1377, 1513, 1780, 2501, 2540, 2553, 2702, 2709,
    2797, 2855, 3066, 3132, 3142, 3469
\__enumext_store_addto_prop:n 57, 63, 1673, 1674,
    1682, 1805, 2001, 3560
\__enumext_store_addto_seq:n 58, 65, 1683, 1683,
    1687, 1694, 1708, 1716, 1725, 1743, 1751, 1869, 2091
\l__enumext_store_ans_bool 119, 1514, 1527, 1690,
    1714, 1721, 1749, 1793
\l__enumext_store_anskey_arg_tl 24, 60, 61, 82,
    1811, 1820, 1822, 1828, 1836, 1839, 1849, 1854, 1857,
    1863, 1869
\__enumext_store_anskey_code:nnnn . 60, 1799,
    1803, 1803
\__enumext_store_anskey_show_left:n 63, 1810,
    1960, 1960
\__enumext_store_anskey_show_wrap:n 63, 1948,
    1948, 1964, 1979
\l__enumext_store_columns_break_bool . 1774,
    1817
\l__enumext_store_columns_join_int 82, 1825,
    1830
\l__enumext_store_columns_sep_vii_bool 3087
\l__enumext_store_columns_sep_vii_dim 3092,
    3096
\l__enumext_store_columns_sep_X_bool ... 98
l__enumext_store_columns_sep_X_dim .... 98
\l__enumext_store_columns_vii_bool ... 3074
\l__enumext_store_columns_vii_int 3079, 3083
\l__enumext_store_columns_X_bool ..... 98
\l__enumext_store_columns_X_int ..... 98
\__enumext_store_internal_ref: .. 60, 61, 1808,
    1871, 1871
\l__enumext_store_item_symbol_sep_dim 1772,
    1846, 1851
\l__enumext_store_item_symbol_tl . 1770, 1837,
    1841
\l__enumext_store_keyans_item_opt_sep-
    tl .... 1604, 1995, 1997, 2068, 2070, 3543, 3545
\l__enumext_store_keyans_item_opt_tl ... 82
\l__enumext_store_keyans_label_tl 24, 63, 65,
    82, 1984, 1987, 1990, 1997, 1999, 2001, 2058, 2061,
    2064, 2070, 2072, 2082, 2091, 2092, 3540, 3545, 3546,
    3559, 3560, 3562
\__enumext_store_level_close: . 58, 1688, 1712,
    2569
\__enumext_store_level_close_vii: 1719, 1747,
    3146

```

```

\__enumext_store_level_open: .. 57, 58, 75, 1688,
    1688, 2548, 2561
\__enumext_store_level_open_vii: .. 87, 1719,
    1719, 3136
\g__enumext_store_name_tl 24, 77, 82, 1586, 1589,
    2641, 2679, 3125, 3321
\l__enumext_store_name_tl 24, 54, 82, 1369, 1381,
    1504, 1505, 1507, 1509, 1511, 1676, 1678, 1685, 1937,
    1938, 1974, 2041, 2042, 2150, 2163, 2641, 2662, 2665,
    3114, 3117, 3125, 3576
\l__enumext_store_opt_vii_tl . 1723, 1733, 1739,
    1743, 3081, 3094
\l__enumext_store_opt_X_tl ..... 98
\l__enumext_store_ref_key_bool 60, 1624, 1806,
    1860, 2005, 2079
\l__enumext_store_upper_level_X_bool ... 98
\l__enumext_store_write_aux_file_tl 25, 63, 65,
    133, 1940, 1946, 2048, 2054
\__enumext_storing_set:n 51, 54, 1345, 1355, 1502,
    1502
\l__enumext_the_counter_vii_tl ..... 451
\l__enumext_the_counter_viii_tl ..... 461
\l__enumext_the_counter_X_tl ..... 144
\__enumext_tmp:n 30, 34, 44, 51, 52, 57, 64, 69, 70, 81,
    98, 108, 136, 140, 148, 154, 155, 174, 206, 210, 651,
    655, 1520, 1537, 1594, 1629, 1630, 1647, 1873, 1880,
    1881, 1902, 1916, 1919, 1931, 2007, 2014, 2388, 2426,
    2427, 2459
\__enumext_tmp:nn 353, 374, 375, 403, 404, 416, 568,
    587, 632, 650, 708, 716, 717, 731, 796, 812, 813, 827,
    1232, 1248, 1648, 1672, 2176, 2191
\__enumext_tmp:nnn 485, 501, 502, 503, 504, 505, 521,
    522
\__enumext_tmp:nnnnn 588, 613, 616, 619, 621, 623,
    626, 629
\__enumext_tmp:w ..... 3683, 3686
\l__enumext_tmpa_vii_int ..... 2921, 2924
\l__enumext_tmpa_viii_int ..... 3334, 3337
\l__enumext_tmpa_X_int ..... 155
\l__enumext_topsep_v_skip 885, 889, 1039, 1052,
    1060, 1065, 1085, 1089, 2854, 2885
\l__enumext_topsep_vii_skip .. 1116, 1125, 1129
\l__enumext_topsep_viii_skip . 1138, 1160, 1164
\__enumext_use_key_ref: .... 33, 437, 437, 2417
\__enumext_use_key_ref_h: .. 33, 469, 469, 2445
\l__enumext_vspace_a_star_v_bool ..... 1281
\l__enumext_vspace_a_star_vii_bool ... 1303
\l__enumext_vspace_a_star_viii_bool ... 1314
\l__enumext_vspace_a_star_X_bool ..... 70
\__enumext_vspace_above: .. 49, 1249, 1249, 2574
\__enumext_vspace_above_v: . 50, 1277, 1277, 2725
\l__enumext_vspace_above_v_skip .. 1279, 1283,
    1285
\__enumext_vspace_above_vii: .. 50, 1299, 1299,
    3103
\l__enumext_vspace_above_vii_skip 1301, 1305,
    1307
\__enumext_vspace_above_viii: . 50, 1299, 1310,
    3484
\l__enumext_vspace_above_viii_skip 1312, 1316,
    1318
\l__enumext_vspace_b_star_v_bool ..... 1292
\l__enumext_vspace_b_star_vii_bool ... 1325
\l__enumext_vspace_b_star_viii_bool ... 1336
\l__enumext_vspace_b_star_X_bool ..... 70
\__enumext_vspace_below: .. 50, 1263, 1263, 2659
\__enumext_vspace_below_v: . 50, 1288, 1288, 2793
\l__enumext_vspace_below_v_skip .. 1290, 1294,
    1296
\__enumext_vspace_below_vii: .. 51, 1321, 1321,
    3112
\l__enumext_vspace_below_vii_skip 1323, 1327,
    1329
\__enumext_vspace_below_viii: . 51, 1321, 1332,
    3492
\l__enumext_vspace_below_viii_skip 1334, 1338,
    1340
\__enumext_widest_from:nnn .. 35, 552, 552, 567,
    579
\g__enumext_widest_label_tl 23, 31, 40, 341, 345,
    349
\l__enumext_wrap_label_opt_v_bool .... 2285
\l__enumext_wrap_label_opt_vii_bool 89, 3177
\l__enumext_wrap_label_opt_viii_bool 96, 3521
\l__enumext_wrap_label_opt_X_bool ..... 70
\l__enumext_wrap_label_v_bool 2281, 2285, 2293,
    2348
\l__enumext_wrap_label_vii_bool 89, 3176, 3181,
    3189, 3257
\l__enumext_wrap_label_viii_bool .. 96, 3520,
    3525, 3533, 3611
\l__enumext_wrap_label_X_bool ..... 70
\__enumext_wrapper_label_v:n ..... 2350, 2833
\__enumext_wrapper_label_vii:n ..... 3260
\__enumext_wrapper_label_viii:n ..... 3614
\__enumext_zero_count_level: ..... 204, 204
\__enumext_zero_parsep: ..... 44, 936, 991, 991
enumext* ..... 5, 3026
enumXi ..... 313
enumXii ..... 313
enumXiii ..... 313
enumXiv ..... 313
enumXv ..... 313
enumXvi ..... 313
enumXvii ..... 313
enumXviii ..... 313
Environments provide by enumext:
enumext* 22, 23, 25-27, 30, 32-34, 37-40, 46, 47, 50, 51,
    55-62, 64, 67, 73-75, 87, 88, 90, 92, 93, 95, 97, 99, 102,
    104
enumext 22, 23, 25, 27, 30, 31, 33-36, 38-46, 48-51, 55-60,
    62, 64, 67-75, 78, 79, 83-85, 88, 99, 102, 103
keyans* 22-24, 26, 27, 30, 32-34, 37-40, 46, 47, 50, 51, 54,
    57, 64, 67, 73, 94, 95, 102, 104
keyanspic 22-25, 30, 31, 34, 48, 54, 57, 58, 63-66, 81-83,
    103
keyans 22-25, 27, 30, 31, 34-36, 38-40, 42, 45, 46, 48-50,
    54, 57, 58, 63-66, 70-73, 78, 79, 81-83, 85, 95, 102, 103
Environments:
list ..... 27, 28, 71, 73, 74
lrbox ..... 84, 90, 91, 97, 98
minipage ..... 27, 28, 40, 42, 43, 81-84, 90, 91, 98
multicols ..... 41-43, 48, 76-78, 80, 81
exp commands:
\exp_after:wN ..... 3686
\exp_args:Ne ..... 2498, 3674
\exp_not:N 152, 344, 433, 454, 464, 665, 679, 680, 691,
    692, 703, 704, 1865, 1971, 1972, 2084, 2147, 2148,

```


2160, 2161, 3573, 3574, 3683	
\exp_not:n 433, 434, 454, 455, 464, 465, 666, 1450, 1461, 1656, 1663, 1830, 1841, 1851, 1865, 1866, 1943, 2051, 2086, 2088, 2518, 2531, 3083, 3096	
F	
\fbox 1599	
file commands:	
\file_input_stop: 3984	
first <u>717</u>	
font <u>353</u>	
\footnote 67	
\footnote 67, 2200	
\footnotemark 2210	
\footnotesize 1972, 2148, 2161, 3574	
\footnotetext 2194	
G	
\getkeyans 13, 99, <u>3672</u>	
group commands:	
\group_begin: 1792, 1970, 2146, 2159, 3236, 3255, 3572, 3599, 3609, 3694, 3728	
\group_end: 1801, 1977, 2153, 2166, 3265, 3277, 3579, 3619, 3638, 3696, 3735	
H	
\hbadness 3284, 3645	
hbox commands:	
\hbox_set:Nn 333	
\hfill 383, 387, 392, 393, 1203, 1221, 1865, 2084, 3003, 3416	
hook commands:	
\hook_gput_code:nnn 9, 182, 186, 251	
\hook_gset_rule:nnnn 252	
\hspace 3295, 3656	
\hyperlink 61, 65	
\hyperlink 1865, 2084	
\hypertarget 29	
\hypertarget 281	
I	
\IfHyperBoolean 259	
\IfPackageLoadedTF 11, 255, 269	
\ignorespaces 668	
\inputlineno 223, 234	
int commands:	
\int_add:Nn 2963, 3376	
\int_case:nn 842, 993, 1540, 1563	
\int_compare:nNnTF 447, 471, 918, 1037, 1182, 1186, 1190, 1583, 1784, 1788, 1985, 2024, 2029, 2034, 2059, 2142, 2483, 2492, 2545, 2558, 2597, 2613, 2627, 2648, 2671, 2710, 2714, 2743, 2768, 2781, 2801, 2805, 2860, 2933, 2943, 2959, 3053, 3134, 3144, 3290, 3299, 3313, 3346, 3356, 3372, 3464, 3470, 3651, 3660, 3795	
\int_compare_p:nNn 220, 231, 1481, 1482, 1493, 1494, 1546, 1886, 1896, 1908, 1909, 1924, 1926	
\int_decr:N 2962, 3375	
\int_eval:n 1678, 1938, 1972, 2042, 2148, 2161, 2403, 2444, 2951, 3364, 3574	
\int_from_alph:n 546, 560	
\int_from_roman:n 548, 562	
\int_gadd:Nn 2964, 3377	
\int_gdecr:N 1549, 1554, 1557, 1560, 1568	
\int_gincr:N 1372, 1384, 1797, 2095, 2231, 2261, 2587, 2734, 3154, 3232, 3498	
\int_gset:Nn 223, 234, 1367, 1379, 2208	
\int_gset_eq:NN 2205, 2661, 2664, 3113, 3116	
\int_gzero:N 208, 1211, 1228, 1591, 1592, 2653, 2786, 3308, 3669	
\int_if_exist:NTF 1515, 2662, 3114	
\int_incr:N 2482, 2706, 2859, 3052, 3153, 3463, 3497	
\int_mod:nn 3301, 3662	
\int_new:N 20, 21, 22, 23, 24, 35, 36, 58, 74, 86, 95, 103, 116, 117, 124, 125, 126, 127, 128, 129, 130, 141, 158, 159, 160, 161, 162, 1517	
\int_set:Nn 542, 546, 548, 1653, 1825, 2898, 2899, 2921, 2932, 2938, 2954, 3284, 3334, 3345, 3351, 3367, 3645, 3791	
\int_set_eq:NN 1373, 1385, 2513, 2961, 3078, 3374	
\int_step_function:nnN 1902, 1916, 1931	
\int_step_inline:nnn 2900, 3818	
\int_to_roman:n 190, 1882, 1920	
\int_use:N 919, 1369, 1381, 2403, 2421, 2444, 2499, 2598, 2607, 2622, 2628, 2936, 2937, 2949, 3349, 3350, 3362	
\int_zero:N 3293, 3654	
\c_one_int 2921, 2940, 2946, 2952, 2956, 2959, 3334, 3353, 3359, 3365, 3369, 3372	
\c_zero_int 220, 231, 1886, 1896, 1908, 1909, 1924, 1926, 3134, 3144, 3304, 3665	
\item 28, 39, 40, 58, 68, 81, 83, 84, 86, 94	
\item 68, 69, 89, 90, 95, 97, 244, 1696, 1702, 1727, 1735, 1822, 2061, 2064, 2268, 2302, 3039, 3041, 3450, 3452, 3562	
\item* 6, 12, <u>2300</u>	
item-pos* <u>2176</u>	
item-sym* <u>2176</u>	
\itemindent 23, 72	
\itemindent 71	
itemindent <u>632</u>	
\itemsep 82, 83	
\itemsep 2874, 2880	
\itemwidth 2928, 2972, 2976, 3341, 3385, 3389	
K	
keyans 11, <u>2682</u>	
keyans* 11, <u>3439</u>	
keyanspic 12, <u>2835</u>	
Keys for environments provide by enumext:	
above* 24, 49, 50	
above 24, 49, 50, 76, 79, 88, 95	
after 38–40, 78, 81, 88, 95	
align 24, 31, 32, 70, 90	
before* 38, 39, 76, 88, 95	
before 38, 39, 79	
below* 24, 49–51	
below 24, 49–51, 78, 81, 88, 95	
check-ans 24, 25, 27, 54, 55, 60, 65, 66, 68, 69, 76–78, 92, 103	
columns-sep* 25, 57, 75, 87	
columns-sep 40, 58, 75, 77, 80, 87	
columns* 25, 57, 75, 87	
columns 23, 40, 43, 49, 58, 75, 77, 80, 87	
first 38–40, 90	
font 31, 70, 90	
item-pos* 59, 61, 67	
item-sym* 23, 59, 61, 67, 69	
item*-sep 69	
itemindent 24, 36, 37, 70, 91	
itemsep 36, 73	
labelsep 31, 68, 72, 90	
labelwidth 30, 31, 34, 35, 72	
label 23, 30, 31, 34, 35, 84	

lisparindent	73
list-indent	23, 36, 37, 83
list-offset	36, 37
listparindent	36, 91
mark-ans	25, 56, 63
mark-pos	56, 57
mark-ref	25, 56, 61
mini-env	24, 40, 43, 48, 49, 67, 76, 79, 85, 88, 93, 95
mini-sep	24, 40, 76, 79
miniright*	24, 40
miniright	24, 40, 47, 86
minirigth*	27
minirigth	27
no-store	25, 55
noitemsep	36, 44
nosep	36, 44
parindent	73
parsep	36, 73, 91
partopsep	36
ref	26, 32, 33
resume*	51
resume	23, 51, 54, 73, 78, 88
rightmargin	36
save-ans	24, 51, 54, 57, 58, 60, 63, 65, 69, 78, 79, 81, 88, 95, 97, 99
save-key	25
save-ref	25, 29, 56, 60, 61, 64, 65, 70, 97
save-sep	56
series	51
show-ans	25, 56, 57, 59, 60, 63, 70, 96, 97
show-length	27, 38, 73, 102
show-pos	25, 56, 57, 59, 60, 63, 66, 70, 96, 97
start	24, 27, 35, 73
store-brk	59, 60
topsep	36
widest	23, 27, 35
wrap-ans	56, 59, 63
wrap-label*	31, 68, 70, 89, 90, 96
wrap-label	31, 70, 89, 90, 96
wrap-opt	56
keys commands:	
\keys_define:nn	355, 377, 406, 487, 507, 523, 570, 590, 634, 653, 710, 719, 798, 815, 1234, 1343, 1353, 1522, 1596, 1632, 1650, 1768, 2178, 3698, 3761
\l_keys_key_str	3852
\keys_set:nn	369, 822, 1239, 1244, 1392, 1404, 1415, 1427, 1814, 2494, 2498, 2721, 3064, 3479, 3763, 3764, 3765, 3766, 3767, 3768, 3769, 3770, 3771, 3772, 3773, 3774, 3812
keyval commands:	
\keyval_parse:Nnn	1438

L

label	485, 505, 523
Labels provide by enumext:	
\Alph*	30, 31
\Roman*	30, 31
\alph*	30, 31
\arabic*	30-32
\roman*	30, 31
\labelsep	83
\labelsep	2875, 2878
labelsep	353
\labelwidth	30, 83
\labelwidth	2875, 2876

labelwidth	353
\leftmargin	23, 72
\leftmargin	71, 2875
legacy commands:	
\legacy_if:nTF	3220, 3223, 3587, 3590
\legacy_if_gset_false:n	912
\legacy_if_set_false:n	3222, 3589
\legacy_if_set_true:n	3182, 3207, 3214, 3227, 3526, 3554, 3594
\linewidth	76, 79
\linewidth	2582, 2731, 2897, 2924, 2985, 3337, 3398
\list	28
\list	242
list-indent	632
list-offset	632
\listparindent	2877
listparindent	632
\lrbox	3237, 3600

M

\makebox	84
\makebox	1758, 1760, 2322, 3251, 3259, 3263, 3613, 3617
\makelabel	68, 70, 71, 84
\makelabel	70, 71, 2328, 2344
\makesavenoteenv	275
mark-ans	1594
mark-pos	1594, 1630
mark-ref	1594
mini-env	796
mini-sep	796
\minipage	28
\minipage	248
\miniright	10, 48, 1180, 2651, 2784
\miniright*	10
mode commands:	
\mode_if_vertical:TF	867, 895, 1018, 1097
\mode_leave_vertical:	665, 679, 691, 703, 1727, 1735, 1756, 2320, 3249
msg commands:	
\msg_error:nn	2712, 2716, 2803, 2862, 3055, 3466, 3472, 3775
\msg_error:nnn	1184, 1188, 1213, 1230, 1407, 1430, 3688, 3693, 3758, 3828
\msg_error:nnnn	1782, 1786, 1790, 2704, 2799, 2807
\msg_fatal:nn	2484
\msg_fatal:nnn	307
\msg_info:nnn	13, 16, 257, 271
\msg_line_context:	3856, 3861, 3866, 3881, 3904, 3908, 3912, 3917, 3922, 3927, 3932, 3936, 3941, 3946, 3950, 3955, 3959, 3964, 3969, 3974, 3978, 3982
\msg_new:nnn	3829, 3833, 3837, 3841, 3846, 3850, 3854, 3859, 3864, 3879, 3894, 3901, 3906, 3910, 3915, 3920, 3925, 3930, 3934, 3939, 3944, 3948, 3953, 3957, 3962, 3967, 3972, 3976, 3980
\msg_term:nnn	1586
\msg_term:nnnn	2411, 2421, 2450, 2455
\msg_warning:nn	2650, 2783
\msg_warning:nnn	1589
\msg_warning:nnnn	2102, 2360, 2365, 2935, 2948, 3348, 3361
\multicolsep	77, 80
\multicolsep	2612, 2756

N

\NeedsTeXFormat	3
-----------------	---

\newcounter 310

\NewDocumentCommand 1180, 1778, 2795, 3672, 3726, 3782

\NewDocumentEnvironment . 2460, 2682, 2835, 3026, 3439

\newlabel 29

\newlabel 293

no-store 1520

\noindent 86, 94

\noindent . 2589, 2736, 2994, 3040, 3292, 3407, 3451, 3653

\nointerlineskip 2589, 2736, 2994, 3407

noitemsep 588

\nopagebreak 878, 906, 1029, 1108, 1171, 1177

\normalfont 1971, 2147, 2160, 3573

nosep 588

P

Packages:

enumext 22, 51, 72, 81, 102

enumitem 30

expl3 84

footnotehyper 29

hyperref 25, 27–29, 33, 61, 65, 90, 102

lua-visual-debug 43

multicol 22, 102

shortlst 84

\par 878, 906, 1029, 1108, 1171, 1177, 1206, 1223, 1950, 2633, 2655, 2773, 2788, 2909, 3012, 3019, 3292, 3306, 3425, 3432, 3653, 3667

\parindent 3269, 3623

\parsep 41, 44, 82, 83

\parsep 1728, 1736, 2441, 2874, 2881, 2886

parsep 588

\parskip 3270, 3624

\partopsep 83

\partopsep 2442, 2879

partopsep 588

peek commands:

\peek_meaning:NFTF 3159, 3173, 3190, 3201, 3503, 3517, 3534

\peek_meaning_remove:NFTF 3166, 3510

\peek_remove_spaces:n 2306

\phantomsection 29

\phantomsection 282

prg commands:

\prg_do_nothing: 286

\prg_new_protected_conditional:Npnn ... 192

\prg_replicate:nn 201, 3899

\prg_return_false: 196

\prg_return_true: 195

\printkeyans 13, 99, 3726

prop commands:

\prop_count:N 1678, 1938, 1974, 2042, 2150, 2163, 3576

\prop_gput_if_not_in:Nnn 1673, 1676

\prop_if_exist:NFTF 1505, 3692

\prop_item:Nn 3695

\prop_new:N 1507

\ProvidesExplPackage 4

R

\raggedcolumns 2621, 2762

\ref 61, 64

ref 485, 505

\refstepcounter 3229, 3596

regex commands:

\regex_match:nnTF 194, 545, 547, 559, 561, 2511, 2524, 3076, 3089

\regex_replace_once:nnN 421

\renewcommand 433, 454, 464

\RenewDocumentCommand ... 2200, 2268, 2302, 2328, 2344

\RequirePackage 17

resume 1343

resume* 1343

rightmargin 632

\Roman 31, 35

\Roman 329

\roman 31, 35

\roman 330, 503, 3714

S

save-ans 1343

save-ref 1594

save-sep 1594

scan commands:

\scan_stop: 83, 2888, 3039, 3450, 3683, 3686

seq commands:

\seq_clear:N 3789

\seq_const_from_clist:Nn 3777

\seq_count:N 2848, 3793

\seq_gclear:N 2198, 2199

\seq_gput_right:Nn 1685, 2211, 2212

\seq_if_empty:NFTF 2217, 3741, 3807

\seq_if_exist:NFTF 1509, 3739

\seq_item:Nn 2906

\seq_map_function:NN 3798

\seq_map_inline:Nn .. 3747, 3752, 3786, 3808, 3809

\seq_map_pairwise_function:NNN 2219

\seq_new:N 96, 97, 114, 142, 143, 1511

\seq_pop_left:NN 3797

\seq_put_right:Nn 2809, 3805, 3822

\seq_set_from_clist:Nn 3790

\seq_set_map_e:NNn 3799

\seq_show:N 3743

series 1343

\setcounter 556, 560, 562, 2403, 2444, 2853

\setenumext . 6–9, 100, 3702, 3707, 3712, 3717, 3722, 3782

\setlength 1729, 1737

show-ans 1594, 1630

show-length 708

skip commands:

\skip_add:Nn . 847, 853, 859, 869, 873, 897, 901, 998, 1004, 1010, 1020, 1024, 1046, 1099, 1103, 2874

\skip_eval:n 1728, 1736

\skip_gset:Nn 1119, 1123, 1127

\skip_gzero_new:N 1114, 1115

\skip_horizontal:N 680, 692, 704, 3252, 3266, 3620

\skip_horizontal:n ... 666, 1757, 1765, 2321, 2323, 3250, 3628

\skip_if_eq:nnTF 845, 851, 857, 921, 955, 996, 1002, 1008, 1039, 1044, 1065, 1116, 1138, 1251, 1265, 1279, 1290, 1301, 1312, 1323, 1334

\skip_new:N 54, 55, 59, 60, 61, 62, 63, 118, 172

\skip_set:Nn . 830, 834, 883, 887, 924, 928, 932, 939, 943, 947, 958, 963, 967, 973, 978, 983, 1041, 1042, 1043, 1050, 1054, 1058, 1067, 1072, 1076, 1079, 1083, 1087, 1118, 1122, 1140, 1144, 1148, 1154, 1158, 1162, 2868, 2882

\skip_set_eq:NN 2401, 2440, 2441, 3269, 3270, 3623, 3624

\skip_use:N 832, 836, 871, 875, 879, 899, 903, 922, 941, 950, 956, 961, 965, 976, 980, 981, 986, 1022, 1026, 1052, 1252, 1256, 1259, 1266, 1270, 1273, 2633

`\skip_zero:N` 2442, 2612, 2756, 2879, 2880
`\skip_zero_new:N` 1034, 1035, 1036, 1113, 1135, 1136, 1137
`\c_zero_skip` 845, 851, 857, 922, 956, 996, 1002, 1008, 1039, 1044, 1065, 1116, 1138, 1252, 1266, 1279, 1290, 1301, 1312, 1323, 1334
`\small` 3704, 3709, 3714, 3719, 3724
`\star` 2182
`start` 568
`\stepcounter` 2204, 2816
str commands:
`\c_backslash_str` 3908, 3917, 3918, 3922, 3923, 3927, 3928, 3959, 3960, 3964, 3969, 3970
`\c_colon_str` 1937, 2041, 3683
`\str_case:nn` 214
`\str_case:nnTF` 1445, 1454
`\str_clear:N` 2491, 3063
`\str_count:n` 201, 3899
`\str_if_empty:N` 1465
`\str_if_eq:nnTF` 2404, 2446
`\str_if_in:nnTF` 3679
`\str_new:N` 113, 167
`\str_set:Nn` . . . 409, 410, 411, 1611, 1612, 1635, 1636
`\string` 275
`\strutbox` . 926, 930, 934, 945, 949, 960, 969, 975, 985, 998, 1004, 1010, 1041, 1042, 1043, 1046, 1056, 1060, 1069, 1076, 1081, 1089, 1118, 1119, 1122, 1129, 1142, 1150, 1156, 1164, 2884

T

TeX and $\TeX_{2\epsilon}$ commands:
`\@auxout` 291
`\@currentenv` 214
`\protected@write` 291
text commands:
`\text_expand:n` 3675
`\textasteriskcentered` 1608, 1622
`\thepage` 297
tl commands:
`\c_space_tl` 2129, 3866, 3881
`\tl_clear:N` 382, 388, 1811, 1984, 2058, 3540
`\tl_clear_new:N` 339
`\tl_const:Nn` 144, 323
`\tl_gclear:N` 1486, 1498, 2104, 2339, 2679, 3023, 3253, 3321, 3436
`\tl_gclear_new:N` 1470
`\tl_gput_right:Nn` 324
`\tl_greplace_all:Nnn` 345
`\tl_gset:Nn` 1471, 1487, 1499, 2092, 2641, 3125, 3196
`\tl_gset_eq:NN` 341, 2249, 3246
`\tl_if_blank:nTF` 3244
`\tl_if_empty:N` . 439, 473, 479, 1389, 1412, 1692, 1723, 1837, 1995, 2068, 2100, 2123, 2318, 3543, 3820
`\tl_if_empty:nTF` 1397, 1420
`\tl_if_exist:N` 1402, 1425
`\tl_if_novalue:nTF` . 1812, 1823, 1992, 2066, 2108, 2202, 2227, 2245, 2250, 2279, 2489, 2846, 3061, 3477,

3541, 3784
`\tl_map_inline:Nn` 342, 419
`\tl_new:N` 32, 37, 38, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 90, 91, 93, 94, 100, 101, 111, 112, 123, 133, 134, 135, 138, 146, 147, 150, 151, 166, 169
`\tl_put_left:Nn` 1700, 1733, 1820, 2135, 2169, 3559, 3562
`\tl_put_right:Nn` 340, 431, 452, 462, 1654, 1661, 1704, 1739, 1822, 1828, 1836, 1839, 1849, 1854, 1857, 1863, 1889, 1899, 1913, 1929, 1935, 1940, 1987, 1990, 1997, 1999, 2026, 2031, 2036, 2039, 2048, 2061, 2064, 2070, 2072, 2082, 2516, 2529, 3081, 3094, 3545, 3546, 3700, 3705, 3710, 3715, 3720
`\tl_remove_all:Nn` 3819
`\tl_remove_once:Nn` 1877, 2011
`\tl_replace_all:Nnn` 344
`\tl_reverse:N` 1876, 1878, 2010, 2012
`\tl_set:Nn` 152, 309, 383, 387, 392, 393, 427, 446, 663, 677, 689, 701, 1504, 1968, 2110, 2144, 2157, 2247, 3548, 3570, 3817
`\tl_set_eq:NN` 350, 428, 430, 449, 451, 459, 461, 1875, 2009, 2022, 2291, 2295, 2827, 2829
`\tl_to_str:n` 1402, 1405, 1425, 1428, 3675
`\tl_trim_spaces:n` 340, 3805, 3817, 3823
`\tl_use:N` . 346, 349, 441, 475, 481, 734, 738, 742, 746, 750, 754, 758, 762, 766, 770, 774, 778, 782, 786, 790, 794, 1762, 1882, 1890, 1901, 1915, 1920, 1932, 2234, 2240, 2264, 2282, 2286, 2294, 2330, 2331, 2338, 2346, 2347, 2353, 2468, 2688, 2832, 3017, 3256, 3267, 3271, 3430, 3610, 3621, 3627, 3631, 3729, 3730, 3731, 3732, 3733, 3801

token commands:
`\token_to_str:N` 293
`\topsep` 1729, 1737
`topsep` 588
`\typeout` 224, 235, 261, 264, 274, 275, 1485, 1497, 1550, 1569

U

`\u` 422
use commands:
`\use:N` 202, 2335, 2470
`\use:n` 1436, 3681
`\use_none:nn` 285
`\usecounter` 2402, 2443

V

`\value` 2661, 2666, 3113, 3118
`\vspace` 913, 1256, 1259, 1270, 1273, 1283, 1285, 1294, 1296, 1305, 1307, 1316, 1318, 1327, 1329, 1338, 1340, 1728, 1736, 2843, 2854, 3307, 3668

W

`widest` 568
`wrap-ans` 1594
`wrap-label` 353
`wrap-label*` 353
`wrap-opt` 1594