

marginalia — Marginal content anywhere with automatic adjustment for Lua¹TeX¹

¹ This document describes v0.84.3, last revised 2026-08-12.

² a.j.cain (AT) gmail.com

Alan J. Cain²

Released 2026-08-12

Abstract

This Lua¹TeX package allows the placement of marginal content anywhere, without `\marginpar`'s limits, and automatically adjusts positions to prevent overlaps or content being pushed off the page, and offers key-value settings that allow fine-grained customization.

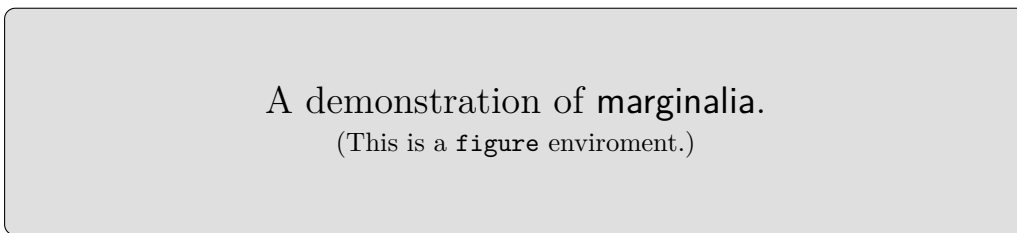
Demonstration

The `marginalia` package permits intelligent placement of marginal content, such as notes. This page demonstrates some of its capabilities.

Key-value options allow global or local settings of the style, width, placement, and spacing of marginal content. In particular, styles and widths can be specified globally depending on the margin in which the content is placed; the correct style or width will be selected for each item. For example, in this demonstration, 15 mm-wide ragged-right sans-serif has been specified globally as the default for items in the right margin and 35 mm-wide justified sans-serif for items in the left margin. Note that the vertical position of items is adjusted automatically to avoid overlaps, but there is an option to specify that items have fixed positions.

By default, this marginal note would have its top line aligned with the last line of the adjacent paragraph, but it has been automatically moved upwards to avoid clashing with the figure caption below, which has been set with an option that fixes its position.

Figure 0
`marginalia` can place content alongside floats, so it can be used to place float captions in the margin. (Here, the text style has been locally switched to Roman.)



(Like this one.)

Notes can be placed on both sides of the paragraph.

This is a top-aligned margin item pointing to the relevant line of text.

Unlike the usual `\marginpar` command, `marginalia` allows marginal content to be placed next to floats, footnotes, or the page head or footer. 'Marks' can also be added pointing to the relevant line of text and there are various vertical alignment options.

The automatic adjustment of the vertical positions of items will not result in items being closer than specified distances from the top or bottom of the page (unless there is actually insufficient space to place the items).

This marginal note, the width and style of which have locally been set to 25 mm and ragged left, would by default have its top line aligned with the last line of the adjacent paragraph, but it has been moved upwards to maintain a minimum distance of 10 mm from the bottom of the page.

This is a middle-aligned margin item pointing to the relevant line of text.

Contents

Demonstration	1
1 Introduction	4
2 Requirements	5
3 Installation	5
4 Getting started	5
5 User commands	5
5.1 Access to page and column	7
6 Options	7
6.1 Document orientation	7
6.2 Type	9
6.3 Horizontal placement	9
6.4 Vertical placement	11
6.5 Appearance	14
6.6 Warning message configuration	18
7 Placement	18
7.1 Horizontal placement	18
7.2 Vertical placement	20
8 Usage notes	21
8.1 Multiple runs	21
8.2 Reports of problems in placement	21
8.3 Right-to-left text	21
9 Incompatibilities	21
10 Limitations	22
11 Implementation (L^AT_EX package)	23
11.1 Initial set-up	23
11.2 Tagging set-up	23
11.3 Lua backend and interface	23
11.4 Auxiliary macros for setting options	25
11.4.1 General	25
11.4.2 outer/inner options	25
11.4.3 Dimension options	26
11.5 Options	27
11.5.1 Document orientation	27
11.5.2 Type	27
11.5.3 Horizontal placement	27
11.5.4 Vertical placement	29
11.5.5 Appearance	31
11.5.6 Warning message configuration	34

11.6	Processing data from the <code>.aux</code> file	35
11.7	Writing version data to the <code>.aux</code> file	36
11.8	Writing page data to the <code>.aux</code> file	37
11.9	Marginal content item processing	38
11.9.1	Variables	38
	Variables set by <code>L^AT_EX</code> .	38
	Variables set by Lua.	38
11.9.2	Core macro	39
11.9.3	Horizontal separation, width, style, mark selection	42
11.9.4	Auxiliary box macros	43
11.9.5	Placement macros	44
11.10	User commands	45
12	Implementation (Lua backend)	46
12.1	Initial set-up	46
12.2	Message functions	46
12.3	Module variables	46
12.4	Constants	47
12.5	Keys for tables	47
12.5.1	Keys for both page and item data tables	47
12.5.2	Keys for page data tables, layout etc.	47
12.5.3	Keys for item data tables	48
12.6	Utility functions	49
12.7	Generic page/item data functions	49
12.8	Processing of page data from <code>.aux</code> file	51
12.9	Processing of item data from <code>.aux</code> file	53
12.10	Writing reports	54
12.11	Computing horizontal positions	56
12.12	Computing vertical positions	60
12.12.1	Auxiliary functions	60
12.12.2	Computing <code>optfixed</code> enabled	61
12.12.3	Computing vertical adjustment	62
12.12.4	Checking vertical adjustment	63
12.12.5	Core vertical position computation	65
12.13	Passing <code>item_data</code> back to <code>L^AT_EX</code>	68
12.14	Export public functions	68
	Index	70

1 Introduction

The $\text{\LaTeX}$ `\marginpar` command is the basic method for placing content in the margin. For purposes such as drawing attention to particular points in the text, it functions well. Its main limitation is that `\marginpar` works via the $\text{\LaTeX}$ float mechanism and so cannot be used to create marginal content next to a figure, table, or other float, or next to a footnote, or to place running heads in the margin, such as are found in the left-hand margin of this document except for the ‘implementation’ section. (Bringinghurst called this style ‘running shoulderheads’ [Bri04, p. 65], but the term may be non-standard.)

Trying to set many separate pieces of marginal content using `\marginpar` can lead to other problems. If two `\marginpars` would clash, $\text{\LaTeX}$ shifts the second item downward. But the cumulative effect can lead to `\marginpars` being shifted downward off the bottom of the page. Further, the asynchronous nature of $\text{\TeX}$ ’s page-breaking can cause: (1) a `\marginpar` to be placed in the wrong margin; (2) the topmost `\marginpar` on a page to be unnecessarily shifted downward because of a hypothetical clash that would have occurred with the previous `\marginpar`, had they been on the same page.

Packages like `mparhack`³ (Tom Sgouros & Stefan Ulrich), `marginnote`⁴ (Markus Kohm), `marginfix`⁵ (Stephen Hicks) and `marginfit`⁶ (Maurice Leclaire) were created to avoid these limitations and problems. `mparhack` only ensures that each `\marginpar` appears on the correct side of the page. `marginnote` allows marginal content to be placed anywhere, but does not adjust positions to avoid clashes. `marginfix` adjusts positions, but the unadjusted vertical positioning can be slightly off, and the package still uses floats. `marginfit` gets positions exactly right, but uses the insert mechanism and so marginal content cannot appear next to floats or footnotes.

This $\text{\LaTeX}$ package, `marginalia`, provides a `\marginalia` command that attempts to avoid these limitations. Marginal content is placed, not via floats or inserts, but by a calculated per-item horizontal shift inside an (invisible) `\rlap` or `\llap` from the position where the `\marginalia` command was issued (which is similar to the technique used by `marginnote`), plus a calculated per-item vertical shift to avoid clashes with other content. The vertical shift is usually downward, but may be upward when necessary to prevent content from being shifted off the bottom of the page (which is similar to the vertical shifts performed by `marginfix` and `marginfit`).

The calculation of the horizontal and vertical shifts uses information written to the `.aux` file during the previous $\text{\LaTeX}$ run. It thus takes at least two runs for all content to appear in the correct places. The package reports any changes from the previous run and any problems encountered.

Note: `marginalia` was written to typeset running heads in the margin, sidenote references, side-captions for floats, and small marginal figures in the author’s book *Form & Number: A History of Mathematical Beauty* [Cai24].⁷ Thus the basic functionality has been tested extensively, and it has performed correctly.

Licence. `marginalia` is released under the $\text{\LaTeX}$ Project Public Licence v1.3c or later.⁸

Acknowledgements. The author thanks Ulrike Fischer for explaining how to add tagging support, and Julien Labbé and François Pantigny for some valuable suggestions.

Translation. A French translation of the documentation has been made by members of GUTenberg (Le Groupe francophone des Utilisateurs de $\text{\TeX}$).⁹

3 URL: <https://ctan.org/pkg/mparhack>

4 URL: <https://ctan.org/pkg/marginnote>

5 URL: <https://ctan.org/pkg/marginfix>

6 URL: <https://ctan.org/pkg/marginfit>

7 *Form & Number* is freely available on the Internet Archive under a Creative Commons licence.

URL: https://archive.org/details/cain_formand_number_ebook_large

8 URL: <https://www.latex-project.org/lppl.txt>

9 URL: <https://gitlab.gutenberg-asso.fr/gutenberg/traduction-de-marginalia/>

User commands

Feature requests and bug reports The development code and issue tracker are hosted at Codeberg.¹⁰

¹⁰ URL: <https://codeberg.org/ajcain/marginalia>

Other resources. An introduction to `marginalia` has appeared in *TUGboat* [Cai25].

2 Requirements

`marginalia` requires

- (1) Lua $\text{\LaTeX}$,
- (2) a recent $\text{\LaTeX}$ kernel with `expl3` support (any kernel version since 2020-02-02 should suffice).

It does not depend on any other packages.

3 Installation

To install `marginalia` manually, run `luatex marginalia.ins` and copy `marginalia.sty` and `marginalia.lua` to somewhere Lua $\text{\LaTeX}$ can find them.

4 Getting started

`marginalia` works ‘out of the box’. Load the package (there are no package options) and use the main `\marginalia` command to place marginal content. Figure 4.1 shows the source code for a small demonstration and the resulting document. *The source code must be processed twice by Lua $\text{\LaTeX}$ for the marginal content to be placed correctly.* (See Subsection 8.1 for discussion of the need for multiple runs.)

Turn to Section 5 for a detailed description of the available user commands, and Section 6 for the various options (such as `style=<code>`) that can be used to change the placement and formatting of the marginal content.

5 User commands

<code>\marginalia</code>	<code>\marginalia[<options>]{<content>}</code>
--------------------------	--

This is the basic command for placing marginal content. The `<content>` can, roughly speaking, be anything: text, mathematics, included graphics, TikZ. The optional argument `<options>` is a key–value list that specifies how the content is typeset. The keys are described in Section 6.

<code>\marginaliasetup</code>	<code>\marginaliasetup{<options>}</code>
-------------------------------	--

This command is used to set options for subsequent calls to `\marginalia`. The argument `<options>` is the same kind of key–value list as the `<options>` argument for the `\marginalia` command, and the keys are described in Subsection 6.

Note that `\marginaliasetup` can be used in the preamble or in the body of the document. Options set using `\marginaliasetup` have effect only within the current group.

User commands

```
\documentclass[11pt,a4paper]{article}

\usepackage{marginalia}

\begin{document}

Here is some body text.\marginalia{Here is a marginal note.} Some more
body text.\marginalia[style=\footnotesize\itshape\raggedright]{Here is another
    marginal note, set in smaller text and italics, whose position has been been
    adjusted automatically.}

\vspace{20mm}

Some final body text after a space.\marginalia[pos=left, valign=b,
style=\sffamily\raggedleft, width=35mm]{This note is placed on the left side
    of the page, wider, in sans serif, ragged left, and bottom-aligned.}

\end{document}
```

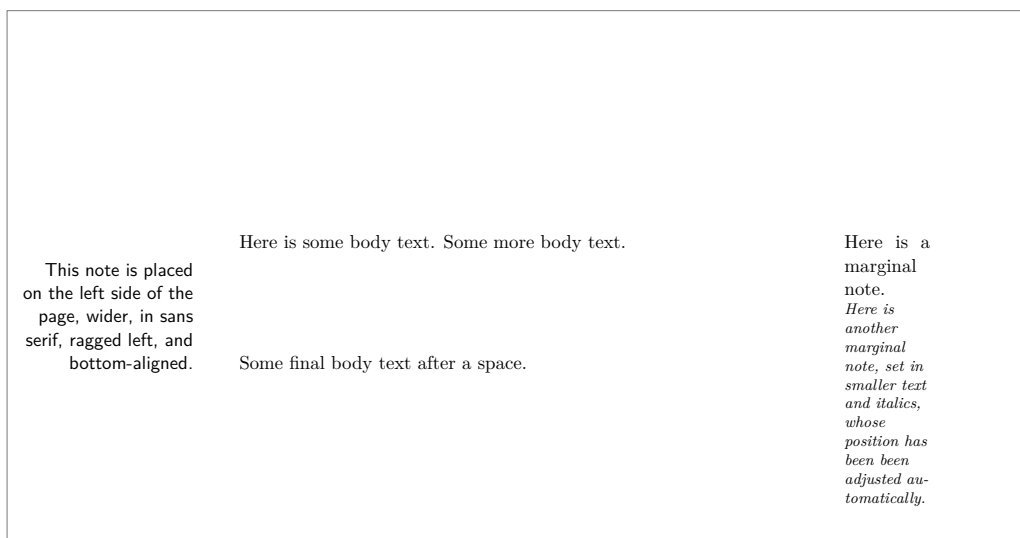


Figure 4.1: A small demonstration of marginalia.

\marginalianewgeometry \marginalianewgeometry

¹¹ URL: <https://ctan.org/pkg/geometry>

This command signals to `marginalia` that the page layout has been changed, for instance by using the `\newgeometry` command from the `geometry` package,¹¹ or by using the `\twocolumn` command to switch to two-column mode. It should be issued immediately after such a change, and certainly before the first page with the new layout has been shipped out. There is no harm in using it unnecessarily.

5.1 Access to page and column

Within the `<content>` of `\marginalia`, two counters are available which specify the actual page and column in which the call to `\marginalia` appears. These counters can be used to select different actions depending on the page on which the content appears or (in two-column mode) whether it pertains to the left or right column. It is best to use the variants of the `style` and `width` keys if marginal content should have different widths or styles depending on whether they appear on a recto/verso page or pertain to a particular column. These counters are made available for purposes not covered by the `style` and `width` variants. The value of each counter is based on the position of the call to `\marginalia` on the previous Lua $\text{\LaTeX}$ run.

`\marginaliapage` A counter register, available within the `<content>` of `\marginalia`, that holds the actual page on which the marginal content appears. The value is based on the previous Lua $\text{\LaTeX}$ run and will default to 1.

`\marginaliacolumn` A counter register, available within the `<content>` of `\marginalia`, that holds the actual column to which the marginal content pertains. The value is 1 for the left column, 2 for the right column. In one-column mode, the value is always 0. (If the key `column` is used to manually specify the column to which the content pertains, the value of `\marginaliacolumn` will change accordingly.) The value is based on the previous Lua $\text{\LaTeX}$ run and will default to 0.

6 Options

The description of keys in this section, which are summarized in [Tables 1](#) and [2](#) (on pages [8](#) and [15](#) respectively), should be read in conjunction with the discussion of how marginal content is placed in [Section 7](#). In particular, the variants of the keys `style`, `width`, and `mark` follow the terminology shown in [Figure 7.1](#).

Note that the initial values of the various keys that take dimensions (namely `width...`, `xsep...`, `ysep...`) are assigned at `\begin{document}`. Thus their defaults are determined by the values of `\marginparwidth`, `\marginparsep`, `\marginparpush` and the page layout at `\begin{document}`, *not* at package load time. Similarly, if the user sets these keys in the preamble, the assignment is evaluated at `\begin{document}`. For example, using `\marginaliasetup{width=.5\oddsidemargin}` in the preamble results in the `width` key being set to half the value of `\oddsidemargin` at `\begin{document}`, *not* to half the value of `\oddsidemargin` at the call to `\marginaliasetup`.

6.1 Document orientation

`binding side` The side of a recto page on which the document is bound and which is thus the ‘inner’ margin of a recto page. It can take the values `left` (the normal for books written in left-to-right languages, such as European languages) and `right` (the normal for books written in right-to-left languages, such as Arabic or Hebrew). *This option can only be set in the preamble. (Default: left)*

Options which involve the term `outer` and `inner` are affected by the value of `binding side`. If `binding side=left`, then `outer` refers to the right-hand margin of recto pages

Options

Table 1: Summary of keys affecting the type and position of the marginal content. (For keys affecting the appearance of marginal content, see [Table 2](#).) All keys can be set using `\marginaliasetup` or passed in the optional argument to `\marginalia`.

Key name	Value	Default
<code>type</code>	<code>{normal, fixed, optfixed}</code>	<code>normal</code>
<code>pos</code>	<code>{auto, reverse, left, right, nearest}</code>	<code>auto</code>
<code>column</code>	<code>{auto, one, left, right}</code>	<code>auto</code>
<code>xsep recto right</code>	Dimension	<code>\marginparsep</code>
<code>xsep recto left</code>	Dimension	<code>\marginparsep</code>
<code>xsep verso right</code>	Dimension	<code>\marginparsep</code>
<code>xsep verso left</code>	Dimension	<code>\marginparsep</code>
<code>xsep right between</code>	Dimension	<code>\marginparsep</code>
<code>xsep left between</code>	Dimension	<code>\marginparsep</code>
<code>xsep recto outer</code>	Dimension	<code>\marginparsep</code>
<code>xsep recto inner</code>	Dimension	<code>\marginparsep</code>
<code>xsep verso outer</code>	Dimension	<code>\marginparsep</code>
<code>xsep verso inner</code>	Dimension	<code>\marginparsep</code>
<code>xsep</code>	Dimension	<code>\marginparsep</code>
<code>xsep between</code>	Dimension	<code>\marginparsep</code>
<code>xsep outer</code>	Dimension	<code>\marginparsep</code>
<code>xsep inner</code>	Dimension	<code>\marginparsep</code>
<code>valign</code>	<code>{t, b, c, m}</code>	<code>t</code>
<code>yshift</code>	Dimension	<code>0pt</code>
<code>ysep</code>	Dimension	<code>\marginparpush</code>
<code>ysep above below</code>	Dimension	<code>\marginparpush</code>
<code>ysep above</code>	Dimension	<code>\marginparpush</code>
<code>ysep below</code>	Dimension	<code>\marginparpush</code>
<code>ysep page top</code>	Dimension	[Margin above textblock]
<code>ysep page bottom</code>	Dimension	[Margin below textblock]
<code>ysep page top margin</code>	[None]	—
<code>ysep page bottom margin</code>	[None]	—
<code>ysep page top bottom margin</code>	[None]	—

Options and to the left-hand margin of verso pages. If `binding side=right`, then `outer` refers to the left-hand margin of recto pages and to the right-hand margin of verso pages.

For example: if `binding side=left`, then setting `width recto outer` has the same effect as setting `width recto right`; if `binding side=right` then setting `width recto outer` has the same effect as setting `width recto left`.

6.2 Type

type The **type** of an item of marginal content can be set to one of the following three values:

normal: The vertical position of the item will be changed automatically if necessary to prevent a clash with another item of content.

fixed: The vertical position of the item will *never* be changed automatically from the position specified by `yshift`, even if there is a clash with another item. (The type **fixed** was designed for setting float captions in the margin, since a caption should not move away from the float with which it is associated.)

optfixed: The vertical position of the item will *never* be changed automatically from the position specified by `yshift`, even if there is a clash with another item. But an **optfixed** item will not appear in the document if it would clash with a **fixed** item. (The type **optfixed** was designed for setting running heads in the margin, which should not appear if they would clash with a figure caption set in the margin.)

(*Default: normal*)

6.3 Horizontal placement

pos The position in which an item of marginal content should be placed. It can be set to one of the the following seven values:

auto: Place the item in the default position as described in [Section 7](#): the outer margin in single-column mode, and on the opposite side from the other column in two-column mode.

reverse: Place the item on the opposite side of the text block (in one-column mode) or column (in two-column mode) from **auto**.

outer: The outer side of the text block or column.

inner: The inner side of the text block or column.

right: The right side of the text block or column.

left: The left side of the text block or column.

nearest: The side of the text block or column nearest to which `\marginalia` was called.

(*Default: auto*)

The values **auto**, **reverse**, **outer**, and **inner** are affected by the key **binding side**. The value **outer** means ‘the opposite side of the page from the binding’ and the value **inner** means ‘the same side of the page as the binding’.

Similarly, in one-column mode, the value **auto** places the item in the margin on the opposite side of the page from the binding. In two-column mode, **auto** places the item in the margin on the opposite side from the other column, regardless of the value of **binding side**. The value **reverse** always places the item in the on the opposite side of the text block or column from **auto**.

The values **right**, **left**, and **nearest** are *not* affected **binding side**.

Options

In two-column mode, `marginalia` tries to determine to which column an item of marginal content pertains using the position of the call to `\marginalia`. If the call is to the left of the mid-point between the columns, the item is assumed to pertain to the left column; otherwise, it is assumed to pertain to the right column. In certain situations, this might lead to undesired placement of the item. In particular, any call to `\marginalia` in a full-width float in two-column mode would be handled as if it were a call from one of the columns and might thus be set in the wrong place. Similarly, an overfull hbox or a piece of `\rlap`-ped text might carry a call to `\marginalia` from the left column text into the area of the page occupied by the right column.

The key `column` can be used to specify which column `marginalia` should place the item in. It can be set to one of four values:

auto: Automatically determine which column an item of marginal content is placed in.

one: Treat the item as being called from one-column mode.

left: Treat the item as pertaining to the left column.

right: Treat the item as pertaining to the right column.

The value of `column` has no effect in one-column mode. (*Default: auto*)

These keys specify the horizontal separation between an item of marginal content and the text block next to which it is placed. Which separation is used will depend on where the item is typeset. The terminology is as in [Figure 7.1](#).

xsep recto right: used for an item in the right margin of a recto page.
xsep recto left: used for an item in the left margin of a recto page.
xsep verso right: used for an item in the right margin of a verso page.
xsep verso left: used for an item in the left margin of a verso page.
xsep right between: used for an item set from the right column between the columns.
xsep left between: used for an item set from the left column between the columns.

These alternative keys similarly specify the horizontal separation between an item of marginal content and the text block. The **inner** keys apply to the margin where the binding is (as specified by `binding side`) and the **outer** keys apply to the opposite margin. Setting each of these keys has the same effect as setting the corresponding **right/left** key listed above. (See the documentation of `binding side` for further details of how **outer/inner** correspond to **right/left**.)

xsep recto outer: used for an item in the outer margin of a recto page.

xsep recto inner: used for an item in the inner margin of a recto page.

xsep verso outer: used for an item in the outer margin of a verso page.

xsep verso inner: used for an item in the inner margin of a verso page.

These keys are shortcuts for setting more than one of the above keys simultaneously.

xsep: a shorthand for setting all of the **xsep...** simultaneously.
xsep between: a shorthand for setting the keys **xsep right between** and **xsep left between** simultaneously to the same value.

xsep outer: a shorthand for setting the keys **xsep recto outer** and **xsep verso outer** simultaneously to the same value.

xsep inner: a shorthand for setting the keys **xsep recto inner** and **xsep verso inner** simultaneously to the same value.

(The shorthands **xsep outer** and **xsep inner** exist because page geometry is usually symmetrical between recto and verso pages as regards outer and inner margins. The shorthand **xsep between** exists because the space between columns, if used at all for

Options marginal content, will often be shared equally.) Each of these keys must be set to a valid dimension. (*Default:* value of `\marginparsep` at `\begin{document}`)

6.4 Vertical placement

valign This key specifies the vertical alignment of the marginal content item, before any **yshift** or automatic adjustment is applied. It can be set to one of the following three values:

- t:** The baseline of the marginal content item is the baseline of the topmost box in its contents. Thus, if no **yshift** is specified and there is no automatic adjustment, the baseline of the topmost box of the marginal content will be vertically aligned with the line where the call to `\marginalia` is located.
- b:** The baseline of the marginal content item is the baseline of the bottommost box in its contents. Thus, if no **yshift** is specified and there is no automatic adjustment, the baseline of the bottommost box of the marginal content will be vertically aligned with the line where the call to `\marginalia` is located.
- c:** The baseline of the marginal content item will be placed centrally between top and bottom of the contents as a whole.
- m:** The baseline of the marginal content item will be midway between the baselines of the topmost and bottommost boxes in its contents. Thus, if the marginal content comprises an *odd* number of *equally-spaced* lines, the baseline of the middle line will be vertically aligned with the line where the call to `\marginalia` is located.

These values are illustrated in [Figure 6.1](#). If **type=normal**, then the alignments described above for **t**, **b**, **m** may not hold because of automatic adjustment. (*Default:* **t**)

The option **valign=c** may be useful if the marginal content contains a picture, which would sit on a single oversized line. Conversely, the option **valign=m** may be useful if one is wants textual alignment with the body text.

yshift The key **yshift** is used to shift the default position of the marginal content item up (positive) or down (negative) from its normal position, which is to have its baseline aligned with the baseline of the callout position. It must be set to a valid dimension. Note that if **type=normal**, then the vertical position may be adjusted from that specified by **yshift**. If this is not desired, specify a different **type**. (*Default:* 0pt).

ysep These keys specify the minimum vertical separation above and below an item of marginal content (see [Figure 6.2](#)).

ysep above **ysep above:** the minimum vertical separation between an item and the one above. (*Default:* value of `\marginparpush` at `\begin{document}`)

ysep below **ysep below:** the minimum vertical separation between an item and the one below. (*Default:* value of `\marginparpush` at `\begin{document}`)

ysep page top **ysep page top:** the minimum vertical separation between an item and top of the page. (*Default:* margin above main textblock at `\begin{document}`)

ysep page bottom **ysep page bottom:** the minimum vertical separation between an item and bottom of the page. (*Default:* margin below main textblock at `\begin{document}`)

ysep above below: is a shorthand for setting both **ysep above** and **ysep below** simultaneously to the same value.

ysep: is a shorthand for setting all of these keys simultaneously to the same value. Each of these keys must be set to a valid dimension. (*Default:* value of `\marginparpush` at `\begin{document}`)

Options

```
\documentclass[11pt,a4paper]{article}

\usepackage{marginalia}
\marginaliasetup{
  ysep page top=0mm,
  style recto outer=\raggedright\footnotesize,
  style recto inner=\raggedleft\footnotesize,
  width=30mm,
}

\begin{document}

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod%
\marginalia[valign=t]{\large\ttfamily valign=t}\\The baseline of this marginal note
  is the baseline of the its top line.}%
\marginalia[valign=b,pos=reverse]{\large\ttfamily valign=b}\\The  baseline of this
  marginal note is the baseline of its bottom line.}

\vspace{30mm}

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi%
\marginalia[valign=c,pos=reverse]{\large\ttfamily valign=c}\\The baseline of this
  marginal note is midway between the top and bottom of its content.}%
\marginalia[valign=m]{\large\ttfamily valign=m}\\The baseline of this marginal note
  is midway between the baseline of its top line and the baseline of its bottom line.}

\end{document}
```

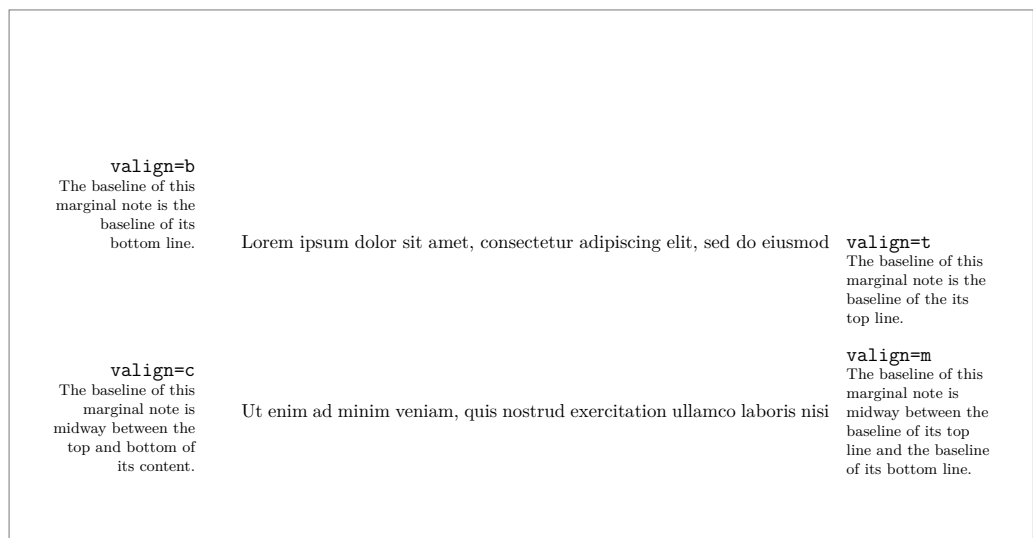


Figure 6.1: A demonstration of the various values of `valign`.

Options

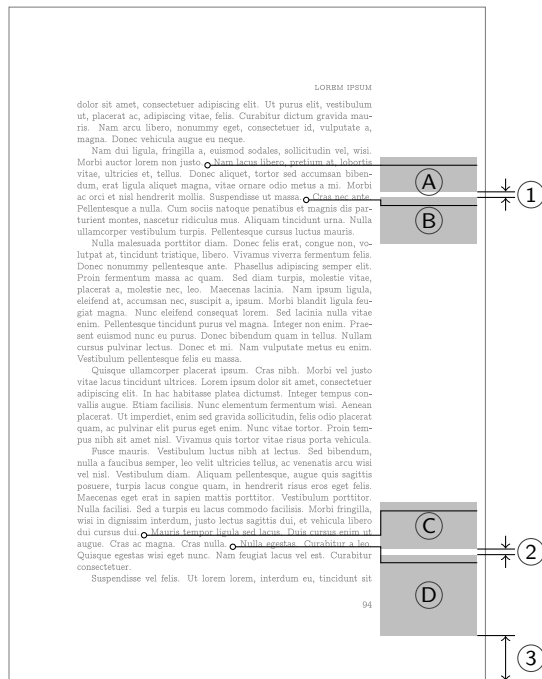


Figure 6.2: (Illustration of `ysep`) The length ① is at least the value of `ysep below` specified (locally or globally) for marginal content item ① and at least the value of `ysep above` specified for item ②. In this example diagram, ② has been automatically moved down from its natural position to maintain the required distance. Similarly, the length ② is at least the value of `ysep below` specified for ③ and at least the value of `ysep above` specified for ④, and the length ③ is at least the value of `ysep page bottom` specified for ④. In this example, to maintain the required distances, ③ and ④ have been automatically moved (respectively) up and down from their natural positions.

`ysep page top margin` These keys automatically set vertical separation between an item of marginal content and the top and bottom of the page to match the main textblock.

`ysep page bottom margin` and

`ysep page top bottom margin` **`ysep page top margin`:** Automatically set `ysep page top` to match the margins above the main textblock; to be precise, `ysep page top` is set to the value of $1\text{in} + \text{\voffset} + \text{\topmargin} + \text{\headheight} + \text{\headsep}$.

`ysep page bottom margin`: Automatically set `ysep page bottom` to match the margins above the main textblock; to be precise, `ysep page bottom` is set to the value of $\text{\paperheight} - (1\text{in} + \text{\voffset} + \text{\topmargin} + \text{\headheight} + \text{\headsep}) - \text{\textheight}$.

`ysep page top bottom margin`: Automatically set `ysep page top` and `ysep page bottom` to match the margins above and below the main textblock; has the same effect as specifying `ysep page top margin` and `ysep page bottom margin` separately.

None of these keys takes a value. Note that if the sizes of the top and bottom margins are changed, the values of `ysep page top` and `ysep page bottom` do not change automatically, even if these options have been used. The options can of course be used immediately after the new margins have been set.

Options 6.5 Appearance

An item of marginal content that appears in the inner margin might be narrower than one that appears in the outer margin, and an item appearing in the outer margin of a recto page might be set ragged right, while an item appearing in the outer margin of a verso page might be set ragged left. And since it is not known where an item will appear until the page is assembled, the keys in this subsection, dealing with the width and style of an item, have variants that apply depending on where the item appears on the page.

<code>width recto right</code>	These keys specify the width of the an item of marginal content (or, more precisely, the <code>\hsize</code> of the box into which the item is typeset). Which width is chosen will depend on the where the item is typeset. The terminology is as in Figure 7.1 .
<code>width recto left</code>	width recto right: used for an item in the right margin of a recto page.
<code>width verso right</code>	width recto left: used for an item in the left margin of a recto page.
<code>width verso left</code>	width verso right: used for an item in the right margin of a verso page.
<code>width right between</code>	width verso left: used for an item in the left margin of a verso page.
<code>width left between</code>	width right between: used for an item set from the right column and placed between the columns.
	width left between: used for an item set from the right column and placed between the columns.
<code>width recto outer</code>	These alternative keys similarly specify the width of an item of marginal content.
<code>width recto inner</code>	The inner keys apply to the margin where the binding is (as specified by binding side)
<code>width verso outer</code>	and the outer keys apply to the opposite margin. Setting each of these keys has
<code>width verso inner</code>	the same effect as setting the corresponding right/left key listed above. (See the documentation of binding side for further details of how outer/inner correspond to right/left .)
	width recto outer: used for an item in the outer margin of a recto page.
	width recto inner: used for an item in the inner margin of a recto page.
	width verso outer: used for an item in the outer margin of a verso page.
	width verso inner: used for an item in the inner margin of a verso page.
<code>width</code>	These keys are shortcuts for setting more than one of the above keys simultaneously.
<code>width between</code>	width: a shorthand for setting all of these keys simultaneously.
<code>width outer</code>	width between: a shorthand for setting the keys width right between and width left between simultaneously to the same value.
<code>width inner</code>	width outer: a shorthand for setting the keys width recto outer and width verso outer simultaneously to the same value.
	width inner: a shorthand for setting the keys width recto inner and width verso inner simultaneously to the same value.
	(The shorthands width outer and width inner exist because page geometry is usually symmetrical between recto and verso pages as regards outer and inner margins. The shorthand width between exists because the space between columns, if used at all for marginal content, will often be shared equally.) Each of these keys must be set to a valid dimension. (<i>Default:</i> value of <code>\marginparwidth</code> at <code>\begin{document}</code> })
<code>style recto outer</code>	These keys specify the style with which an item of marginal content is typeset.
<code>style recto inner</code>	Which style is chosen will depend on where the item is typeset. The terminology is as in
<code>style verso outer</code>	Figure 7.1 .
<code>style verso inner</code>	style recto right: used for an item in the outer margin of a recto page.
<code>style right between</code>	style recto left: used for an item in the inner margin of a recto page.
<code>style left between</code>	style verso right: used for an item in the outer margin of a verso page.
	style verso left: used for an item in the inner margin of a verso page.

Options

Table 2: Summary of keys affecting the appearance of the marginal content. (For keys affecting the type or position of marginal content, see [Table 1](#).) All keys can be set using `\marginaliasetup` or passed in the optional argument to `\marginalia`.

Key name	Value	Default
<code>width recto right</code>	Dimension	<code>\marginparwidth</code>
<code>width recto left</code>	Dimension	<code>\marginparwidth</code>
<code>width verso right</code>	Dimension	<code>\marginparwidth</code>
<code>width verso left</code>	Dimension	<code>\marginparwidth</code>
<code>width right between</code>	Dimension	<code>\marginparwidth</code>
<code>width left between</code>	Dimension	<code>\marginparwidth</code>
<code>width recto outer</code>	Dimension	<code>\marginparwidth</code>
<code>width recto inner</code>	Dimension	<code>\marginparwidth</code>
<code>width verso outer</code>	Dimension	<code>\marginparwidth</code>
<code>width verso inner</code>	Dimension	<code>\marginparwidth</code>
<code>width</code>	Dimension	<code>\marginparwidth</code>
<code>width between</code>	Dimension	<code>\marginparwidth</code>
<code>width outer</code>	Dimension	<code>\marginparwidth</code>
<code>width inner</code>	Dimension	<code>\marginparwidth</code>
<code>style recto right</code>	L ^A T _E X code	[Empty]
<code>style recto left</code>	L ^A T _E X code	[Empty]
<code>style verso right</code>	L ^A T _E X code	[Empty]
<code>style verso left</code>	L ^A T _E X code	[Empty]
<code>style right between</code>	L ^A T _E X code	[Empty]
<code>style left between</code>	L ^A T _E X code	[Empty]
<code>style recto outer</code>	L ^A T _E X code	[Empty]
<code>style recto inner</code>	L ^A T _E X code	[Empty]
<code>style verso outer</code>	L ^A T _E X code	[Empty]
<code>style verso inner</code>	L ^A T _E X code	[Empty]
<code>style</code>	L ^A T _E X code	[Empty]
<code>mark recto right</code>	L ^A T _E X code	[Empty]
<code>mark recto left</code>	L ^A T _E X code	[Empty]
<code>mark verso right</code>	L ^A T _E X code	[Empty]
<code>mark verso left</code>	L ^A T _E X code	[Empty]
<code>mark right between</code>	L ^A T _E X code	[Empty]
<code>mark left between</code>	L ^A T _E X code	[Empty]
<code>mark recto outer</code>	L ^A T _E X code	[Empty]
<code>mark recto inner</code>	L ^A T _E X code	[Empty]
<code>mark verso outer</code>	L ^A T _E X code	[Empty]
<code>mark verso inner</code>	L ^A T _E X code	[Empty]
<code>mark</code>	L ^A T _E X code	[Empty]

Options

	style right between: used for an item set from the right column between the columns.
	style left between: used for an item set from the right column between the columns.
style recto outer	These alternative keys similarly specify the style with which an item of marginal content is typeset. The inner keys apply to the margin where the binding is (as specified by binding side) and the outer keys apply to the opposite margin. Setting each of these keys has the same effect as setting the corresponding right/left key listed above. (See the documentation of binding side for further details of how outer/inner correspond to right/left .)
style recto inner	
style verso outer	
style verso inner	
	style recto outer: used for an item in the outer margin of a recto page.
	style recto inner: used for an item in the inner margin of a recto page.
	style verso outer: used for an item in the outer margin of a verso page.
	style verso inner: used for an item in the inner margin of a verso page.
style	There is one relevant shorthand:
	style: is a shorthand for setting all of these keys simultaneously.
	Each of these keys should be set to L ^A T _E X code that specifies the style. (<i>Default:</i> [Empty])
mark recto right	These keys specify code to typeset something alongside the line where the call to <code>\marginalia</code> appears, on the same side on which the marginal content is placed. Roughly speaking, they can be set to any L ^A T _E X code, from a single symbol to a TikZ picture. Which code is chosen will depend on where the marginal item is typeset. If the marginal item is placed on the right of the text, the mark code will be executed in an <code>\rlap</code> flush with the right end of the line; if the marginal item is placed on the left of the text, the mark code will be executed in an <code>\llap</code> flush with the left end of the line. These could be used to place arrowheads or more complicated indicators associating a marginal item with the text; see Figure 6.3
mark recto left	
mark verso right	
mark verso left	
mark right between	The terminology for the various mark options is as in Figure 7.1 .
mark left between	
	mark recto right: used for an item in the outer margin of a recto page.
	mark recto left: used for an item in the inner margin of a recto page.
	mark verso right: used for an item in the outer margin of a verso page.
	mark verso left: used for an item in the inner margin of a verso page.
	mark right between: used for an item set from the right column between the columns.
	mark left between: used for an item set from the right column between the columns.
mark recto outer	These alternative keys similarly specify code to typeset something alongside the line where the call to <code>\marginalia</code> appears. The inner keys apply to the margin where the binding is (as specified by binding side) and the outer keys apply to the opposite margin. Setting each of these keys has the same effect as setting the corresponding right/left key listed above. (See the documentation of binding side for further details of how outer/inner correspond to right/left .)
mark recto inner	
mark verso outer	
mark verso inner	
	mark recto outer: used for an item in the outer margin of a recto page.
	mark recto inner: used for an item in the inner margin of a recto page.
	mark verso outer: used for an item in the outer margin of a verso page.
	mark verso inner: used for an item in the inner margin of a verso page.
mark	There is one relevant shorthand:
	mark: is a shorthand for setting all of these keys simultaneously.
	Each of these keys should be set to L ^A T _E X code. (<i>Default:</i> [Empty])

Options

```

\documentclass[11pt,a4paper]{article}

\usepackage{marginalia}
\marginaliasetup{
  xsep=5mm, style=\raggedright\sffamily,
}

\usepackage{tikz}
\newcommand\drawarrow{%
  \tikz[remember picture,overlay]
    \draw[->,thick] (arrowstart) -- ++(-1.5mm,0) |- (.5mm,.5ex);%
}
\newcommand\savearrowstart{%
  \tikz[remember picture,overlay] \coordinate (arrowstart) at (-.5mm,.5ex);%
}

\begin{document}

Lorem ipsum dolor sit amet,%
\marginalia[mark={~$\triangleleft$}]{A marginal note. (Extra text to push down the
note below.)) consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.%
\marginalia[mark={\drawarrow}]{\savearrowstart Another marginal note.}
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur.

\end{document}

```

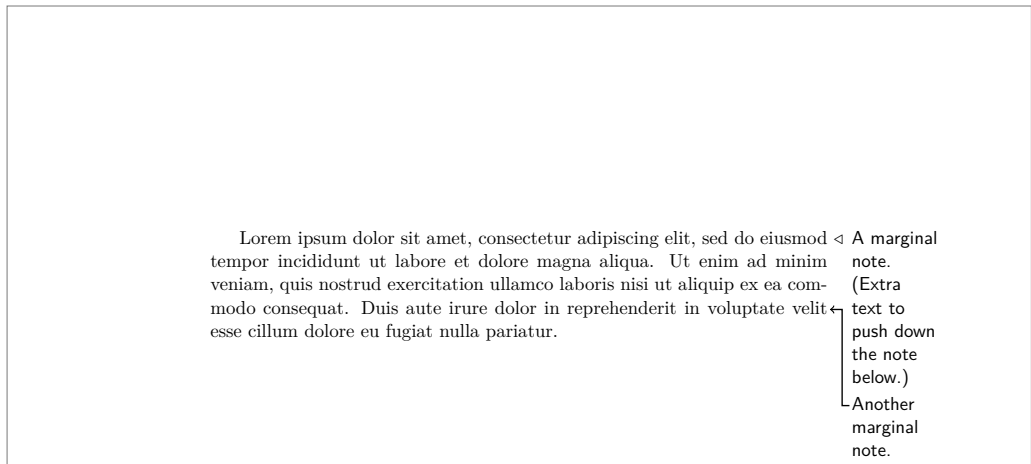


Figure 6.3: A demonstration of the `mark` option, in particular showing how it can be used to draw a TikZ arrow from a (moved) note to the relevant line. Notes: (1) The code in the marginal content is processed *before* the mark, so the `mark` TikZ code refers to a coordinate defined in the marginal content, not the other way around. (2) The code must be compiled *three* times to give this output, because the `arrowstart` coordinate is not placed correctly until after `marginalia` has placed the second marginal note on the second run. Thus one more run is necessary for TikZ to draw the arrow correctly.

Placement 6.6 Warning message configuration

The options described in this subsection have no effect on the placement of marginal content, but simply affect the number of problems or changes that are listed at `\end{document}`. Note that each of them can be set to 0 to suppress the list entirely (although warnings will still be emitted).

<code>max problem count</code>	These options specify the maximum number of placement problems, the maximum
<code>max item data change count</code>	number of changes in marginal content item data, and the maximum number of changes
<code>max page data change count</code>	in page data to list. (<i>Default: 40, 10, 10, respectively.</i>)

7 Placement

The placement of an item of marginal content depends on where the call to `\marginalia` appears in the finished document. Both horizontal and vertical placement can be complicated.

7.1 Horizontal placement

To understand the horizontal placement, first recall some terminology: a recto page is an odd-numbered page in two-sided mode, or any page in one-sided mode; a verso page is an even-numbered page in two-sided mode. The description in the paragraphs that follow is summarized in [Figure 7.1](#).

In a document where the binding is on the left (and thus on the left of recto pages), which is specified with `binding side=left`, the outer margin is on the right on recto pages and on the left on verso pages; the inner margin is on the left on recto pages and on the right on verso pages.

In a document where the binding is on the right (and thus on the right of recto pages), which is specified with `binding side=right`, the outer margin is on the left on recto pages and on the right on verso pages; the inner margin is on the right on recto pages and on the left on verso pages.

In one-column mode, marginal content is placed by default in the outer margin. If `pos=reverse` is applied, it is placed in the inner margin.

In two-column mode, the default placement is next to the column in which the call to `\marginalia` appears, on the side opposite to the other column. Thus, if the call to `\marginalia` was in the left column, the marginal content item is placed by default on the left. If `pos=reverse` is applied, it is placed between the two columns, adjacent to the left column. If the call to `\marginalia` was in the right column, the item is placed by default on the right. If `pos=reverse` is applied, it is placed between the two columns, adjacent to the right column.

`pos=outer` specifies that the item is to be placed on the outer side of the text block or column containing the call to `\marginalia`. In two-column mode, if the column is adjacent to the inner margin, its outer side lies between the columns.

`pos=right` similarly specifies that the item is to be placed on the inner side of the text block or column containing the call to `\marginalia`. In two-column mode, if the column is adjacent to the outer margin, its inner side lies between the columns.

`pos=left` specifies that the item is to be placed on the left of the text block or column containing the call to `\marginalia`.

Placement

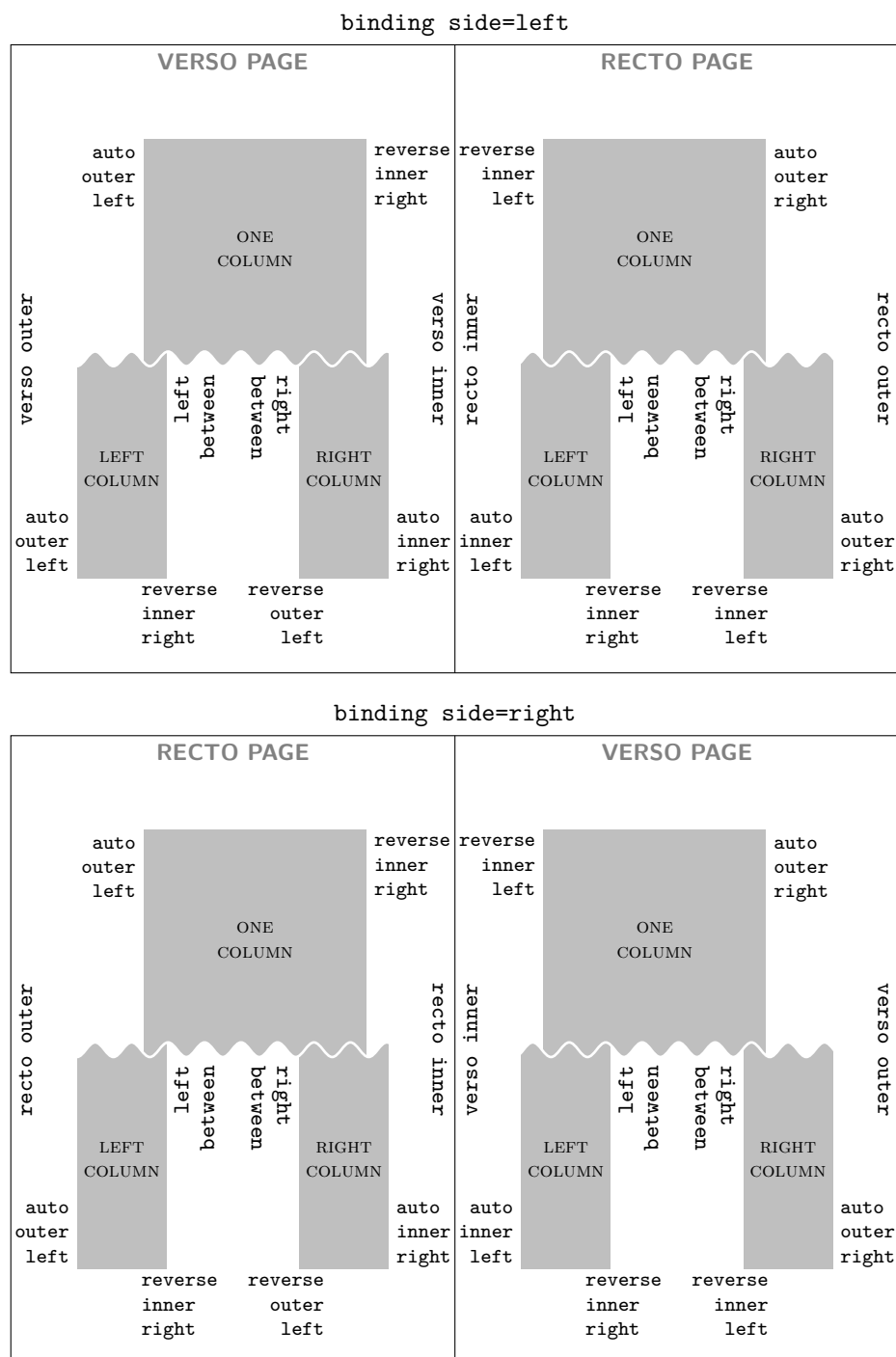


Figure 7.1: Summary of the positioning of marginal content using `pos`, and terminology used in `width` and `style` keys, on recto and verso pages, in both one-column and two-column mode. Above: when binding `side=left`. Below: when binding `side=right`.

Placement

`pos=right` similarly specifies that the item is to be placed on the right of the text block or column containing the call to `\marginalia`.

`marginalia` determines in which column the call to `\marginalia` was made using its horizontal position. As discussed in the description of the key `column`, there are situations where this can go wrong and which necessitate a manual specification of a particular column.

7.2 Vertical placement

`marginalia` tries by default to place each item of marginal content with its baseline shifted by the value of `yshift` (by default, 0pt) from the baseline where `\marginalia` was called. The actual vertical placement is calculated by the procedure described below, carried out for the items appearing in a particular horizontal location. (As shown in [Figure 7.1](#), in one-column mode the possible locations are in outer and inner margins; in two-column mode the possible locations are the outer and inner margins and on the left and right sides of the space between the columns.) A *clash* exists when two items are closer than specified by `ysep below` for the upper item or `ysep above` for the lower item, whichever is greater.

For the items in each horizontal location, the procedure is as follows:

1. Place the items appearing in a given horizontal location on the page into a list.
2. Set the vertical shift of each item to the one specified by `yshift`.
3. For each `type=optfixed` item, if it clashes with any `type=fixed` item, delete it from the list of items that appear on the page.
4. Sort the list by the position of the call to `\marginalia`, top-to-bottom, left-to-right, breaking ties by the order of calls. (Because of floats, footnotes, etc., the sorted order of the list is not necessarily the same as the order of appearance of `\marginalia` commands in the source code.)
5. Pass through the list of items in sorted order. For each `type=normal` item, if necessary shift it in a negative (downward) direction so that it (1) does not reach closer to the top of the page than specified by `ysep page top`, and (2) does not clash with the previous (above) item. (After this stage, it is possible for an assigned vertical shift to push a `type=normal` item off the bottom of the page.)
6. Pass through the list of items in the reverse of the sorted order. For each `type=normal` item, if necessary shift it in a positive (upward) direction so that it (1) does not reach closer to the bottom of the page than specified by `ysep page bottom`, and (2) does not clash with the next (below) item.

During this process, it may be found that it is impossible to prevent clashes or items reaching beyond the limits (e.g. fixed items clash with each other; a fixed item conflicts with `ysep page top` or `ysep page bottom`, or there are simply too many items of marginal content to fit (in which case, the top of some of them will be above the limit specified by `ysep page top` or will clash with fixed items)). In these cases, warnings are issued at the end of the Lua^AT_EX run.

8 Usage notes

8.1 Multiple runs

`marginalia` requires a minimum of two `LuaATEX` runs, and often more, to place items of marginal content correctly. On the first pass, information about items, including their vertical size, is written to the `.aux` file, and this information is used to position them correctly on the next run. However, because `width` and `style` have variants dependent on the margin in which the item is placed, an item may only be typeset at the correct size on this second run. Thus the vertical size of the item may have changed and so the information written to the `.aux` file on the previous run may be out of date. In this case a third run may be needed for correct placement.

More runs may be needed if the position of the call to `\marginalia` changes between runs. (For example, if `\marginalia` is used to set numbered sidenotes with per-page numbering, the number may change between runs and the small difference in widths can change line breaks and page/column breaks.) Provided that the main text stabilizes, the placement of items using `\marginalia` should be correct two runs later.

8.2 Reports of problems in placement

At the end of the `LuaATEX` run, `marginalia` reports any problems encountered in the vertical placement of items (as described at the end of [Subsection 7.2](#)). These problems are based on calculations made on the basis of information from the previous run written to the `.aux` file on the previous run, and may not arise if item positions or sizes (i.e. height or depth) have changed. `marginalia` also reports any changes in positions or sizes compared to the previous run.

In these reports, a page number refers to a visible page number if it is prefixed with ‘p’; it otherwise refers to the absolute page number of the output.

8.3 Right-to-left text

`marginalia` should place items of marginal content correctly regardless of whether `\marginalia` is called from left-to-right text or right-to-left text. The text direction of the main text will be inherited by the marginal content. The `style` key is the natural place to alter this, if one wishes to have a different text direction in the marginal content.

If the document as a whole is set in right-to-left text, then its binding is normally on the right. Using `binding side=right` changes the meanings of keys involving `inner` and `outer` to reflect this binding position and also changes the meanings of `pos=auto`, `reverse`, `outer`, `inner` appropriately.

Note: `marginalia` has not been tested extensively with right-to-left text and documents. Feedback (positive or negative) is welcome.

9 Incompatibilities

Using `marginalia` alongside `\marginpar` or packages like `mparhak`, `marginnote`, `marginfix`, or `marginfit` should not produce any errors, but `marginalia` will ignore marginal content not created using `\marginalia`; for example, an item of marginal content created using `\marginpar` might overlap with one created using `\marginalia`.

10 Limitations

As noted in the introduction, `marginalia` was originally written to typeset a particular kind of book. It thus has several limitations. Three of these are:

Lua $\LaTeX$ only Most of the code for deciding the placement of items of marginal content is written in Lua. In principle, it could be replaced with a pure $\LaTeX$ solution.

No support for ‘moving past’ fixed items The adjustment of vertical positions will never cause a `type=normal` item to be shifted past a `type=fixed` one, even when there is space on the other side. It may be desirable to have this available as an option.

No support for nested content items Nesting might be desirable for typesetting editions of manuscripts which sometimes contain marginal glosses, and then glosses upon those glosses.

The lack of any built-in facility for producing (for example) numbered sidenotes is a conscious design choice. This is properly the concern of a command that merely uses `\marginalia` to place the notes correctly.

References

- [Bri04] R. Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, version 3.0, 2004.
- [Cai24] A. J. Cain. *Form & Number: A History of Mathematical Beauty*. Lisbon, 2024. URL: https://archive.org/details/cain_formandnumber_ebook_large.
- [Cai25] A. J. Cain ‘`marginalia` at work: Running heads, float captions, citations, and small figures in the margins’. *TUGboat*. The Communications of the $\TeX$ Users Group. 46, no.1 (2025), pp.49–53. DOI: [10.47397/tb/46-1/tb142cain-marginalia](https://doi.org/10.47397/tb/46-1/tb142cain-marginalia)

11 Implementation (L^AT_EX package)

```
1 <*package>
2 <@@=marginalia>
```

11.1 Initial set-up

Package identification/version information.

```
3 \NeedsTeXFormat{LaTeX2e}[2020-02-02]
4 \ProvidesExplPackage{marginalia}{2026-08-12}{0.84.3}
5 {Non-floating marginal content for LuaLaTeX}
```

Check that Lua_T_EX is in use.

```
6 \sys_if_engine luatex:F
7 {
8   \msg_new:nnn{marginalia}{lualatex_required}
9   {LuaLaTeX-required.~Package-loading-will-abort.}
10  \msg_critical:nn{marginalia}{lualatex_required}
11 }
```

11.2 Tagging set-up

If L^AT_EX has tagging support, set up sockets if necessary and define `\__marginalia_tagging_socket:n` to be `\UseTaggingSocket`.

```
12 \@ifundefined{UseTaggingSocket}
13 {
14   \cs_new:Npn \__marginalia_tagging_socket:n #1 {}
15 }
16 {
17   \str_if_exist:cF { l__socket_tagsupport/marginpar/begin_plug_str }
18   {
19     \socket_new:nn {tagsupport/marginpar/begin}{0}
20     \socket_new:nn {tagsupport/marginpar/end}{0}
21   }
22   \str_if_exist:cF { l__socket_tagsupport/para/restore_plug_str }
23   {
24     \socket_new:nn {tagsupport/para/restore}{0}
25   }
26   \cs_new:Npn \__marginalia_tagging_socket:n #1
27   {
28     \UseTaggingSocket{#1}
29   }
30 }
```

11.3 Lua backend and interface

Load the Lua backend.

```
31 \lua_now:n{ require('marginalia') }
```

The following 10 macros interface between L^AT_EX and Lua code. Each control sequence `\__marginalia_lua_XYZ` simply calls the corresponding Lua function `marginalia.XYZ`.

`\_marginalia_lua_store_default_page_data:` The first 8 macros do not require expansion of parameters: they either have none, or process data not containing control sequences (read from the .aux file); hence `\lua_now:n` is used.

```

\marginalia_lua_store_page_data:n
\marginalia_lua_check_page_data:n
\marginalia_lua_store_item_data:n
\marginalia_lua_check_item_data:n
\marginalia_lua_compute_items:
\marginalia_lua_write_problem_report:
\marginalia_lua_write_item_change_report:
32 \cs_new:Npn\_marginalia_lua_store_default_page_data:
33 {
34   \lua_now:n{ marginalia.store_default_page_data() }
35 }
36 \cs_new:Npn\_marginalia_lua_store_page_data:n #1
37 {
38   \lua_now:n{ marginalia.store_page_data('#1') }
39 }
40 \cs_new:Npn\_marginalia_lua_check_page_data:n #1
41 {
42   \lua_now:n{ marginalia.check_page_data('#1') }
43 }
44 \cs_new:Npn\_marginalia_lua_write_page_change_report:
45 {
46   \lua_now:n{ marginalia.write_page_change_report() }
47 }
48 \cs_new:Npn\_marginalia_lua_store_item_data:n #1
49 {
50   \lua_now:n{ marginalia.store_item_data('#1') }
51 }
52 \cs_new:Npn\_marginalia_lua_check_item_data:n #1
53 {
54   \lua_now:n{ marginalia.check_item_data('#1') }
55 }
56 \cs_new:Npn\_marginalia_lua_compute_items:
57 {
58   \lua_now:n{ marginalia.compute_items() }
59 }
60 \cs_new:Npn\_marginalia_lua_write_problem_report:
61 {
62   \lua_now:n{ marginalia.write_problem_report() }
63 }
64 \cs_new:Npn\_marginalia_lua_write_item_change_report:
65 {
66   \lua_now:n{ marginalia.write_item_change_report() }
67 }

```

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

`\_marginalia_lua_load_item_data:n` This macro will receive a control sequence parameter and so requires expansion; hence `\lua_now:e` is used.

```

68 \cs_new:Npn\_marginalia_lua_load_item_data:n #1
69 {
70   \lua_now:e{ marginalia.load_item_data('#1') }
71 }

```

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

`\_marginalia_lua_call_boolean:nN` Call the Lua function (in the `marginalia` module) named by the first parameter to the value of the `expl3boolean` given in the second parameter. Since the boolean must be converted to a string, `\lua_now:e` is used.

```

72 \cs_new:Npn \__marginalia_lua_call_boolean:nN #1#2
73 {
74   \lua_now:ef marginalia.#1(\bool_to_str:N #2) }
75 }

```

(End of definition for __marginalia_lua_call_boolean:nN.)

11.4 Auxiliary macros for setting options

11.4.1 General

`\__marginalia_setup_preamble:n` Macro used to set the configuration in the preamble. This only has effect at the outer group level: inside a group, options should be confined to that group, so adding the options that use the `begindocument` hook should have no effect. And since the `\marginalia` command cannot be used in the preamble, setting other options inside a group in the preamble is pointless.

```

76 \cs_new:Npn \__marginalia_setup_preamble:n #1
77 {
78   \int_if_zero:nT{ \currentgrouplevel }
79   {
80     \keys_set:nn{marginalia}{ #1 }
81   }
82 }

```

(End of definition for __marginalia_setup_preamble:n.)

`\__marginalia_setup_body:n` Macro used to set the configuration in the document body.

```

83 \cs_new:Npn \__marginalia_setup_body:n #1
84 {
85   \keys_set:nn{marginalia}{ #1 }
86 }

```

(End of definition for __marginalia_setup_body:n.)

`\__marginalia_setup:n` The macro `\__marginalia_setup:n` is defined to be `\__marginalia_setup_preamble:n` initially and is redefined to `\__marginalia_setup_body:n` at `begindocument/end`.

```

87 \cs_set_eq:NN\__marginalia_setup:n\__marginalia_setup_preamble:n
88 \hook_gput_code:nnn{ begindocument/end }{ marginalia/setup }
89 {
90   \cs_set_eq:NN\__marginalia_setup:n\__marginalia_setup_body:n
91 }

```

(End of definition for __marginalia_setup:n.)

11.4.2 outer/inner options

`\__marginalia_outer_inner_set_immediate:NNNn` Use immediately the setting command supplied in the first parameter to set the variables supplied as the second parameter (if `\l__marginalia_binding_left_bool` is true) or the third parameter (if `\l__marginalia_binding_left_bool` is false) to the value supplied as the fourth parameter.

```

92 \cs_set:Npn \__marginalia_outer_inner_set_immediate:NNNn #1#2#3#4
93 {
94   \bool_if:NTF\l__marginalia_binding_left_bool
95   { #1#2{#4} }

```

```

96     { #1#3{#4} }
97   }

```

(End of definition for `\__marginalia_outer_inner_set_immediate:NNNn`.)

`\__marginalia_outer_inner_set_deferred:NNNn`

At the `begindocument` hook, use the setting command supplied in the first parameter to set the variables supplied as the second parameter (if `\l__marginalia_binding_left_bool` is true) or the third parameter (if `\l__marginalia_binding_left_bool` is false) to the value supplied as the fourth parameter.

```

98 \cs_set:Npn \__marginalia_outer_inner_set_deferred:NNNn #1#2#3#4
99   {
100     \hook_gput_code:nnn { begindocument } { marginalia/outer-inner-set }
101     {
102       \__marginalia_outer_inner_set_immediate:NNNn #1#2#3{#4}
103     }
104   }

```

(End of definition for `\__marginalia_outer_inner_set_deferred:NNNn`.)

`\__marginalia_outer_inner_set:Nn`

Initially, `\__marginalia_outer_inner_set:Nn` is `\__marginalia_outer_inner_set_deferred:Nn`, but is redefined at `begindocument/end` to be `\__marginalia_outer_inner_set_immediate:Nn`.

```

105 \cs_set_eq:NN
106   \__marginalia_outer_inner_set:NNNn
107   \__marginalia_outer_inner_set_deferred:NNNn
108 \hook_gput_code:nnn { begindocument/end } { marginalia/outer-inner-set-redefinition }
109   {
110     \cs_set_eq:NN
111       \__marginalia_outer_inner_set:NNNn
112       \__marginalia_outer_inner_set_immediate:NNNn
113   }

```

(End of definition for `\__marginalia_outer_inner_set:NNNn`.)

11.4.3 Dimension options

`\__marginalia_dim_set_deferred:Nn`

Set the dimension variable passed as the first parameter to value specified in second parameter at `\begin{document}`.

```

114 \cs_new:Nn \__marginalia_dim_set_deferred:Nn
115   {
116     \hook_gput_code:nnn { begindocument } { marginalia/dim }
117     {
118       \dim_set:Nn #1 { #2 }
119     }
120   }

```

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

`\__marginalia_dim_set:Nn`

Initially, `\__marginalia_dim_set:Nn` is `\__marginalia_dim_set_deferred:Nn`, but is redefined at `begindocument/end` to be `\dim_set:Nn`.

```

121 \cs_set_eq:NN \__marginalia_dim_set:Nn \__marginalia_dim_set_deferred:Nn
122 \hook_gput_code:nnn { begindocument/end } { marginalia/dim-set-redefinition }
123   {
124     \cs_set_eq:NN \__marginalia_dim_set:Nn \dim_set:Nn
125   }

```

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

11.5 Options

Set up the key–value options and the variables in which the settings will be stored.

11.5.1 Document orientation

`\l__marginalia_binding_left_bool`

```
126 \bool_new:N\l__marginalia_binding_left_bool
127 \keys_define:nn { marginalia }
128 {
129   binding-side .choices:nn = { right,left }{
130     \int_compare:nNnTF{\l_keys_choice_int}={2}
131       { \bool_set_true:N \l__marginalia_binding_left_bool }
132       { \bool_set_false:N\l__marginalia_binding_left_bool }
133   },
134   binding-side .usage:n = preamble,
135   binding-side .initial:n = left,
136 }
```

(End of definition for \l__marginalia_binding_left_bool.)

The value of `\l__marginalia_binding_left_bool` must be passed to the Lua backend at `\begin{document}`, since the resolution of `pos=auto`, `reverse`, `outer`, `inner` takes place there.

```
137 \hook_gput_code:nnn{ begindocument }{ marginalia/options }{
138   \__marginalia_lua_call_boolean:nN
139     {set_binding_left}\l__marginalia_binding_left_bool
140 }
```

11.5.2 Type

`\l__marginalia_type_int`

A key to store the type of the marginal content item. The setting is held in an integer variable: 1 = normal, 2 = fixed, 3 = optfixed.

```
141 \int_new:N\l__marginalia_type_int
142 \keys_define:nn { marginalia }
143 {
144   type .choices:nn = {normal,fixed,optfixed}{
145     \int_set:Nn\l__marginalia_type_int{\l_keys_choice_int}
146   },
147   type .initial:n = normal,
148 }
```

(End of definition for \l__marginalia_type_int.)

11.5.3 Horizontal placement

`\l__marginalia_pos_int`

A key to store the specified position of the marginal content item. The setting is held in an integer variable: 1 = `auto`, (the outer margin in one-column mode; left margin in left column, right margin in right column in two-column mode) 2 = `reverse` (inner margin in one-column mode; between the columns in two-column mode), 3 = `outer`, 4 = `inner`, 5 = `right`, 6 = `left`, 7 = `nearest`.

```
149 \int_new:N\l__marginalia_pos_int
150 \keys_define:nn { marginalia }
151 {
```

```

152 pos .choices:nn = {auto,reverse,outer,inner,right,left,nearest}{
153   \int_set:Nn\l__marginalia_pos_int{\l_keys_choice_int}
154 },
155 pos .initial:n = auto
156 }

```

(End of definition for \l__marginalia_pos_int.)

\l__marginalia_column_int A key to force the marginal content item to be treated in one-column mode or as being set from the left or right column. The setting is held in an integer variable: `-1 = auto` (automatic), `0 = one` (one-column mode), `1 = left` (left column) `2 = right` (right column).

```

157 \int_new:N\l__marginalia_column_int
158 \keys_define:nn { marginalia }
159 {
160   column .choices:nn = {auto,one,left,right}{
161     \int_set:Nn\l__marginalia_column_int{\l_keys_choice_int-2}
162   },
163   column .initial:n = auto,
164 }

```

(End of definition for \l__marginalia_column_int.)

\l__marginalia_xsep_recto_right_dim \l__marginalia_xsep_recto_left_dim \l__marginalia_xsep_verso_left_dim \l__marginalia_xsep_verso_right_dim \l__marginalia_xsep_right_between_dim \l__marginalia_xsep_left_between_dim Dimension keys to hold the separation between the marginal content item and the main text, which can be dependent on where it appears on the page.

```

165 \dim_new:N\l__marginalia_xsep_recto_right_dim
166 \dim_new:N\l__marginalia_xsep_recto_left_dim
167 \dim_new:N\l__marginalia_xsep_verso_left_dim
168 \dim_new:N\l__marginalia_xsep_verso_right_dim
169 \dim_new:N\l__marginalia_xsep_right_between_dim
170 \dim_new:N\l__marginalia_xsep_left_between_dim
171 \keys_define:nn { marginalia }
172 {
173   xsep~recto~right .code:n
174     = \__marginalia_dim_set:Nn\l__marginalia_xsep_recto_right_dim{#1},
175   xsep~recto~left .code:n
176     = \__marginalia_dim_set:Nn\l__marginalia_xsep_recto_left_dim{#1},
177   xsep~verso~right .code:n
178     = \__marginalia_dim_set:Nn\l__marginalia_xsep_verso_right_dim{#1},
179   xsep~verso~left .code:n
180     = \__marginalia_dim_set:Nn\l__marginalia_xsep_verso_left_dim{#1},
181   xsep~right~between .code:n
182     = \__marginalia_dim_set:Nn\l__marginalia_xsep_right_between_dim{#1},
183   xsep~left~between .code:n
184     = \__marginalia_dim_set:Nn\l__marginalia_xsep_left_between_dim{#1},
185   xsep .meta:n = {
186     xsep~recto~right=#1,
187     xsep~recto~left=#1,
188     xsep~verso~right=#1,
189     xsep~verso~left=#1,
190     xsep~right~between=#1,
191     xsep~left~between=#1,
192   },
193   xsep .initial:n = \marginparsep,
194 }

```

Set up outer/inner keys, which will set the appropriate left/right keys.

```

195 \keys_define:nn { marginalia }
196 {
197   xsep~recto~outer .code:n = {
198     \__marginalia_outer_inner_set:NNNn
199     \dim_set:Nn
200     \l__marginalia_xsep_recto_right_dim
201     \l__marginalia_xsep_recto_left_dim
202     {#1}
203   },
204   xsep~recto~inner .code:n = {
205     \__marginalia_outer_inner_set:NNNn
206     \dim_set:Nn
207     \l__marginalia_xsep_recto_left_dim
208     \l__marginalia_xsep_recto_right_dim
209     {#1}
210   },
211   xsep~verso~outer .code:n = {
212     \__marginalia_outer_inner_set:NNNn
213     \dim_set:Nn
214     \l__marginalia_xsep_verso_left_dim
215     \l__marginalia_xsep_verso_right_dim
216     {#1}
217   },
218   xsep~verso~inner .code:n = {
219     \__marginalia_outer_inner_set:NNNn
220     \dim_set:Nn
221     \l__marginalia_xsep_verso_right_dim
222     \l__marginalia_xsep_verso_left_dim
223     {#1}
224   },
225   xsep~outer .meta:n = {
226     xsep~recto~outer=#1,
227     xsep~verso~outer=#1,
228   },
229   xsep~inner .meta:n = {
230     xsep~recto~inner=#1,
231     xsep~verso~inner=#1,
232   },
233 }

```

(End of definition for \l__marginalia_xsep_recto_right_dim and others.)

11.5.4 Vertical placement

`\l__marginalia_valign_int` A key to store the vertical alignment of the marginal content item. The setting is held in a integer variable: 1 = t (aligned at the baseline of the topmost line of the item), 2 = b (aligned at the baseline of the bottommost line of the item).

```

234 \int_new:N\l__marginalia_valign_int
235 \keys_define:nn { marginalia }
236 {
237   valign .choices:nn = {t,b,c,m}{
238     \int_set_eq:NN\l__marginalia_valign_int\l_keys_choice_int
239   },

```

```

240   valign .initial:n = t,
241 }

```

(End of definition for \l__marginalia_valign_int.)

\l__marginalia_default_yshift_dim Dimension key to hold the default vertical shift of the marginal content item from its natural position.

```

242 \keys_define:nn { marginalia }
243 {
244   yshift .dim_set:N = \l__marginalia_default_yshift_dim,
245   yshift .initial:n = 0pt,
246 }

```

(End of definition for \l__marginalia_default_yshift_dim.)

__marginalia_margin_top: These macros are simply the calculations necessary for the space above and below the main textblock. They are simply a convenience to avoid specifying the calculation twice in the definition of the ysep keys.

```

247 \cs_new:Npn \__marginalia_margin_top:
248 {
249   1in + \voffset + \topmargin + \headheight + \headsep
250 }
251 \cs_new:Npn \__marginalia_margin_bottom:
252 {
253   \pageheight - 1in - \voffset - \topmargin - \headheight - \headsep
254   - \textheight
255 }

```

(End of definition for __marginalia_margin_top: and __marginalia_margin_bottom:.)

\l__marginalia_ysep_above_dim Dimension keys to hold the the minimum vertical spacing between a marginal content item and (respectively) the item above, the item below, the page top, and the page bottom.

```

\l__marginalia_ysep_below_dim
\l__marginalia_ysep_page_top_dim
\l__marginalia_ysep_page_bottom_dim
256 \dim_new:N\l__marginalia_ysep_above_dim
257 \dim_new:N\l__marginalia_ysep_below_dim
258 \dim_new:N\l__marginalia_ysep_page_top_dim
259 \dim_new:N\l__marginalia_ysep_page_bottom_dim
260 \keys_define:nn { marginalia }
261 {
262   ysep~above .code:n
263     = \__marginalia_dim_set:Nn\l__marginalia_ysep_above_dim{#1},
264   ysep~below .code:n
265     = \__marginalia_dim_set:Nn\l__marginalia_ysep_below_dim{#1},
266   ysep~page~top .code:n
267     = \__marginalia_dim_set:Nn\l__marginalia_ysep_page_top_dim{#1},
268   ysep~page~bottom .code:n
269     = \__marginalia_dim_set:Nn\l__marginalia_ysep_page_bottom_dim{#1},
270   ysep~page~top~margin .meta:n = {
271     ysep~page~top
272     = \__marginalia_margin_top:
273   },
274   ysep~page~bottom~margin .meta:n = {
275     ysep~page~bottom
276     = \__marginalia_margin_bottom:

```

```

277 },
278 ysep~page~top~bottom~margin .meta:n = {
279   ysep~page~top~margin,
280   ysep~page~bottom~margin,
281 },
282 ysep~above~below .meta:n = {
283   ysep~below=#1,
284   ysep~above=#1,
285 },
286 ysep .meta:n = {
287   ysep~below=#1,
288   ysep~above=#1,
289   ysep~page~top=#1,
290   ysep~page~bottom=#1,
291 },
292 ysep~above~below .initial:n = \marginparpush,
293 ysep~page~top .initial:n = \__marginalia_margin_top:,
294 ysep~page~bottom .initial:n = \__marginalia_margin_bottom:,
295 }

```

(End of definition for \l__marginalia_ysep_above_dim and others.)

11.5.5 Appearance

Dimension keys to hold the width of the marginal content item, which can be dependent on where it appears on the page.

```

\l__marginalia_width_recto_right_dim
\l__marginalia_width_recto_left_dim
\l__marginalia_width_verso_left_dim
\l__marginalia_width_verso_right_dim
\l__marginalia_width_right_between_dim
\l__marginalia_width_left_between_dim
296 \dim_new:N\l__marginalia_width_recto_right_dim
297 \dim_new:N\l__marginalia_width_recto_left_dim
298 \dim_new:N\l__marginalia_width_verso_left_dim
299 \dim_new:N\l__marginalia_width_verso_right_dim
300 \dim_new:N\l__marginalia_width_right_between_dim
301 \dim_new:N\l__marginalia_width_left_between_dim
302 \keys_define:nn { marginalia }
303 {
304   width~recto~right .code:n
305     = \__marginalia_dim_set:Nn\l__marginalia_width_recto_right_dim{#1},
306   width~recto~left .code:n
307     = \__marginalia_dim_set:Nn\l__marginalia_width_recto_left_dim{#1},
308   width~verso~right .code:n
309     = \__marginalia_dim_set:Nn\l__marginalia_width_verso_right_dim{#1},
310   width~verso~left .code:n
311     = \__marginalia_dim_set:Nn\l__marginalia_width_verso_left_dim{#1},
312   width~right~between .code:n
313     = \__marginalia_dim_set:Nn\l__marginalia_width_right_between_dim{#1},
314   width~left~between .code:n
315     = \__marginalia_dim_set:Nn\l__marginalia_width_left_between_dim{#1},
316   width .meta:n = {
317     width~recto~right=#1,
318     width~recto~left=#1,
319     width~verso~right=#1,
320     width~verso~left=#1,
321     width~right~between=#1,
322     width~left~between=#1,
323   },

```

```

324 width~between .meta:n = {
325   width~right~between=#1,
326   width~left~between=#1,
327 },
328 width .initial:n = \marginparwidth,
329 }

```

Set up outer/inner keys, which will set the appropriate left/right keys.

```

330 \keys_define:nn { marginalia }
331 {
332   width~recto~outer .code:n = {
333     \__marginalia_outer_inner_set:NNNn
334     \dim_set:Nn
335     \l__marginalia_width_recto_right_dim
336     \l__marginalia_width_recto_left_dim
337     {#1}
338   },
339   width~recto~inner .code:n = {
340     \__marginalia_outer_inner_set:NNNn
341     \dim_set:Nn
342     \l__marginalia_width_recto_left_dim
343     \l__marginalia_width_recto_right_dim
344     {#1}
345   },
346   width~verso~outer .code:n = {
347     \__marginalia_outer_inner_set:NNNn
348     \dim_set:Nn
349     \l__marginalia_width_verso_left_dim
350     \l__marginalia_width_verso_right_dim
351     {#1}
352   },
353   width~verso~inner .code:n = {
354     \__marginalia_outer_inner_set:NNNn
355     \dim_set:Nn
356     \l__marginalia_width_verso_right_dim
357     \l__marginalia_width_verso_left_dim
358     {#1}
359   },
360   width~outer .meta:n = {
361     width~recto~outer=#1,
362     width~verso~outer=#1,
363   },
364   width~inner .meta:n = {
365     width~recto~inner=#1,
366     width~verso~inner=#1,
367   },
368 }

```

(End of definition for \l__marginalia_width_recto_right_dim and others.)

```

\l__marginalia_style_recto_right_tl
\l__marginalia_style_recto_left_tl
\l__marginalia_style_verso_left_tl
\l__marginalia_style_verso_right_tl
\l__marginalia_style_right_between_tl
\l__marginalia_style_left_between_tl

```

Token list keys to hold the style with which a marginal content item is typeset, which can be dependent on where it appears on the page.

```

369 \keys_define:nn { marginalia }
370 {
371   style~recto~right .tl_set:N = \l__marginalia_style_recto_right_tl,

```

```

372 style~recto~left .tl_set:N = \l__marginalia_style_recto_left_tl,
373 style~verso~right .tl_set:N = \l__marginalia_style_verso_right_tl,
374 style~verso~left .tl_set:N = \l__marginalia_style_verso_left_tl,
375 style~right~between .tl_set:N = \l__marginalia_style_right_between_tl,
376 style~left~between .tl_set:N = \l__marginalia_style_left_between_tl,
377 style .meta:n = {
378   style~recto~right=#1,
379   style~recto~left=#1,
380   style~verso~right=#1,
381   style~verso~left=#1,
382   style~right~between=#1,
383   style~left~between=#1,
384 },
385 style .initial:n = {},
386 }

```

Set up outer/inner keys, which will set the appropriate left/right keys.

```

387 \keys_define:nn { marginalia }
388 {
389   style~recto~outer .code:n = {
390     \__marginalia_outer_inner_set:NNNn
391     \tl_set:Nn
392       \l__marginalia_style_recto_right_tl
393       \l__marginalia_style_recto_left_tl
394       {#1}
395   },
396   style~recto~inner .code:n = {
397     \__marginalia_outer_inner_set:NNNn
398     \tl_set:Nn
399       \l__marginalia_style_recto_left_tl
400       \l__marginalia_style_recto_right_tl
401       {#1}
402   },
403   style~verso~outer .code:n = {
404     \__marginalia_outer_inner_set:NNNn
405     \tl_set:Nn
406       \l__marginalia_style_verso_left_tl
407       \l__marginalia_style_verso_right_tl
408       {#1}
409   },
410   style~verso~inner .code:n = {
411     \__marginalia_outer_inner_set:NNNn
412     \tl_set:Nn
413       \l__marginalia_style_verso_right_tl
414       \l__marginalia_style_verso_left_tl
415       {#1}
416   },
417 }

```

(End of definition for \l__marginalia_style_recto_right_tl and others.)

```

\l__marginalia_mark_recto_right_tl
\l__marginalia_mark_recto_left_tl
\l__marginalia_mark_verso_left_tl
\l__marginalia_mark_verso_right_tl
\l__marginalia_mark_right_between_tl
\l__marginalia_mark_left_between_tl

```

Token list keys to hold code for a mark to be placed adjacent to the line where the call to `\marginalia` is located, on the same side as the marginal content item.

```

418 \keys_define:nn { marginalia }
419 {

```

```

420 mark~recto~right .tl_set:N = \l__marginalia_mark_recto_right_tl,
421 mark~recto~left .tl_set:N = \l__marginalia_mark_recto_left_tl,
422 mark~verso~right .tl_set:N = \l__marginalia_mark_verso_right_tl,
423 mark~verso~left .tl_set:N = \l__marginalia_mark_verso_left_tl,
424 mark~right~between .tl_set:N = \l__marginalia_mark_right_between_tl,
425 mark~left~between .tl_set:N = \l__marginalia_mark_left_between_tl,
426 mark .meta:n = {
427     mark~recto~right=#1,
428     mark~recto~left=#1,
429     mark~verso~right=#1,
430     mark~verso~left=#1,
431     mark~right~between=#1,
432     mark~left~between=#1,
433 },
434 mark .initial:n = {},
435 }

```

Set up outer/inner keys, which will set the appropriate left/right keys.

```

436 \keys_define:nn { marginalia }
437 {
438     mark~recto~outer .code:n = {
439         \__marginalia_outer_inner_set:NNNn
440         \tl_set:Nn
441             \l__marginalia_mark_recto_right_tl
442             \l__marginalia_mark_recto_left_tl
443             {#1}
444     },
445     mark~recto~inner .code:n = {
446         \__marginalia_outer_inner_set:NNNn
447         \tl_set:Nn
448             \l__marginalia_mark_recto_left_tl
449             \l__marginalia_mark_recto_right_tl
450             {#1}
451     },
452     mark~verso~outer .code:n = {
453         \__marginalia_outer_inner_set:NNNn
454         \tl_set:Nn
455             \l__marginalia_mark_verso_left_tl
456             \l__marginalia_mark_verso_right_tl
457             {#1}
458     },
459     mark~verso~inner .code:n = {
460         \__marginalia_outer_inner_set:NNNn
461         \tl_set:Nn
462             \l__marginalia_mark_verso_right_tl
463             \l__marginalia_mark_verso_left_tl
464             {#1}
465     },
466 }

```

(End of definition for \l__marginalia_mark_recto_right_tl and others.)

11.5.6 Warning message configuration

\l__marginalia_max_problem_int	Integer keys to control the number of changes to be output
\l__marginalia_max_page_data_change_int	
\l__marginalia_max_item_data_change_int	

```

467 \keys_define:nn { marginalia }
468 {
469   max~problem~count .int_set:N = \l__marginalia_max_problem_int,
470   max~problem~count .initial:n= 40,
471   max~page~data~change~count .int_set:N = \l__marginalia_max_page_data_change_int,
472   max~page~data~change~count .initial:n = 10,
473   max~item~data~change~count .int_set:N = \l__marginalia_max_item_data_change_int,
474   max~item~data~change~count .initial:n = 10,
475 }

```

(End of definition for \l__marginalia_max_problem_int, \l__marginalia_max_page_data_change_int, and \l__marginalia_max_item_data_change_int.)

11.6 Processing data from the .aux file

`\marginalia@pagedata` This command is used to store version information in the .aux file. It currently does nothing, but may be used in future to avoid errors if changes are made in the format of the data written to the .aux file.

```

476 \cs_new:Npn \marginalia@version #1
477   {}

```

(End of definition for \marginalia@pagedata.)

`\marginalia@pagedata` This command is used to store page data in the .aux file.

```

478 \cs_new:Npn \marginalia@pagedata #1
479   {
480     \__marginalia_process_page_data:n{#1}
481   }

```

Initially `\__marginalia_process_page_data:n` is set to `\__marginalia_lua_store_page_data:n`. Thus, when the .aux file is read, `\marginalia@pagedata` will pass the page data to the Lua backend to be stored.

```

482 \cs_set_eq:NN
483   \__marginalia_process_page_data:n
484   \__marginalia_lua_store_page_data:n

```

(End of definition for \marginalia@pagedata.)

`\marginalia@itemdata` This command is used to store data for each marginal content item in the .aux file.

```

485 \cs_new:Npn \marginalia@itemdata #1
486   {
487     \__marginalia_process_item_data:n{#1}
488   }

```

(End of definition for \marginalia@itemdata.)

Initially `\__marginalia_process_item_data:n` is set to `\__marginalia_lua_store_item_data:n`. Thus, when the .aux file is read, `\marginalia@itemdata` will pass the item data to the Lua backend to be stored.

```

489 \cs_set_eq:NN
490   \__marginalia_process_item_data:n
491   \__marginalia_lua_store_item_data:n

```

At the `begindocument` hook, the `.aux` file has been read and closed. The Lua backend now stores the geometry and computes the vertical shift for each item. Then the handle for the main `.aux` file is stored for use in this package.

```

492 \hook_gput_code:nnn{ begindocument }{ marginalia/prepare }{
493   \__marginalia_lua_store_default_page_data:
494   \__marginalia_lua_compute_items:
495   \cs_set_eq:NN\l__marginalia_aux_iow\@mainaux
496 }

```

The `enddocument/afterlastpage` hook is before the `.aux` file is read back, so this is where `\__marginalia_process_page_data:n` and `\__marginalia_process_item_data:n` are set, respectively, to `\__marginalia_lua_check_page_data:n` and `\__marginalia_lua_check_item_data:n`. Thus, when the `.aux` file is read back, `\marginalia@pagedata` and `\marginalia@itemdata` will pass data to the Lua backend to be checked for changes.

```

497 \hook_gput_code:nnn{enddocument/afterlastpage}{ marginalia/check }{
498   \cs_set_eq:NN
499     \__marginalia_process_page_data:n
500     \__marginalia_lua_check_page_data:n
501   \cs_set_eq:NN
502     \__marginalia_process_item_data:n
503     \__marginalia_lua_check_item_data:n
504 }

```

`\__marginalia_write_reports:` Write reports if the `\marginalia` command has been used at least once, which is determined by checking whether `\g__marginalia_itemno_int` is non-zero.

```

505 \cs_new:Npn\__marginalia_write_reports:
506 {
507   \int_if_zero:nF{ \g__marginalia_itemno_int }
508   {
509     \__marginalia_lua_write_problem_report:
510     \__marginalia_lua_write_item_change_report:
511     \__marginalia_lua_write_page_change_report:
512   }
513 }

```

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

Use the `enddocument/info` hook to write the reports of changes and/or problems.

```

514 \hook_gput_code:nnn{ enddocument/info }{ marginalia/report } {
515   \__marginalia_write_reports:
516 }

```

11.7 Writing version data to the `.aux` file

`\__marginalia_write_version:` This command will be used to write the package version to the `.aux` file.

```

517 \cs_new:Npn\__marginalia_write_version:
518 {
519   \iow_now:Ne\l__marginalia_aux_iow{
520     \token_to_str:N\marginalia@version{
521       \use:c{ver@marginalia.sty}
522     }
523   }
524 }

```

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

11.8 Writing page data to the .aux file

To compute the positions of marginal content items, certain page layout data is required. And since all the computation takes place at the beginning of the document, it is necessary to write this information to the .aux file.

`\g_marginalia_pagedatano_int` Global integer variable to index page data items written to the .aux file.

```
525 \int_new:N\g_marginalia_pagedatano_int
```

(End of definition for \g_marginalia_pagedatano_int.)

`\_marginalia_write_page_data:` This command will be used to write the current page data to the .aux file. It is initially defined to do nothing, so that the use of `\marginalianewgeometry` in the preamble does not cause errors (because the .aux file is not available for writing until `begindocument/end`).

```
526 \cs_set_eq:NN\_marginalia_write_page_data:\prg_do_nothing:
527 \cs_new:Npn\_marginalia_write_page_data_real:
528 {
529   \int_gincr:N\g_marginalia_pagedatano_int
530   \legacy_if:nTF{@twocolumn}
531     { \int_set:Nn\l_tmpa_int{2} }
532     { \int_set:Nn\l_tmpa_int{1} }
533   \iow_now:Ne\l_marginalia_aux_iow{
534     \token_to_str:N\marginalia@pagedata{
535       pagedatano=\int_value:w\g_marginalia_pagedatano_int,
536       abspageno=\int_eval:n{\g_shipout_readonly_int+1},
537       hoffset=\int_value:w\hoffset,
538       voffset=\int_value:w\voffset,
539       pageheight=\int_value:w\pageheight,
540       oddsidemargin=\int_value:w\oddsidemargin,
541       evensidemargin=\int_value:w\evensidemargin,
542       textwidth=\int_value:w\textwidth,
543       columncount=\int_value:w\l_tmpa_int,
544       columnwidth=\int_value:w\columnwidth,
545       columnsep=\int_value:w\columnsep,
546       twoside=\bool_to_str:n{\legacy_if_p:n{@twoside}},
547     }
548   }
549 }
```

At the `begindocument/end` hook, the .aux file has been opened for writing, and so the macro `\_marginalia_write_page_data:` is enabled and the initial page data is written out.

```
550 \hook_gput_code:nnn{ begindocument/end }{ marginalia/initial }
551 {
552   \_marginalia_write_version:
553   \cs_set_eq:NN
554     \_marginalia_write_page_data:
555     \_marginalia_write_page_data_real:
556   \_marginalia_write_page_data:
557 }
```

(End of definition for _marginalia_write_page_data:.)

11.9 Marginal content item processing

11.9.1 Variables

Variables set by L^AT_EX.

`\g__marginalia_itemno_int` Global integer variable to index marginal content items.
558 `\int_new:N\g__marginalia_itemno_int`
(End of definition for `\g__marginalia_itemno_int`.)

`\l__marginalia_item_box` Box variable to hold the typeset marginal content item.
559 `\box_new:N\l__marginalia_item_box`
(End of definition for `\l__marginalia_item_box`.)

`\l__marginalia_item_height_dim` Dimension variables to hold the height and depth of the typeset margin content item.
`\l__marginalia_item_depth_dim` 560 `\dim_new:N\l__marginalia_item_height_dim`
561 `\dim_new:N\l__marginalia_item_depth_dim`
(End of definition for `\l__marginalia_item_height_dim` and `\l__marginalia_item_depth_dim`.)

Variables set by Lua. The following variables will be set by the Lua backend via `tex.count` and `tex.dimen` when `\__marginalia_lua_load_item_data:n` is called.

`\l__marginalia_page_int` Integer variable for the page on which the marginal content item appears. This variable will be made available via `\marginaliapage` within the `\content` of `\marginalia`.
562 `\int_new:N\l__marginalia_page_int`
(End of definition for `\l__marginalia_page_int`.)

`\l__marginalia_column_computed_int` Integer variable for the column next to which the marginal content item appears. This variable will be made available via `\marginaliacolumn` within the `\content` of `\marginalia`.
563 `\int_new:N\l__marginalia_column_computed_int`
(End of definition for `\l__marginalia_column_computed_int`.)

`\l__marginalia_xshift_computed_dim` Dimension variables to hold the differences in x and y coordinates between the call to `\marginalia` and the position where the marginal content item should appear.
`\l__marginalia_yshift_computed_dim` 564 `\dim_new:N\l__marginalia_xshift_computed_dim`
565 `\dim_new:N\l__marginalia_yshift_computed_dim`
(End of definition for `\l__marginalia_xshift_computed_dim` and `\l__marginalia_yshift_computed_dim`.)

`\l__marginalia_side_computed_int` Integer variable to indicate the side of the text block or column on which the marginal content item should be placed: 0 = right and 1 = left.
566 `\int_new:N\l__marginalia_side_computed_int`
(This variable could be a boolean, but an integer is used because there is no canonical access to booleans from Lua.)
(End of definition for `\l__marginalia_side_computed_int`.)

`\l__marginalia_marginno_computed_int` Integer variable to indicate in which margin the content will be placed, to enable quick selection of width and style: 0 = recto outer, 1 = recto inner, 2 = verso outer, 3 = verso inner, 4 = right between, 5 = left between.

```
567 \int_new:N\l__marginalia_marginno_computed_int
```

(End of definition for `\l__marginalia_marginno_computed_int`.)

`\l__marginalia_enabled_computed_int` Integer variable to indicate whether the marginal content item is enabled: 0 = disabled, 1 = enabled.

```
568 \int_new:N\l__marginalia_enabled_computed_int
```

(This variable could be a boolean, but an integer is used because there is no canonical access to booleans from Lua.)

(End of definition for `\l__marginalia_enabled_computed_int`.)

11.9.2 Core macro

`\__marginalia_process_item:nn` This macro does most of the work in setting the marginal content item. The first parameter is `<options>`, the second is `<content>`.

```
569 \cs_new:Npn\__marginalia_process_item:nn #1#2
570 {
```

First, increment the index, then enter a group where all the action will happen.

```
571 \int_gincr:N\g__marginalia_itemno_int
572 \group_begin:
```

Process `<options>`. These settings apply locally inside the group.

```
573 \keys_set:nn{marginalia}{ #1 }
```

Get item data from the Lua backend: the integer variables `\l__marginalia_page_int`, `\l__marginalia_column_computed_int`, `\l__marginalia_side_computed_int`, `\l__marginalia_enabled_computed_int`, and the dimension variables `\l__marginalia_xshift_computed_dim`, and `\l__marginalia_yshift_computed_dim` are set by Lua via `tex.count` and `tex.dimen`. If no data is available (if, for instance, no data has been stored from a previous run), default values will be set by Lua. On later runs, the Lua backend will supply the values computed from the data written to the `.aux` file on the previous run.

```
574 \__marginalia_lua_load_item_data:n
575 { \int_value:w\g__marginalia_itemno_int }
```

Choose the correct auxiliary function for typesetting, depending on which mode $\text{\TeX}$ is in.

```
576 \mode_if_math:TF
577 {
578   \cs_set_eq:NN
579   \__marginalia_typeset:n
580   \__marginalia_typeset_mmode:n
581 }
582 {
583   \legacy_if:nT{@inlabel}
584   { \leavevmode }
585   \mode_if_horizontal:TF
586   {
587     \cs_set_eq:NN
```

```

588         \l__marginalia_typeset:n
589         \l__marginalia_typeset_hmode:n
590     }
591     {
592         \cs_set_eq:NN
593         \l__marginalia_typeset:n
594         \l__marginalia_typeset_vmode:n
595     }
596 }

```

Choose the correct box in which to typeset the item for the desired vertical alignment, which has been stored in `\l__marginalia_valign_int`.

```

597 \int_case:nn{\l__marginalia_valign_int}
598 {
599     {1}{ \cs_set_eq:NN\l__marginalia_item_box_set:Nn\ vbox_set_top:Nn }
600     {2}{ \cs_set_eq:NN\l__marginalia_item_box_set:Nn\ vbox_set:Nn }
601     {3}{ \cs_set_eq:NN\l__marginalia_item_box_set:Nn\l__marginalia_vbox_set_center:Nn }
602     {4}{ \cs_set_eq:NN\l__marginalia_item_box_set:Nn\l__marginalia_vbox_set_midway:Nn }
603 }

```

Choose the correct horizontal separation, width, style, and mark for the item.

```

604 \l__marginalia_set_xsep_width_style_mark:

```

Typeset the `<content>` into `\l__marginalia_item_box`. Use `\@parboxrestore` for brevity, even though `\hsize` and `\linewidth` are subsequently set to `\l__marginalia_width_dim`. Make available `\marginaliapage` and `\marginaliacolumn`.

```

605 \l__marginalia_tagging_socket:n {marginpar/begin}
606 \l__marginalia_item_box_set:Nn\l__marginalia_item_box{
607     \@parboxrestore
608     \l__marginalia_tagging_socket:n {para/restore}
609     \normalfont\normalsize
610
611     \tl_use:N\l__marginalia_style_tl
612     \dim_set_eq:NN\hsize\l__marginalia_width_dim
613     \dim_set_eq:NN\linewidth\hsize
614
615     \cs_set_eq:NN\marginaliapage\l__marginalia_page_int
616     \cs_set_eq:NN\marginaliacolumn\l__marginalia_column_computed_int
617
618     \group_begin:
619     \ignorespaces
620     #2
621     \par
622     \group_end:
623 }
624 \l__marginalia_tagging_socket:n{marginpar/end}

```

Measure `\l__marginalia_item_box`.

```

625 \dim_set:Nn\l__marginalia_item_height_dim
626     {\box_ht:N\l__marginalia_item_box}
627 \dim_set:Nn\l__marginalia_item_depth_dim
628     {\box_dp:N\l__marginalia_item_box}

```

Everything is now ready to place the item on the page and write the necessary data to the `.aux` file. Use the chosen auxiliary function for typesetting, and immediately use `\savepos` to store the callout position.

```

629     \_marginalia_typeset:n{
630         \savepos

```

Write the item data to the .aux file. All tokens that will change for future items, and which are currently meaningful, are expanded now; the remainder will be expanded at shipout time, when *they* are meaningful.

```

631     \iow_shipout_e:Ne\l__marginalia_aux_iow{
632         \token_to_str:N\marginalia@itemdata{
633             itemno=\int_value:w\g__marginalia_itemno_int,
634             abspageno=\exp_not:N\int_eval:n{g_shipout_readonly_int},
635             pageno=\exp_not:N\int_value:w\c@page,
636             type=\str_use:N\int_value:w\l__marginalia_type_int,
637             xpos=\exp_not:N\int_value:w\lastxpos,
638             ypos=\exp_not:N\int_value:w\lastypos,
639             height=\int_value:w\l__marginalia_item_height_dim,
640             depth=\int_value:w\l__marginalia_item_depth_dim,
641             pos=\int_value:w\l__marginalia_pos_int,
642             column=\int_value:w\l__marginalia_column_int,
643             yshift=\int_value:w\l__marginalia_default_yshift_dim,
644             ysep~above=\int_value:w\l__marginalia_ysep_above_dim,
645             ysep~below=\int_value:w\l__marginalia_ysep_below_dim,
646             ysep~page~top=\int_value:w\l__marginalia_ysep_page_top_dim,
647             ysep~page~bottom=\int_value:w\l__marginalia_ysep_page_bottom_dim,
648         }
649     }

```

Finally, if the item is enabled, typeset it onto the page: shift the item by

$$|\l__marginalia_xshift_computed_dim| + |\l__marginalia_xsep_dim|$$

to the right or left (depending on `\l__marginalia_side_computed_int` in the appropriate overlap `\hbox`, then use `\_marginalia_place_item_box:` for the vertical placement. To make sure that item placement is correct in right-to-left (TRT) text, placement happens inside a zero-width `\hbox` in which the text direction is set to left-to-right (TLT). This change does not affect the content of the item, since it has already been typeset into `\l__marginalia_item_box`.

```

650     \int_if_zero:nF{\l__marginalia_enabled_computed_int}
651     {
652         \hbox_to_zero:n{
653             \textdir TLT
654             \int_if_zero:nTF{ \l__marginalia_side_computed_int }
655             {
656                 \hbox_overlap_right:n{
657                     \kern\l__marginalia_xshift_computed_dim
658                     \tl_if_empty:NF\l__marginalia_mark_tl
659                     {
660                         \hbox_overlap_right:n{ \tl_use:N\l__marginalia_mark_tl }
661                     }
662                     \kern\l__marginalia_xsep_dim
663                     \_marginalia_place_item_box:
664                 }
665             }
666         }
667         \hbox_overlap_left:n{

```

```

668         \_marginalia_place_item_box:
669         \kern\l__marginalia_xsep_dim
670         \tl_if_empty:NF\l__marginalia_mark_tl
671         {
672             \hbox_overlap_left:n{ \tl_use:N\l__marginalia_mark_tl }
673         }
674         \kern-\l__marginalia_xshift_computed_dim
675     }
676 }
677 }
678 }
679 }

```

Close the group started near the beginning of `\_marginalia_process_item:nn`.

```

680     \group_end:
681 }

```

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

11.9.3 Horizontal separation, width, style, mark selection

`\_marginalia_set_xsep_width_style_mark:` Set `\l__marginalia_xsep_dim`, `\l__marginalia_width_dim`, and `\l__marginalia_style_tl`, based on `\l__marginalia_marginno_computed_int`.

```

682 \cs_new:Npn\_marginalia_set_xsep_width_style_mark:
683 {
684     \int_case:nn{\l__marginalia_marginno_computed_int}
685     {
686         {0}
687         {
688             \cs_set_eq:NN\l__marginalia_xsep_dim
689             \l__marginalia_xsep_recto_right_dim
690             \cs_set_eq:NN\l__marginalia_width_dim
691             \l__marginalia_width_recto_right_dim
692             \cs_set_eq:NN\l__marginalia_style_tl
693             \l__marginalia_style_recto_right_tl
694             \cs_set_eq:NN\l__marginalia_mark_tl
695             \l__marginalia_mark_recto_right_tl
696         }
697         {1}
698         {
699             \cs_set_eq:NN\l__marginalia_xsep_dim
700             \l__marginalia_xsep_recto_left_dim
701             \cs_set_eq:NN\l__marginalia_width_dim
702             \l__marginalia_width_recto_left_dim
703             \cs_set_eq:NN\l__marginalia_style_tl
704             \l__marginalia_style_recto_left_tl
705             \cs_set_eq:NN\l__marginalia_mark_tl
706             \l__marginalia_mark_recto_left_tl
707         }
708         {2}
709         {
710             \cs_set_eq:NN\l__marginalia_xsep_dim
711             \l__marginalia_xsep_verso_right_dim
712             \cs_set_eq:NN\l__marginalia_width_dim

```

```

713         \l__marginalia_width_verso_right_dim
714         \cs_set_eq:NN\l__marginalia_style_tl
715         \l__marginalia_style_verso_right_tl
716         \cs_set_eq:NN\l__marginalia_mark_tl
717         \l__marginalia_mark_verso_right_tl
718     }
719 {3}
720 {
721     \cs_set_eq:NN\l__marginalia_xsep_dim
722     \l__marginalia_xsep_verso_left_dim
723     \cs_set_eq:NN\l__marginalia_width_dim
724     \l__marginalia_width_verso_left_dim
725     \cs_set_eq:NN\l__marginalia_style_tl
726     \l__marginalia_style_verso_left_tl
727     \cs_set_eq:NN\l__marginalia_mark_tl
728     \l__marginalia_mark_verso_left_tl
729 }
730 {4}
731 {
732     \cs_set_eq:NN\l__marginalia_xsep_dim
733     \l__marginalia_xsep_right_between_dim
734     \cs_set_eq:NN\l__marginalia_width_dim
735     \l__marginalia_width_right_between_dim
736     \cs_set_eq:NN\l__marginalia_style_tl
737     \l__marginalia_style_right_between_tl
738     \cs_set_eq:NN\l__marginalia_mark_tl
739     \l__marginalia_mark_right_between_tl
740 }
741 {5}
742 {
743     \cs_set_eq:NN\l__marginalia_xsep_dim
744     \l__marginalia_xsep_left_between_dim
745     \cs_set_eq:NN\l__marginalia_width_dim
746     \l__marginalia_width_left_between_dim
747     \cs_set_eq:NN\l__marginalia_style_tl
748     \l__marginalia_style_left_between_tl
749     \cs_set_eq:NN\l__marginalia_mark_tl
750     \l__marginalia_mark_left_between_tl
751 }
752 }
753 }

```

(End of definition for __marginalia_set_xsep_width_style_mark:.)

11.9.4 Auxiliary box macros

__marginalia_vbox_set_center:Nn Typesets the second parameter at natural height and stores the result inside a box register supplied as the first parameter, with the baseline being mid-way between the top and bottom of the box.

```

754 \cs_new:Npn \__marginalia_vbox_set_center:Nn #1#2
755 {
756     \vbox_set_top:Nn #1{#2}
757     \dim_set:Nn \l_tmpa_dim{ \box_ht:N#1 + \box_dp:N#1 }
758     \box_set_ht:Nn #1 { .5\l_tmpa_dim }

```

```

759     \box_set_dp:Nn #1 { .5\l_tmpa_dim }
760 }

```

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

`\__marginalia_vbox_set_midway:Nn` Typesets the second parameter at natural height and stores the result inside a box register supplied as the first parameter, with the baseline being mid-way between the top and bottom baselines of the items in the box.

```

761 \cs_new:Npn \__marginalia_vbox_set_midway:Nn #1#2
762 {

```

The distance between the topmost and bottommost baselines is equal to the difference in depths between the results of typesetting using a `\vtop` and a `\vbox`. So typeset into both (suspending tagging for the latter), calculate the difference, and adjust the height and depth of the first.

```

763     \vbox_set_top:Nn #1{#2}
764     \tag_suspend:n{ marginalia }
765     \vbox_set:Nn \l_tmpa_box {#2}
766     \tag_resume:n{ marginalia }
767     \dim_set:Nn \l_tmpa_dim{ \box_dp:N#1 - \box_dp:N\l_tmpa_box}
768     \box_set_ht:Nn #1 { \box_ht:N#1 + .5\l_tmpa_dim }
769     \box_set_dp:Nn #1 { \box_dp:N#1 - .5\l_tmpa_dim }
770 }

```

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

11.9.5 Placement macros

`\__marginalia_place_item_box:` Place the item that has been set in `\l__marginalia_item_box`, vertically shifted by `\l__marginalia_yshift_computed_dim` and `\smashed` to avoid altering vertical spacing in the main text.

```

771 \cs_new:Npn \__marginalia_place_item_box:
772 {
773     \smash
774     {
775         \box_move_up:nn{\l__marginalia_yshift_computed_dim}
776         {
777             \box_use:N\l__marginalia_item_box
778         }
779     }
780 }

```

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

`\__marginalia_typeset_mmode:n` These three macros handle typesetting in math mode, horizontal mode, and vertical mode. Nothing special needs to be done in math mode. In horizontal mode, `\@bsphack...``\@esphack` avoids double spacing. In vertical mode, `\if@nobreak` is saved, a new paragraph is started, the item is typeset, the paragraph is ended, a vertical skip of `-\baselineskip` is added, which should ‘hide’ that invisible paragraph, and `\if@nobreak` is restored to the saved value.

```

781 \cs_new:Npn \__marginalia_typeset_mmode:n #1
782 {
783     #1
784 }

```

```

785 \cs_new:Npn\__marginalia_typeset_hmode:n #1
786 {
787   \@bsphack
788   #1
789   \@esphack
790 }
791 \bool_new:N\l__marginalia_nobreak_bool
792 \cs_new:Npn\__marginalia_typeset_vmode:n #1
793 {
794   \bool_set:Nn\l__marginalia_nobreak_bool{ \legacy_if_p:n{@nobreak} }
795   \nobreak\noindent #1\par
796   \skip_vertical:n{-\baselineskip}
797   \legacy_if_gset:nn{ @nobreak }{ \l__marginalia_nobreak_bool }
798 }

```

(End of definition for `\__marginalia_typeset_mmode:n`, `\__marginalia_typeset_hmode:n`, and `\__marginalia_typeset_vmode:n`.)

11.10 User commands

Finally, set up the commands for the user.

`\marginalia` This is the main user command for creating a marginal content item. This macro does nothing but hand off to `\__marginalia_process_item:nn`.

```

799 \NewDocumentCommand{\marginalia}{ 0{} +m }
800 {
801   \__marginalia_process_item:nn{#1}{#2}
802 }

```

(End of definition for `\marginalia`. This function is documented on page 5.)

`\marginaliasetup` The user command to set the configuration. This macro does nothing but hand off to `\__marginalia_setup:n`.

```

803 \NewDocumentCommand{\marginaliasetup}{ 1 m }
804 {
805   \__marginalia_setup:n{ #1 }
806 }

```

(End of definition for `\marginaliasetup`. This function is documented on page 5.)

`\marginalianewgeometry` The user command to signal that the page geometry has been changed.

```

807 \NewDocumentCommand{\marginalianewgeometry}{}
808 {
809   \__marginalia_write_page_data:
810 }

```

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

```

811 </package>

```

12 Implementation (Lua backend)

812 $\langle *lua \rangle$

12.1 Initial set-up

Module identification/version information.

```
813 local module_name = 'marginalia'
814 local marginalia_module = {
815     name      = module_name,
816     version   = '0.84.3',
817     date      = '2026-08-12',
818     description = 'marginalia',
819     author    = 'Alan J. Cain',
820     copyright  = 'Alan J. Cain',
821     license    = 'LPPL v1.3c'
822 }
823 luatexbase.provides_module(marginalia_module)
```

12.2 Message functions

`info` Functions for writing info or warning messages. First argument is used as a format string
`warning` for later arguments.

```
824 local function info(s,...)
825     luatexbase.module_info(module_name,s:format(...))
826 end
827 local function warning(s,...)
828     luatexbase.module_warning(module_name,s:format(...))
829 end
830 local function error_(s,...)
831     luatexbase.module_error(module_name,s:format(...))
832 end
```

(End of definition for info and warning.)

12.3 Module variables

Module tables for page_data and item_data.

```
833 local PAGE_DATA_MAIN_TABLE = {}
834 local ITEM_DATA_MAIN_TABLE = {}
```

Module tables for compiling reports.

```
835 local PROBLEM_REPORT_TABLE = {}
836 local PAGE_CHANGE_REPORT_TABLE = {}
837 local ITEM_CHANGE_REPORT_TABLE = {}
```

Module variable indicating whether the document binding is on the left.

```
838 local binding_left = true
839
840 local function set_binding_left(value)
841     binding_left = value
842 end
```

12.4 Constants

Type constants. These match the possible values for the `type` key.

```
843 local TYPE_NORMAL = 1
844 local TYPE_FIXED = 2
845 local TYPE_OPTFIXED = 3
```

Position constants. These match the possible values for the `pos` key.

```
846 local POS_AUTO = 1
847 local POS_REVERSE = 2
848 local POS_OUTER = 3
849 local POS_INNER = 4
850 local POS_RIGHT = 5
851 local POS_LEFT = 6
852 local POS_NEAREST = 7
```

12.5 Keys for tables

The strings listed in this subsection are constants used to index the tables. Also listed are the types of values that are indexed by each key. Note that values listed below as **dimensions** are actually integers, giving the dimension in $\text{\TeX}$ scaled points (sp)

12.5.1 Keys for both page and item data tables

Integer: Absolute page number in output file (not on-page number), used in both `page__data` and `item__data` tables

```
853 local KEY_ABSPAGENO = 'abspageno'
```

Boolean: Used to mark `page__data` or `item__data` as checked when the `.aux` file is read back at the end of the document

```
854 local KEY_CHECKED = 'checked'
```

12.5.2 Keys for page data tables, layout etc.

Integer: Used only to distinguish instances of data written to `.aux` file

```
855 local KEY_PAGEDATANO = 'pagedatano'
```

Dimensions: Value of next two will always be equivalent of 1 in, but it is simpler to keep all geometry data together.

```
856 local KEY_HOFFSETORIGIN = 'hoffsetorigin'
857 local KEY_VOFFSETORIGIN = 'voffsetorigin'
```

Dimensions: corresponding to obvious $\text{\LaTeX}$ dimensions

```
858 local KEY_HOFFSET = 'hoffset'
859 local KEY_VOFFSET = 'voffset'
860 local KEY_PAGEHEIGHT = 'pageheight'
861 local KEY_ODDSIDEMARGIN = 'oddsidemargin'
862 local KEY_EVENSIDEMARGIN = 'evensidemargin'
863 local KEY_TEXTWIDTH = 'textwidth'
864 local KEY_COLUMNWIDTH = 'columnwidth'
865 local KEY_COLUMNSEP = 'columnsep'
```

Integer: either 1 or 2, depending on whether $\text{\LaTeX}$ was in one- or two-column mode

```
866 local KEY_COLUMNCOUNT = 'columncount'
```

Boolean: true iff $\text{\LaTeX}$ is in twoside mode

```
867 local KEY_TWOSIDE = 'twoside'
```

12.5.3 Keys for item data tables

Integer: Used to identify data with item

```
868 local KEY_ITEMNO = 'itemno'
```

Integer: On-page number

```
869 local KEY_PAGENO = 'pageno'
```

Dimensions: x and y positions of call to `\marginalia`

```
870 local KEY_XPOS = 'xpos'
```

```
871 local KEY_YPOS = 'ypos'
```

Dimensions: Height and depth of typeset item

```
872 local KEY_HEIGHT = 'height'
```

```
873 local KEY_DEPTH = 'depth'
```

Integer: Specified type, following `TYPE_*`

```
874 local KEY_TYPE = 'type'
```

Integer: corresponds to value of `pos` key and the constants `POS_*`: 1 = `auto`, 2 = `reverse`, 3 = `outer`, 4 = `inner`, 5 = `right`, 6 = `left`, 7 = `nearest`.

```
875 local KEY_POS = 'pos'
```

Integer: corresponds to value of `column` key: -1 = `auto`, 0 = `one`, 1 = `left`, 2 = `right`

```
876 local KEY_COLUMN = 'column'
```

Dimension: specified vertical shift

```
877 local KEY_YSHIFT = 'yshift'
```

Dimensions: specified vertical separations

```
878 local KEY_YSEP_ABOVE = 'ysep above'
```

```
879 local KEY_YSEP_BELOW = 'ysep below'
```

```
880 local KEY_YSEP_PAGE_TOP = 'ysep page top'
```

```
881 local KEY_YSEP_PAGE_BOTTOM = 'ysep page bottom'
```

The preceding keys refer to values that will be supplied from $\text{\LaTeX}$. The remaining values will be computed in Lua and passed back to $\text{\LaTeX}$.

Integer: column in which the call to `\marginalia` was located: 0 = one-column, 1 = left, 2 = right

```
882 local KEY_COLNO_COMPUTED = 'colno computed'
```

Dimension: Horizontal shift between the call to `\marginalia` and the margin in which the item should be located

```
883 local KEY_XSHIFT_COMPUTED = 'xshift computed'
```

Dimension: Computed vertical shift

```
884 local KEY_YSHIFT_COMPUTED = 'yshift computed'
```

Integer: Side of text on which the item will appear: 0 = right, 1 = left

```
885 local KEY_SIDE_COMPUTED = 'side computed'
```

Integer: Number of margin in which the item will appear, 0 = recto right, 1 = recto left, 2 = verso left, 3 = verso right, 4 = right between, 5 = left between

```
886 local KEY_MARGINNO_COMPUTED = 'marginno computed'
```

Boolean: Whether the item will actually appear on the page

```
887 local KEY_ENABLED_COMPUTED = 'enabled computed'
```

12.6 Utility functions

list_filter
12 Code adapted from
<https://stackoverflow.com/a/53038524>.

Take a list `t` and remove from it any elements for which the function `f` does not return true. (The index `j` is always the destination index to which a ‘keep’ element is moved.)¹²

```
888 local function list_filter(t, f)
889     local j = 1
890     local n = #t
891
892     for i=1,n do
893         if (f(t[i])) then
894             if (i ~= j) then
895                 t[j] = t[i]
896                 t[i] = nil
897             end
898             j = j + 1
899         else
900             t[i] = nil
901         end
902     end
903
904 end
```

(End of definition for list_filter.)

list_filter Return boolean true iff `s` is exactly the string ‘true’.

```
905 local function toboolean(s)
906     return s == "true"
907 end
```

(End of definition for list_filter.)

get_data_page_number Take a item or page data and return a human-readable string indicating the page to which the data pertains.

```
908 local function get_data_page_number(data)
909     local pageno = data[KEY_PAGENO]
910     if pageno ~= nil then
911         return 'p' .. pageno .. ' (' .. data[KEY_ABSPAGENO] .. ')'
912     else
913         return data[KEY_ABSPAGENO]
914     end
915 end
```

(End of definition for get_data_page_number.)

12.7 Generic page/item data functions

parse_data Parse `keyvalue_string` and return the corresponding data as a table. The `keyvalue_string` is expected to be of precisely the kind written to the `.aux` file as the parameter of `\marginalia@pagedata` or `\marginalia@notedata`.

Ignore any keys in `keyvalue_string` that are not listed in `conversion_table`. Fill in any missing value with values from `defaults_table`.

`conversion_table` is indexed by possible keys, with values equal to functions to convert the corresponding value string to the value that should appear in the returned table.

`defaults_table` is indexed by keys that *will* appear in the returned table, using the corresponding value unless it was given in `keyvalue_string` and the key appeared in `conversion_table`.

```

916 local function parse_data(keyvalue_string,conversion_table,defaults_table)
917
918     local key
919     local value
920     local result = {}
921
922     for s in string.gmatch(keyvalue_string,'([^\,]+)') do
923
924         key,value = string.match(s,'^(.+)=(.+)$')
925         local conv = conversion_table[key]
926         if conv ~= nil then
927             result[key] = conv(value)
928         end
929
930     end
931
932     for key,value in pairs(defaults_table) do
933         if not(result[key] ~= nil) then
934             result[key] = value
935         end
936     end
937
938     return result
939
940 end

```

(End of definition for parse_data.)

`check_data` Check `keyvalue_string` against stored data. If it is new or has changed, append a report to `report_table`. Set the `KEY_CHECKED` of the data item to true.

The `keyvalue_string` is processed using `conversion_table` and `defaults_table` as per the `parse_data` function. The resulting table is compared to the table in `data_table` with the same value whose key is `data_table_key`. The tables are compared using the fields indexed by keys in `conversion_table`.

```

941 local function check_data(keyvalue_string,conversion_table,defaults_table,
942                           data_table,data_table_key_field,report_table)
943
944     local new_data = parse_data(keyvalue_string,
945                                 conversion_table,defaults_table)
946
947     local data_table_key = new_data[data_table_key_field]
948
949     local stored_data = data_table[data_table_key]
950     if stored_data == nil then
951         table.insert(
952             report_table,
953             get_data_page_number(new_data) .. ' New'
954         )
955     else
956         local change_report = ''

```

```

957     for k,_ in pairs(conversion_table) do
958         if stored_data[k] ~= new_data[k] then
959             change_report = change_report
960             .. ' ' .. k .. ':' ..
961             tostring(stored_data[k]) .. '->' .. tostring(new_data[k])
962         end
963     end
964     if change_report ~= '' then
965         table.insert(
966             report_table,
967             get_data_page_number(new_data) .. ' ' .. change_report
968         )
969     end
970     stored_data[KEY_CHECKED] = true
971 end
972
973 end

```

(End of definition for check_data.)

`check_removed_data` Check whether data have been removed from `data_table`, which corresponds to some entry having the value of `KEY_CHECKED` being false. In this case, append a report to `report_table`.

```

974 local function check_removed_data(data_table,report_table)
975     for _,data in pairs(data_table) do
976         if not data[KEY_CHECKED] then
977             table.insert(
978                 report_table,
979                 ' Removed'
980             )
981             break
982         end
983     end
984 end

```

(End of definition for check_removed_data.)

12.8 Processing of page data from .aux file

Conversion and default tables.

```

985 local PAGE_DATA_CONVERSION_TABLE = {
986     [KEY_PAGEDATANO] = tonumber,
987     [KEY_ABSPAGENO] = tonumber,
988     [KEY_HOFFSETORIGIN] = tonumber,
989     [KEY_VOFFSETORIGIN] = tonumber,
990     [KEY_HOFFSET] = tonumber,
991     [KEY_VOFFSET] = tonumber,
992     [KEY_PAGEHEIGHT] = tonumber,
993     [KEY_ODDSIDEMARGIN] = tonumber,
994     [KEY_EVENSIDEMARGIN] = tonumber,
995     [KEY_COLUMNCOUNT] = tonumber,
996     [KEY_COLUMNWIDTH] = tonumber,
997     [KEY_COLUMNSEP] = tonumber,
998     [KEY_TEXTWIDTH] = tonumber,

```

```

999     [KEY_TWOSIDE] = toboolean,
1000 }
1001 local PAGE_DATA_DEFAULT_TABLE = {
1002     [KEY_PAGEDATANO] = 0,
1003     [KEY_A BSPAGENO] = 0,
1004     [KEY_HOFFSETORIGIN] = tex.sp('1in'),
1005     [KEY_VOFFSETORIGIN] = tex.sp('1in'),
1006     [KEY_HOFFSET] = tex.dimen['hoffset'],
1007     [KEY_VOFFSET] = tex.dimen['voffset'],
1008     [KEY_PAGEHEIGHT] = tex.dimen['pageheight'],
1009     [KEY_ODDSIDEMARGIN] = tex.dimen['oddsidemargin'],
1010     [KEY_EVENSIDEMARGIN] = tex.dimen['evensidemargin'],
1011     [KEY_TEXTWIDTH] = tex.dimen['textwidth'],
1012     [KEY_COLUMNWIDTH] = tex.dimen['columnwidth'],
1013     [KEY_COLUMNSEP] = tex.dimen['columnsep'],
1014     [KEY_COLUMNCOUNT] = 1,
1015     [KEY_TWOSIDE] = false,
1016     [KEY_CHECKED] = false,
1017 }

```

store_page_data Store page data supplied by keyvalue_string in PAGE_DATA_MAIN_TABLE.

```

1018 local function store_page_data(keyvalue_string)
1019
1020     local page_data = parse_data(keyvalue_string,
1021                                 PAGE_DATA_CONVERSION_TABLE,
1022                                 PAGE_DATA_DEFAULT_TABLE)
1023
1024     PAGE_DATA_MAIN_TABLE[page_data[KEY_PAGEDATANO]] = page_data
1025
1026 end

```

(End of definition for store_page_data.)

store_default_page_data Store default page data in PAGE_DATA_MAIN_TABLE, so that there is some data to work with when computing item positions, even on a first run, when no page data has been written to the .aux file.

```

1027 local function store_default_page_data()
1028
1029     default_page_data = parse_data('',
1030                                   PAGE_DATA_CONVERSION_TABLE,
1031                                   PAGE_DATA_DEFAULT_TABLE)
1032
1033     default_page_data[KEY_A BSPAGENO] = 1
1034     default_page_data[KEY_CHECKED] = true
1035
1036     PAGE_DATA_MAIN_TABLE[0] = default_page_data
1037
1038 end

```

(End of definition for store_default_page_data.)

check_page_data Check whether page_data supplied by keyvalue_string differs from that in PAGE_DATA_MAIN_TABLE, appending reports to PAGE_CHANGE_REPORT_TABLE if so.

```

1039 local function check_page_data(keyvalue_string)

```

```

1040
1041     check_data(keyvalue_string,
1042                 PAGE_DATA_CONVERSION_TABLE,PAGE_DATA_DEFAULT_TABLE,
1043                 PAGE_DATA_MAIN_TABLE,KEY_PAGEDATANO,
1044                 PAGE_CHANGE_REPORT_TABLE)
1045
1046 end

```

(End of definition for check_page_data.)

12.9 Processing of item data from .aux file

Conversion and default tables.

```

1047 local ITEM_DATA_CONVERSIONS = {
1048     [KEY_ITEMNO] = tonumber,
1049     [KEY_ABSPAGENO] = tonumber,
1050     [KEY_PAGENO] = tonumber,
1051     [KEY_XPOS] = tonumber,
1052     [KEY_YPOS] = tonumber,
1053     [KEY_HEIGHT] = tonumber,
1054     [KEY_DEPTH] = tonumber,
1055     [KEY_TYPE] = tonumber,
1056     [KEY_POS] = tonumber,
1057     [KEY_COLUMN] = tonumber,
1058     [KEY_YSHIFT] = tonumber,
1059     [KEY_YSEP_ABOVE] = tonumber,
1060     [KEY_YSEP_BELOW] = tonumber,
1061     [KEY_YSEP_PAGE_TOP] = tonumber,
1062     [KEY_YSEP_PAGE_BOTTOM] = tonumber,
1063     [KEY_CHECKED] = toboolean,
1064 }
1065 local ITEM_DATA_DEFAULTS = {
1066     [KEY_ITEMNO] = 0,
1067     [KEY_ABSPAGENO] = 1,
1068     [KEY_PAGENO] = 1,
1069     [KEY_XPOS] = 0,
1070     [KEY_YPOS] = 0,
1071     [KEY_HEIGHT] = 0,
1072     [KEY_DEPTH] = 0,
1073     [KEY_TYPE] = 0,
1074     [KEY_POS] = 0,
1075     [KEY_COLUMN] = -1,
1076     [KEY_YSHIFT] = 0,
1077     [KEY_YSEP_ABOVE] = tex.dimen['marginparpush'],
1078     [KEY_YSEP_BELOW] = tex.dimen['marginparpush'],
1079     [KEY_YSEP_PAGE_TOP] = tex.dimen['marginparpush'],
1080     [KEY_YSEP_PAGE_BOTTOM] = tex.dimen['marginparpush'],
1081     [KEY_COLNO_COMPUTED] = 0,
1082     [KEY_XSHIFT_COMPUTED] = 0,
1083     [KEY_YSHIFT_COMPUTED] = 0,
1084     [KEY_SIDE_COMPUTED] = 0,
1085     [KEY_MARGINNO_COMPUTED] = 0,
1086     [KEY_ENABLED_COMPUTED] = true,
1087     [KEY_CHECKED] = false,

```

```
1088 }
```

ITEM_DATA_DEFAULTS is also used by load_item_data when no stored item data is found in ITEM_DATA_MAIN_TABLE.

store_item_data Store item_data supplied by keyvalue_string in ITEM_DATA_MAIN_TABLE.

```
1089 local function store_item_data(keyvalue_string)
1090
1091     local item = parse_data(keyvalue_string,
1092                             ITEM_DATA_CONVERSIONS,
1093                             ITEM_DATA_DEFAULTS)
1094
1095     ITEM_DATA_MAIN_TABLE[item[KEY_ITEMNO]] = item
1096
1097 end
```

(End of definition for store_item_data.)

check_item_data Check whether item_data supplied by keyvalue_string differs from that in ITEM_DATA_MAIN_TABLE, appending reports to ITEM_CHANGE_REPORT_TABLE if so.

```
1098 local function check_item_data(keyvalue_string)
1099
1100     check_data(keyvalue_string,
1101                ITEM_DATA_CONVERSIONS, ITEM_DATA_DEFAULTS,
1102                ITEM_DATA_MAIN_TABLE, KEY_ITEMNO,
1103                ITEM_CHANGE_REPORT_TABLE)
1104
1105 end
```

(End of definition for check_item_data.)

12.10 Writing reports

write_report If report_table is non-empty, write the data contained in it as a module warning. Use at most max_length items. text is written before the report. If max_length is non-zero, after_items_text is written after the items. noun refers to the kind of items.

```
1106 local function write_report(report_table,max_length,text,after_items_text,noun)
1107
1108     if #report_table == 0 then
1109         return
1110     end
1111
1112     if max_length == 0 then
1113         warning(text)
1114         return
1115     end
1116
1117     local report_text
1118     local report_length
1119
1120     if #report_table <= max_length then
1121         report_length = #report_table
1122         report_text = ' Here are the ' .. noun .. ':\n'
1123     else
```

```

1124     report_length = max_length
1125     report_text = ' Here are the first ' .. report_length .. ' ' .. noun .. ':\n'
1126 end
1127
1128 for i=1,report_length do
1129     report_text = report_text .. report_table[i]
1130     if i < report_length then
1131         report_text = report_text .. '\n'
1132     end
1133 end
1134
1135 warning(text .. ' ' .. report_text .. '\n' .. after_items_text)
1136
1137 end

```

(End of definition for write_report.)

`write_problem_report` Write a report about placement problems to T_EX in a format suitable for a package warning.

```

1138 local function write_problem_report()
1139
1140     write_report(
1141         PROBLEM_REPORT_TABLE,
1142         tex.count['l__marginalia_max_problem_int'],
1143         'Problems in placement.',
1144         'Problems listed at \\end{document}',
1145         'problems'
1146     )
1147
1148
1149 end

```

(End of definition for write_problem_report.)

`write_item_change_report` Write a report about changes in item data to T_EX in a format suitable for a package warning.

```

1150 local function write_item_change_report()
1151
1152     check_removed_data(ITEM_DATA_MAIN_TABLE,ITEM_CHANGE_REPORT_TABLE)
1153     write_report(
1154         ITEM_CHANGE_REPORT_TABLE,
1155         tex.count['l__marginalia_max_item_data_change_int'],
1156         'Changes in item data. Rerun to get correct placement.',
1157         'Changes listed at \\end{document}',
1158         'changes'
1159     )
1160
1161 end

```

(End of definition for write_item_change_report.)

`write_page_change_report` Write a report about changes in page data to T_EX in a format suitable for a package warning.

```

1162 local function write_page_change_report()

```

```

1163
1164   check_removed_data(PAGE_DATA_MAIN_TABLE,PAGE_CHANGE_REPORT_TABLE)
1165   write_report(
1166     PAGE_CHANGE_REPORT_TABLE,
1167     tex.count['l__marginalia_max_page_data_change_int'],
1168     'Changes in page data. Rerun to get correct placement.',
1169     'Changes listed at \\end{document}',
1170     'changes'
1171   )
1172
1173 end

```

(End of definition for write_page_change_report.)

12.11 Computing horizontal positions

It is necessary to determine whether an item should be placed on the right or left of the text block, and in which column it lies. The following lookup tables are used.

The value found in `RIGHTSIDE_LOOKUP_TABLE` is either `true` (right) or `false` (left). It is indexed by whether the binding is on the left (`true/false`), whether the item is on a recto page (`true/false`), whether it pertains to single-column text, the left column, or the right column (0/1/2), and the value of `pos` being either `auto` or `reverse`.

`RIGHTSIDE_LOOKUP_TABLE` is only used for `pos=auto`, `reverse`, `outer`, `inner` and the result reflects binding-on-the-left.

```

1174 local RIGHTSIDE_LOOKUP_TABLE = {
1175   [true] = {
1176     [true] = {
1177       [0] = {
1178         [POS_AUTO]=true,  [POS_REVERSE]=false, [POS_OUTER]=true,  [POS_INNER]=false,
1179       },
1180       [1] = {
1181         [POS_AUTO]=false, [POS_REVERSE]=true,  [POS_OUTER]=true,  [POS_INNER]=false,
1182       },
1183       [2] = {
1184         [POS_AUTO]=true,  [POS_REVERSE]=false, [POS_OUTER]=true,  [POS_INNER]=false,
1185       },
1186     },
1187     [false] = {
1188       [0] = {
1189         [POS_AUTO]=false, [POS_REVERSE]=true,  [POS_OUTER]=false, [POS_INNER]=true,
1190       },
1191       [1] = {
1192         [POS_AUTO]=false, [POS_REVERSE]=true,  [POS_OUTER]=false, [POS_INNER]=true,
1193       },
1194       [2] = {
1195         [POS_AUTO]=true,  [POS_REVERSE]=false, [POS_OUTER]=false, [POS_INNER]=true,
1196       },
1197     },
1198   },
1199   [false] = {
1200     [true] = {
1201       [0] = {
1202         [POS_AUTO]=false, [POS_REVERSE]=true,  [POS_OUTER]=false, [POS_INNER]=true,

```

```

1203     },
1204     [1] = {
1205         [POS_AUTO]=false, [POS_REVERSE]=true,  [POS_OUTER]=false, [POS_INNER]=true,
1206     },
1207     [2] = {
1208         [POS_AUTO]=true,  [POS_REVERSE]=false, [POS_OUTER]=false, [POS_INNER]=true,
1209     },
1210 },
1211 [false] = {
1212     [0] = {
1213         [POS_AUTO]=true,  [POS_REVERSE]=false, [POS_OUTER]=true,  [POS_INNER]=false,
1214     },
1215     [1] = {
1216         [POS_AUTO]=false, [POS_REVERSE]=true,  [POS_OUTER]=true,  [POS_INNER]=false,
1217     },
1218     [2] = {
1219         [POS_AUTO]=true,  [POS_REVERSE]=false, [POS_OUTER]=true,  [POS_INNER]=false,
1220     },
1221 },
1222 },
1223 }

```

The value found in `MARGINNO_LOOKUP_TABLE` ranges from 0 to 5 (see `KEY_MARGINNO_COMPUTED` for the meaning of these values). It is indexed by whether the item is on a recto page (`true/false`), whether it pertains to single-column text, the left column, or the right column (0/1/2), and whether it is to be placed on the right of the text block (`true/false`).

```

1224 local MARGINNO_LOOKUP_TABLE = {
1225     [true] = {
1226         [0] = {
1227             [false] = 1,
1228             [true] = 0,
1229         },
1230         [1] = {
1231             [false] = 1,
1232             [true] = 5,
1233         },
1234         [2] = {
1235             [false] = 4,
1236             [true] = 0,
1237         },
1238     },
1239     [false] = {
1240         [0] = {
1241             [false] = 3,
1242             [true] = 2,
1243         },
1244         [1] = {
1245             [false] = 3,
1246             [true] = 5,
1247         },
1248         [2] = {
1249             [false] = 4,
1250             [true] = 2,

```

```

1251     },
1252   },
1253 }

```

`compute_items_horizontal` For every `item_data` in `item_data_list`, compute the fields relevant to horizontal positioning, namely `KEY_COLNO_COMPUTED`, `KEY_XSHIFT_COMPUTED`, `KEY_SIDE_COMPUTED`, based on the layout information in `page_data`. Every item described in `item_data_list` is assumed to be on the same page.

```

1254 local function compute_items_horizontal(item_data_list,page_data)

```

Immediately return if `item_data_list` is empty, to avoid edge cases.

```

1255   if #item_data_list == 0 then
1256     return
1257   end

```

Information used frequently and which is the same for every item.

```

1258   local pageno = item_data_list[1][KEY_PAGENO]
1259   local twoside = page_data[KEY_TWOSIDE]
1260   local recto = ((pageno % 2) == 1) or (not twoside)
1261   local columncount = page_data[KEY_COLUMNCOUNT]

```

Tables to contain the x -coordinates of left edge, right edge, and middle of the current text, whether a single column (index 0), the left column (index 1), or the right column (index 2).

```

1262   local x_textleft = {}
1263   local x_textright = {}
1264   local x_textmiddle = {}

```

First, compute necessary dimensions for single-column text, since most of these calculations would be used anyway for two-column text. The terms used in calculating `x_textleft[0]` respectively take one to the origin of `\hoffset`, to the origin of `\oddsidemargin` and `\evensidemargin`, and to the left-hand side of the text block.

```

1265   if recto then
1266     x_textleft[0] = (
1267       page_data[KEY_HOFFSETORIGIN]
1268       + page_data[KEY_HOFFSET]
1269       + page_data[KEY_ODDSIDEMARGIN]
1270     )
1271     x_textright[0] = (
1272       x_textleft[0]
1273       + page_data[KEY_TEXTWIDTH]
1274     )
1275   else
1276     x_textleft[0] = (
1277       page_data[KEY_HOFFSETORIGIN]
1278       + page_data[KEY_HOFFSET]
1279       + page_data[KEY_EVENSIDEMARGIN]
1280     )
1281     x_textright[0] = (
1282       x_textleft[0]
1283       + page_data[KEY_TEXTWIDTH]
1284     )
1285   end
1286   x_textmiddle[0] = (x_textleft[0] + x_textright[0])/2
1287

```

```

1288
1289   if columncount == 1 then

```

If the page is one-column, the field KEY_COLNO_COMPUTED can be set immediately for every item_data.

```

1290       for i=1,#item_data_list do
1291           item_data_list[i][KEY_COLNO_COMPUTED] = 0
1292       end
1293   else

```

If the page is two-column, calculate the *x*-coordinates of the left and right edges and the mid-point of each column.

```

1294       x_textleft[1] = x_textleft[0]
1295       x_textright[1] = (
1296           x_textleft[1]
1297           + page_data[KEY_COLUMNWIDTH]
1298       )
1299       x_textmiddle[1] = (x_textleft[1] + x_textright[1])/2
1300
1301       x_textleft[2] = (
1302           x_textright[1]
1303           + page_data[KEY_COLUMNSEP]
1304       )
1305       x_textright[2] = (
1306           x_textleft[2]
1307           + page_data[KEY_COLUMNWIDTH]
1308       )
1309       x_textmiddle[2] = (x_textleft[2] + x_textright[2])/2
1310

```

Calculate the cut-off (mid-way between the columns) that distinguishes items from left and right columns.

```

1311       local left_column_x_limit = (
1312           x_textright[1]
1313           + .5*page_data[KEY_COLUMNSEP]
1314       )

```

Now set the field KEY_COLNO_COMPUTED for each item.

```

1315       for i=1,#item_data_list do
1316           local item_data = item_data_list[i]
1317
1318           if item_data[KEY_COLUMN] >= 0 then
1319               item_data[KEY_COLNO_COMPUTED] = item_data[KEY_COLUMN]
1320           else
1321               if item_data[KEY_XPOS] <= left_column_x_limit then
1322                   item_data[KEY_COLNO_COMPUTED] = 1
1323               else
1324                   item_data[KEY_COLNO_COMPUTED] = 2
1325               end
1326           end
1327       end
1328
1329   end

```

For every item_data in item_data_list, compute and set the fields KEY_SIDE_COMPUTED, KEY_XSHIFT_COMPUTED, and KEY_MARGINNO_COMPUTED.

```

1330 for i=1,#item_data_list do
1331     local item = item_data_list[i]
1332
1333     local pos = item[KEY_POS]
1334     local colnocomputed = item[KEY_COLNO_COMPUTED]
1335
1336     if pos == POS_LEFT then
1337         rightside = false
1338     elseif pos == POS_RIGHT then
1339         rightside = true
1340     elseif pos == POS_NEAREST then
1341         rightside = (item[KEY_XPOS] >= x_textmiddle[colnocomputed])
1342     else
(In this case, pos must be POS_AUTO, POS_REVERSE, POS_OUTER, or POS_INNER.)
1343         rightside = RIGHTSIDE_LOOKUP_TABLE[binding_left][recto][colnocomputed][pos]
1344     end
1345
1346     local marginno = MARGINNO_LOOKUP_TABLE[recto][colnocomputed][rightside]
1347
1348     if rightside then
1349         item[KEY_SIDE_COMPUTED] = 0
1350         item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
1351                                     + x_textright[colnocomputed]
1352     else
1353         item[KEY_SIDE_COMPUTED] = 1
1354         item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
1355                                     + x_textleft[colnocomputed]
1356     end
1357     item[KEY_MARGINNO_COMPUTED] = marginno
1358
1359 end
1360
1361 end

```

(End of definition for compute_items_horizontal.)

12.12 Computing vertical positions

12.12.1 Auxiliary functions

`get_y_item_top` Return the y -coordinate of the top of the item described by `item_data`.

```

1362 local function get_y_item_top(item_data)
1363     return item_data[KEY_YPOS]
1364           + item_data[KEY_YSHIFT_COMPUTED]
1365           + item_data[KEY_HEIGHT]
1366 end

```

(End of definition for get_y_item_top.)

`get_y_item_bottom` Return the y -coordinate of the bottom of the item described by `item_data`.

```

1367 local function get_y_item_bottom(item_data)
1368     return item_data[KEY_YPOS]
1369           - item_data[KEY_DEPTH]
1370           + item_data[KEY_YSHIFT_COMPUTED]
1371 end

```

(End of definition for `get_y_item_bottom`.)

`get_ysep_list` Calculate the separation to be used between adjacent marginal content items as described in `item_data_list`. The list is assumed to be sorted so that items are in the order they should appear on the page, top to bottom.

The idea is that we have the following arrangement for $i = 1, \dots, \#item_data_list$:

```

:
item_data_list[i]
ysep_list[i]
item_data_list[i+1]
:

```

Also set `ysep_list[0]` and `ysep_list[#item_data_list]` to 0, to avoid checking when these values are accessed (although they are not used).

```

1372 local function get_ysep_list(item_data_list)
1373
1374   local ysep_list = {}
1375
1376   ysep_list[0] = 0
1377   for i=1,#item_data_list-1 do
1378     ysep_list[i] = math.max(
1379       item_data_list[i][KEY_YSEP_BELOW],
1380       item_data_list[i+1][KEY_YSEP_ABOVE]
1381     )
1382   end
1383   ysep_list[#item_data_list] = 0
1384
1385   return ysep_list
1386
1387 end

```

(End of definition for `get_ysep_list`.)

12.12.2 Computing optfixed enabled

`compute_items_vertical_optfixed_enabled` For every `item_data` in `item_data_list` describing an item of type `TYPE_OPTFIXED`, check for a clash with an item of type `TYPE_FIXED`. If so, set `item_data[KEY_ENABLED_COMPUTED]` to false. Every item described in `item_data_list` is assumed to be on the same page and to have `KEY_YSHIFT` set to the default.

```

1388 local function compute_items_vertical_optfixed_enabled(item_data_list)
1389
1390   local optfixed_item_data_list = {}
1391   local fixed_item_data_list = {}
1392
1393   for _,item_data in pairs(item_data_list) do
1394     if item_data[KEY_TYPE] == TYPE_OPTFIXED then
1395       optfixed_item_data_list[#optfixed_item_data_list+1] = item_data
1396     elseif item_data[KEY_TYPE] == TYPE_FIXED then
1397       fixed_item_data_list[#fixed_item_data_list+1] = item_data
1398     end
1399   end
1400
1401   for _,optfixed_item_data in pairs(optfixed_item_data_list) do

```

```

1402     local optfixed_y_item_top = get_y_item_top(optfixed_item_data)
1403     local optfixed_y_item_bottom = get_y_item_bottom(optfixed_item_data)
1404
1405     for _,fixed_item_data in pairs(fixed_item_data_list) do
1406         local fixed_y_item_top = get_y_item_top(fixed_item_data)
1407         local fixed_y_item_bottom = get_y_item_bottom(fixed_item_data)
1408
1409         if (
1410             (
1411                 (fixed_y_item_bottom - optfixed_y_item_top)
1412                 <
1413                 math.max(
1414                     fixed_item_data[KEY_YSEP_BELOW],
1415                     optfixed_item_data[KEY_YSEP_ABOVE]
1416                 )
1417             )
1418             and
1419             (
1420                 (optfixed_y_item_bottom - fixed_y_item_top)
1421                 <
1422                 math.max(
1423                     optfixed_item_data[KEY_YSEP_BELOW],
1424                     fixed_item_data[KEY_YSEP_ABOVE]
1425                 )
1426             )
1427         ) then
1428             optfixed_item_data[KEY_ENABLED_COMPUTED] = false
1429             break
1430         end
1431     end
1432 end
1433
1434 end

```

(End of definition for `compute_items_vertical_optfixed_enabled`.)

12.12.3 Computing vertical adjustment

`compute_items_vertical_adjustment`

For every `item_data` in `item_data_list`, compute the field relevant to vertical positioning, namely `KEY_YSHIFT_COMPUTED`, based on the layout information in `page_data`. Every item described in `item_data_list` is assumed to be on the same page and to have `KEY_YSHIFT` set to the default, and the list is assumed to be sorted so that items are in the order they should appear on the page, top to bottom.

```

1435 local function compute_items_vertical_adjustment(item_data_list,page_data)

```

Immediately return if `item_data_list` is empty, to avoid edge cases

```

1436     if #item_data_list == 0 then
1437         return
1438     end

```

```

1439
1440     local ysep_list = get_ysep_list(item_data_list)

```

First pass of computation (downward). `y_limit_above` will always be the highest `y`-coordinate at which the top of next item below can appear.

```

1441 local y_limit_above = (
1442   page_data[KEY_VOFFSET]
1443   + page_data[KEY_PAGEHEIGHT]
1444   - item_data_list[1][KEY_YSEP_PAGE_TOP]
1445 )
1446
1447 for i=1,#item_data_list do
1448   local item_data = item_data_list[i]
1449
1450   local y_item_top = get_y_item_top(item_data)
1451
1452   if y_item_top > y_limit_above then
1453     if item_data[KEY_TYPE] == TYPE_NORMAL then
1454       item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1455                                     + (y_limit_above - y_item_top)
1456     end
1457   end
1458
1459   y_limit_above = get_y_item_bottom(item_data) - ysep_list[i]
1460 end

```

Second pass of computation (upward). y_limit_below will always be the lowest y -coordinate at which the bottom of next item above can appear.

```

1461 local y_limit_below = (
1462   page_data[KEY_VOFFSET]
1463   + item_data_list[#item_data_list][KEY_YSEP_PAGE_BOTTOM]
1464 )
1465
1466 for i=#item_data_list,1,-1 do
1467   local item_data = item_data_list[i]
1468
1469   local y_item_bottom = get_y_item_bottom(item_data)
1470
1471   if y_item_bottom < y_limit_below then
1472     if item_data[KEY_TYPE] == TYPE_NORMAL then
1473       item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1474                                     + (y_limit_below - y_item_bottom)
1475     end
1476   end
1477
1478   y_limit_below = get_y_item_top(item_data) + ysep_list[i-1]
1479 end
1480
1481 end

```

(End of definition for compute_items_vertical_adjustment.)

12.12.4 Checking vertical adjustment

Messages to use when checking results of vertical adjustment.

```

1482 local ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES = {
1483   [TYPE_NORMAL] = 'Moveable item > ysep page top',
1484   [TYPE_FIXED] = 'Topmost fixed item > ysep page top',
1485   [TYPE_OPTFIXED] = 'Topmost optfixed item > ysep page top',

```

```

1486 }
1487 local ITEM_CLASH_MESSAGES = {
1488   [TYPE_NORMAL] = {
1489     [TYPE_NORMAL] = 'moveable items'
1490     .. ' (this shouldn\'t happen)',
1491     [TYPE_FIXED] = 'moveable item above fixed item',
1492     [TYPE_OPTFIXED] = 'moveable item above optfixed item',
1493   },
1494   [TYPE_FIXED] = {
1495     [TYPE_NORMAL] = 'moveable item below fixed item',
1496     [TYPE_FIXED] = 'fixed items',
1497     [TYPE_OPTFIXED] = 'fixed item above optfixed item '
1498     .. ' (this shouldn\'t happen)',
1499   },
1500   [TYPE_OPTFIXED] = {
1501     [TYPE_NORMAL] = 'moveable items below optfixed item',
1502     [TYPE_FIXED] = 'fixed item below optfixed item '
1503     .. ' (this shouldn\'t happen)',
1504     [TYPE_OPTFIXED] = 'optfixed items '
1505     .. ' (this shouldn\'t happen)',
1506   },
1507 }
1508 local ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE = {
1509   [TYPE_NORMAL] = 'Moveable item < ysep page bottom',
1510   [TYPE_FIXED] = 'Bottommost fixed item < ysep page bottom',
1511   [TYPE_OPTFIXED] = 'Bottommost optfixed item < ysep page bottom',
1512 }

```

`check_items_vertical` For the items described by the `item_data` in `item_data_list`, check whether any clash or fail to obey `ysep page top` or `ysep page bottom`. If so, write messages to `PROBLEM_REPORT_TABLE`.

```

1513 local function check_items_vertical(item_data_list,page_data)

```

Immediately return if `item_data_list` is empty, to avoid edge cases

```

1514   if (#item_data_list) == 0 then
1515     return
1516   end
1517
1518   local ysep_list = get_ysep_list(item_data_list)
1519
1520   local item_data
1521

```

If any item fails to obey `ysep page top`, the first one in the list does.

```

1522   item_data = item_data_list[1]
1523   if (
1524     get_y_item_top(item_data) > page_data[KEY_VOFFSET]
1525     + page_data[KEY_PAGEHEIGHT]
1526     - item_data[KEY_YSEP_PAGE_TOP]
1527   ) then
1528     table.insert(
1529       PROBLEM_REPORT_TABLE,
1530       get_data_page_number(item_data)
1531       .. ' ' .. ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES[item_data[KEY_TYPE]]
1532     )

```

```

1533 end
1534
1535 for i=2,#item_data_list do
1536     local item_data = item_data_list[i]
1537     local prev_item_data = item_data_list[i-1]
1538     if (
1539         get_y_item_top(item_data) > get_y_item_bottom(prev_item_data)
1540             - ysep_list[i-1]
1541     ) then
1542         table.insert(
1543             PROBLEM_REPORT_TABLE,
1544             get_data_page_number(item_data)
1545             .. ' Clash: ' ..
1546             ITEM_CLASH_MESSAGES[prev_item_data[KEY_TYPE]][item_data[KEY_TYPE]]
1547         )
1548     end
1549 end

```

If any item fails to obey ysep page bottom, the last one in the list does.

```

1550 item_data = item_data_list[#item_data_list]
1551 if (
1552     get_y_item_bottom(item_data) < page_data[KEY_VOFFSET]
1553         + item_data[KEY_YSEP_PAGE_BOTTOM]
1554 ) then
1555     table.insert(
1556         PROBLEM_REPORT_TABLE,
1557         get_data_page_number(item_data)
1558         .. ' ' .. ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE[item_data[KEY_TYPE]]
1559     )
1560 end
1561
1562 end

```

(End of definition for check_items_vertical.)

12.12.5 Core vertical position computation

`compute_items_vertical` For every `item_data` in `item_data_list`, compute the field relevant to vertical positioning, namely `KEY_YSHIFT_COMPUTED`, based on the layout information in `page_data`. This may involve setting the field `KEY_ENABLED_COMPUTED` to false. In such a case, the relevant `item_data` is removed from `item_data_list`.

```

1563 local function compute_items_vertical(item_data_list,page_data)

```

Set `KEY_YSHIFT_COMPUTED` of each `item_data` to the user-supplied value.

```

1564     for i=1,#item_data_list do
1565         local item_data = item_data_list[i]
1566
1567         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT]
1568     end

```

Decide which items of `TYPE_OPTFIXED` are to be disabled.

```

1569     compute_items_vertical_optfixed_enabled(item_data_list)

```

Strip any `item_data` with `KEY_ENABLED_COMPUTED` set to false from `item_data_list`.

```

1570 list_filter(item_data_list,function(item_data)
1571     return item_data[KEY_ENABLED_COMPUTED]
1572 end)

```

Sort `item_data_list` according to the stored position from top to bottom and left to right on the page, resolving ties using `KEY_ITEMNO`.

```

1573 table.sort(
1574     item_data_list,
1575     function(left,right)
1576         local y_diff = left[KEY_YPOS] - right[KEY_YPOS]
1577
1578         if y_diff > 0 then
1579             return true
1580         elseif y_diff < 0 then
1581             return false
1582         end
1583
1584         local x_diff = left[KEY_XPOS] - right[KEY_XPOS]
1585
1586         if x_diff < 0 then
1587             return true
1588         elseif x_diff > 0 then
1589             return false
1590         end
1591
1592         return (left[KEY_ITEMNO] < right[KEY_ITEMNO])
1593     end
1594 )
1595
1596 compute_items_vertical_adjustment(item_data_list,page_data)
1597
1598 check_items_vertical(item_data_list,page_data)
1599
1600 end

```

(End of definition for compute_items_vertical.)

`compute_items` For every item represented in `ITEM_DATA_MAIN_TABLE`, use the `page_data` stored in `PAGE_DATA_MAIN_TABLE` to compute the `item_data` values necessary to place the item correctly on the page, namely those indexed by: `KEY_COLNO_COMPUTED`, `KEY_XSHIFT_COMPUTED`, `KEY_YSHIFT_COMPUTED`, `KEY_SIDE_COMPUTED`, `KEY_ENABLED_COMPUTED`.

```

1601 local function compute_items()

```

Compute the maximum `abspageno`, which will be the last page of the document on which a item appears.

```

1602 local max_abspageno = 0
1603
1604 for k,v in pairs(ITEM_DATA_MAIN_TABLE) do
1605     max_abspageno = math.max(v[KEY_ABSPAGENO],max_abspageno)
1606 end

```

`per_abspage_item_data_list` will be a list indexed by absolute page numbers. Each entry will be a list (possibly empty) of `item_data` describing the items that appear on the corresponding page.

```
1607 local per_abspage_item_data_list = {}
```

Prepare `per_abspage_item_data_list` by making each entry an empty list, then fill it from `ITEM_DATA_MAIN_TABLE`.

```
1608 for i=1,max_abspageno do
1609     per_abspage_item_data_list[i] = {}
1610 end
1611 for _,item_data in pairs(ITEM_DATA_MAIN_TABLE) do
1612     local temp_table = per_abspage_item_data_list[item_data[KEY_ABSPAGENO]]
1613     temp_table[#temp_table+1] = item_data
1614 end
```

`per_abspage_item_data_list` will be a list indexed by absolute page numbers. Each entry will be a `page_data` describing the corresponding page. Usually multiple entries will be the same `page_data`: in the loop, `pagedatano` will be the index of the last entry in `PAGE_DATA_MAIN_TABLE` with `KEY_ABSPAGENO` value less than or equal to `abspageno`. (There may be several such entries in `PAGE_DATA_MAIN_TABLE` because `\marginalianewgeometry` may have been called multiple times on the same page.) Note that `PAGE_DATA_MAIN_TABLE[0]` is available even if there was no data in the `.aux` file, because the defaults were stored by `store_default_page_data`.

```
1615 local per_abspage_page_data_list = {}
1616 local pagedatano = 0
1617 for abspageno = 1,max_abspageno do
1618     while (
1619         PAGE_DATA_MAIN_TABLE[pagedatano+1] ~= nil
1620         and
1621         PAGE_DATA_MAIN_TABLE[pagedatano+1][KEY_ABSPAGENO] == abspageno
1622     ) do
1623         pagedatano = pagedatano+1
1624     end
1625     per_abspage_page_data_list[abspageno] = PAGE_DATA_MAIN_TABLE[pagedatano]
1626 end
```

Iterate through all pages and perform the necessary computations.

```
1627 for abspageno=1,#per_abspage_item_data_list do
1628     local current_page_data = per_abspage_page_data_list[abspageno]
1629     local current_page_item_data_list = per_abspage_item_data_list[abspageno]
```

First, compute the horizontal positions, which includes sorting items into columns in two-column mode.

```
1630     compute_items_horizontal(current_page_item_data_list,current_page_data)
```

Sort the items into sublists corresponding to the margins in which they are located.

```
1631     local current_page_item_data_sublists = {}
1632
1633     for i=0,5 do
1634         current_page_item_data_sublists[i] = {}
1635     end
1636
1637     for _,item_data in pairs(current_page_item_data_list) do
1638         table.insert(
1639             current_page_item_data_sublists[item_data[KEY_MARGINNO_COMPUTED]],
1640             item_data
1641         )
1642     end
```

Compute vertical positons for each sublist.

```

1643     for i=0,5 do
1644         compute_items_vertical(
1645             current_page_item_data_sublists[i],
1646             current_page_data
1647         )
1648     end
1649 end
1650 end

```

(End of definition for compute_items.)

12.13 Passing item_data back to L^AT_EX

`load_item_data` Set the relevant L^AT_EX counter and dimension variables to the values computed for `itemno`.

```

1651 local function load_item_data(itemno)
1652
1653     item = ITEM_DATA_MAIN_TABLE[tonumber(itemno)]
1654     if item == nil then
1655         item = ITEM_DATA_DEFAULTS
1656     end
1657
1658     tex.count['l__marginalia_page_int'] = item[KEY_PAGENO]
1659     tex.count['l__marginalia_column_computed_int'] = item[KEY_COLNO_COMPUTED]
1660     tex.dimen['l__marginalia_xshift_computed_dim'] = item[KEY_XSHIFT_COMPUTED]
1661     tex.dimen['l__marginalia_yshift_computed_dim'] = item[KEY_YSHIFT_COMPUTED]
1662     tex.count['l__marginalia_side_computed_int'] = item[KEY_SIDE_COMPUTED]
1663     tex.count['l__marginalia_marginno_computed_int']
1664         = item[KEY_MARGINNO_COMPUTED]
1665     if item[KEY_ENABLED_COMPUTED] then
1666         tex.count['l__marginalia_enabled_computed_int'] = 1
1667     else
1668         tex.count['l__marginalia_enabled_computed_int'] = 0
1669     end
1670
1671 end

```

(End of definition for load_item_data.)

12.14 Export public functions

Finally, make available the functions that will be called from L^AT_EX using `\lua_now:n` and `\lua_now:e`.

```

1672 marginalia = marginalia or {}
1673 marginalia.store_default_page_data = store_default_page_data
1674 marginalia.store_page_data = store_page_data
1675 marginalia.check_page_data = check_page_data
1676
1677 marginalia.store_item_data = store_item_data
1678 marginalia.check_item_data = check_item_data
1679
1680 marginalia.compute_items = compute_items

```

```
1681
1682 marginalia.load_item_data = load_item_data
1683
1684 marginalia.write_problem_report = write_problem_report
1685
1686 marginalia.write_page_change_report = write_page_change_report
1687 marginalia.write_item_change_report = write_item_change_report
1688
1689 marginalia.set_binding_left = set_binding_left
1690 </lua>
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\'	1490, 1498, 1503, 1505
\\	1144, 1157, 1169
B	
\baselineskip	44, 796
\begin	7, 11, 14, 26, 27
binding side (option)	7
bool commands:	
\bool_if:N _{TF}	94
\bool_new:N	126, 791
\bool_set:Nn	794
\bool_set_false:N	132
\bool_set_true:N	131
\bool_to_str:N	74
\bool_to_str:n	546
box commands:	
\box_dp:N	628, 757, 767, 769
\box_ht:N	626, 757, 768
\box_move_up:nn	775
\box_new:N	559
\box_set_dp:Nn	759, 769
\box_set_ht:Nn	758, 768
\box_use:N	777
\l_tmpa_box	765, 767
C	
check commands:	
check_data	941
check_item_data	1098
check_items_vertical	1513
check_page_data	1039
check_removed_data	974
column (option)	10
\columnsep	545
\columnwidth	544
compute commands:	
compute_items	1601
compute_items_horizontal	1254
compute_items_vertical	1563
compute_items_vertical_adjustment	1435
compute_items_vertical_optfixed-enabled	1388
cs commands:	
\cs_new:Nn	114
\cs_new:Npn	14, 26, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 83, 247, 251, 476, 478, 485, 505, 517, 527, 569, 682, 754, 761, 771, 781, 785, 792
\cs_set:Npn	92, 98
\cs_set_eq:NN	87, 90, 105, 110, 121, 124, 482, 489, 495, 498, 501, 526, 553, 578, 587, 592, 599, 600, 601, 602, 615, 616, 688, 690, 692, 694, 699, 701, 703, 705, 710, 712, 714, 716, 721, 723, 725, 727, 732, 734, 736, 738, 743, 745, 747, 749
\currentgrouplevel	78
D	
dim commands:	
\dim_new:N	165, 166, 167, 168, 169, 170, 256, 257, 258, 259, 296, 297, 298, 299, 300, 301, 560, 561, 564, 565
\dim_set:Nn	26, 118, 124, 199, 206, 213, 220, 334, 341, 348, 355, 625, 627, 757, 767
\dim_set_eq:NN	612, 613
\l_tmpa_dim	757, 758, 759, 767, 768, 769
E	
\end	18
\evensidemargin	58, 541
exp commands:	
\exp_not:N	634, 635, 637, 638
G	
get commands:	
get_data_page_number	908
get_y_item_bottom	1367
get_y_item_top	1362
get_ysep_list	1372
group commands:	
\group_begin:	572, 618
\group_end:	622, 680
H	
\hbox	41
hbox commands:	
\hbox_overlap_left:n	667, 672
\hbox_overlap_right:n	656, 660
\hbox_to_zero:n	652
\headheight	13, 249, 253
\headsep	13, 249, 253
\hoffset	58, 537

hook commands:

\hook_gput_code:nnn 88, 100,
108, 116, 122, 137, 492, 497, 514, 550
\hsize 14, 40, 612, 613

I

\ignorespaces 619
info 824

int commands:

\int_case:nn 597, 684
\int_compare:nNnTF 130
\int_eval:n 536, 634
\int_gincr:N 529, 571
\int_if_zero:nTF ... 78, 507, 650, 654
\int_new:N 141, 149, 157,
234, 525, 558, 562, 563, 566, 567, 568
\int_set:Nn 145, 153, 161, 531, 532
\int_set_eq:NN 238
\int_value:w 535, 537, 538,
539, 540, 541, 542, 543, 544, 545,
575, 633, 635, 636, 637, 638, 639,
640, 641, 642, 643, 644, 645, 646, 647
\l_tmpa_int 531, 532, 543

iow commands:

\iow_now:Nn 519, 533
\iow_shipout_e:Nn 631

K

\kern 657, 662, 669, 674

keys commands:

\l_keys_choice_int
130, 145, 153, 161, 238
\keys_define:nn 127,
142, 150, 158, 171, 195, 235, 242,
260, 302, 330, 369, 387, 418, 436, 467
\keys_set:nn 80, 85, 573

L

\lastxpos 637
\lastypos 638
\leavevmode 584

legacy commands:

\legacy_if:nTF 530, 583
\legacy_if_gset:nn 797
\legacy_if_p:n 546, 794
\linewidth 40, 613

list commands:

list_filter 888, 905
\llap 4, 16

load commands:

load_item_data 1651

lua commands:

\lua_now:n 24, 68, 31,
34, 38, 42, 46, 50, 54, 58, 62, 66, 70, 74

M

\marginalia 4, 5, 7–11, 15,
16, 18, 20–22, 25, 33, 36, 38, 48, 799

marginalia internal commands:

\l_marginalia_aux_iow
495, 519, 533, 631
\l_marginalia_binding_left_bool
..... 25–27, 94, 126, 139
\l_marginalia_column_computed_-
int 39, 563, 616
\l_marginalia_column_int . 157, 642
\l_marginalia_default_yshift_-
dim 242, 643
_marginalia_dim_set:Nn
26, 121, 121, 124, 174,
176, 178, 180, 182, 184, 263, 265,
267, 269, 305, 307, 309, 311, 313, 315
_marginalia_dim_set_deferred:Nn
..... 26, 114, 114, 121
\l_marginalia_enabled_computed_-
int 39, 568, 650
\l_marginalia_item_box
40, 41, 44, 559, 606, 626, 628, 777
_marginalia_item_box_set:Nn ..
599, 600, 601, 602, 606
\l_marginalia_item_depth_dim ..
560, 627, 640
\l_marginalia_item_height_dim .
560, 625, 639
\g_marginalia_itemno_int
36, 507, 558, 571, 575, 633
_marginalia_lua_call_boolean:nN
..... 72, 72, 138
_marginalia_lua_check_item_-
data:n 36, 32, 52, 503
_marginalia_lua_check_page_-
data:n 36, 32, 40, 500
_marginalia_lua_compute_items:
..... 32, 56, 494
_marginalia_lua_load_item_-
data:n 38, 68, 68, 574
_marginalia_lua_store_default_-
page_data: 32, 32, 493
_marginalia_lua_store_item_-
data:n 35, 32, 48, 491
_marginalia_lua_store_page_-
data:n 35, 32, 36, 484
_marginalia_lua_write_item_-
change_report: 32, 64, 510
_marginalia_lua_write_page_-
change_report: 44, 511
_marginalia_lua_write_problem_-
report: 32, 60, 509

`\_marginalia_margin_bottom:` ...
 ... [247](#), [251](#), [276](#), [294](#)
`\_marginalia_margin_top:` ...
 ... [247](#), [247](#), [272](#), [293](#)
`\l_marginalia_marginno_computed-`
 `int` ... [42](#), [567](#), [684](#)
`\l_marginalia_mark_left-`
 `between_tl` ... [418](#), [750](#)
`\l_marginalia_mark_recto_left-`
 `tl` ... [418](#), [706](#)
`\l_marginalia_mark_recto_right-`
 `tl` ... [418](#), [695](#)
`\l_marginalia_mark_right-`
 `between_tl` ... [418](#), [739](#)
`\l_marginalia_mark_tl` . [658](#), [660](#),
 [670](#), [672](#), [694](#), [705](#), [716](#), [727](#), [738](#), [749](#)
`\l_marginalia_mark_verso_left-`
 `tl` ... [418](#), [728](#)
`\l_marginalia_mark_verso_right-`
 `tl` ... [418](#), [717](#)
`\l_marginalia_max_item_data-`
 `change_int` ... [467](#)
`\l_marginalia_max_page_data-`
 `change_int` ... [467](#)
`\l_marginalia_max_problem_int` . [467](#)
`\l_marginalia_nobreak_bool` ...
 ... [791](#), [794](#), [797](#)
`\_marginalia_outer_inner_set:Nn` [26](#)
`\_marginalia_outer_inner-`
 `set:NNNn` ... [105](#), [106](#), [111](#), [198](#),
 [205](#), [212](#), [219](#), [333](#), [340](#), [347](#), [354](#),
 [390](#), [397](#), [404](#), [411](#), [439](#), [446](#), [453](#), [460](#)
`\_marginalia_outer_inner_set-`
 `deferred:Nn` ... [26](#)
`\_marginalia_outer_inner_set-`
 `deferred:NNNn` ... [98](#), [98](#), [107](#)
`\_marginalia_outer_inner_set-`
 `immediate:Nn` ... [26](#)
`\_marginalia_outer_inner_set-`
 `immediate:NNNn` ... [92](#), [92](#), [102](#), [112](#)
`\l_marginalia_page_int` . [39](#), [562](#), [615](#)
`\g_marginalia_pagedatano_int` ..
 ... [525](#), [529](#), [535](#)
`\_marginalia_place_item_box:` ..
 ... [41](#), [663](#), [668](#), [771](#), [771](#)
`\l_marginalia_pos_int` [149](#), [641](#)
`\_marginalia_process_item:nn` ..
 ... [42](#), [45](#), [569](#), [569](#), [801](#)
`\_marginalia_process_item-`
 `data:n` ... [35](#), [36](#), [487](#), [490](#), [502](#)
`\_marginalia_process_page-`
 `data:n` ... [35](#), [36](#), [480](#), [483](#), [499](#)
`\_marginalia_set_xsep_width-`
 `style_mark:` ... [604](#), [682](#), [682](#)
`\_marginalia_setup:n`
 ... [25](#), [45](#), [87](#), [87](#), [90](#), [805](#)
`\_marginalia_setup_body:n`
 ... [25](#), [83](#), [83](#), [90](#)
`\_marginalia_setup_preamble:n` .
 ... [25](#), [76](#), [76](#), [87](#)
`\l_marginalia_side_computed_int`
 ... [39](#), [41](#), [566](#), [654](#)
`\l_marginalia_style_left-`
 `between_tl` ... [369](#), [748](#)
`\l_marginalia_style_recto_left-`
 `tl` ... [369](#), [704](#)
`\l_marginalia_style_recto-`
 `right_tl` ... [369](#), [693](#)
`\l_marginalia_style_right-`
 `between_tl` ... [369](#), [737](#)
`\l_marginalia_style_tl`
 . [42](#), [611](#), [692](#), [703](#), [714](#), [725](#), [736](#), [747](#)
`\l_marginalia_style_verso_left-`
 `tl` ... [369](#), [726](#)
`\l_marginalia_style_verso-`
 `right_tl` ... [369](#), [715](#)
`\_marginalia_tagging_socket:n` .
 ... [23](#), [14](#), [26](#), [605](#), [608](#), [624](#)
`\l_marginalia_type_int` ... [141](#), [636](#)
`\_marginalia_typeset:n`
 ... [579](#), [588](#), [593](#), [629](#)
`\_marginalia_typeset_hmmode:n` . [781](#)
`\_marginalia_typeset_hmode:n` ..
 ... [589](#), [785](#)
`\_marginalia_typeset_mmode:n` ..
 ... [580](#), [781](#), [781](#)
`\_marginalia_typeset_vmode:n` ..
 ... [594](#), [781](#), [792](#)
`\l_marginalia_valign_int` [40](#), [234](#), [597](#)
`\_marginalia_vbox_set_center:Nn`
 ... [601](#), [754](#), [754](#)
`\_marginalia_vbox_set_midway:Nn`
 ... [602](#), [761](#), [761](#)
`\l_marginalia_width_dim` [40](#),
 [42](#), [612](#), [690](#), [701](#), [712](#), [723](#), [734](#), [745](#)
`\l_marginalia_width_left-`
 `between_dim` ... [296](#), [746](#)
`\l_marginalia_width_recto_left-`
 `dim` ... [296](#), [702](#)
`\l_marginalia_width_recto-`
 `right_dim` ... [296](#), [691](#)
`\l_marginalia_width_right-`
 `between_dim` ... [296](#), [735](#)
`\l_marginalia_width_verso_left-`
 `dim` ... [296](#), [724](#)
`\l_marginalia_width_verso-`
 `right_dim` ... [296](#), [713](#)

<code>\vbox</code>	44		
<code>vbox</code> commands:			
<code>\vbox_set:Nn</code>	600, 765		
<code>\vbox_set_top:Nn</code>	599, 756, 763		
<code>\voffset</code>	13, 249, 253, 538		
<code>\vtop</code>	44		
W			
<code>warning</code>	824		
<code>width (option)</code>	14		
<code>width between (option)</code>	14		
<code>width inner (option)</code>	14		
<code>width left between (option)</code>	14		
<code>width outer (option)</code>	14		
<code>width recto inner (option)</code>	14		
<code>width recto left (option)</code>	14		
<code>width recto outer (option)</code>	14		
<code>width recto right (option)</code>	14		
<code>width right between (option)</code>	14		
<code>width verso inner (option)</code>	14		
<code>width verso left (option)</code>	14		
<code>width verso outer (option)</code>	14		
<code>width verso right (option)</code>	14		
<code>write</code> commands:			
<code>write_item_change_report</code>	1150		
<code>write_page_change_report</code>	1162		
<code>write_problem_report</code>	1138		
<code>write_report</code>	1106		
		X	
		<code>xsep (option)</code>	10
		<code>xsep between (option)</code>	10
		<code>xsep inner (option)</code>	10
		<code>xsep left between (option)</code>	10
		<code>xsep outer (option)</code>	10
		<code>xsep recto inner (option)</code>	10
		<code>xsep recto left (option)</code>	10
		<code>xsep recto outer (option)</code>	10
		<code>xsep recto right (option)</code>	10
		<code>xsep right between (option)</code>	10
		<code>xsep verso inner (option)</code>	10
		<code>xsep verso left (option)</code>	10
		<code>xsep verso outer (option)</code>	10
		<code>xsep verso right (option)</code>	10
		Y	
		<code>ysep (option)</code>	11
		<code>ysep above (option)</code>	11
		<code>ysep below (option)</code>	11
		<code>ysep page bottom (option)</code>	11
		<code>ysep page bottom margin (option)</code>	13
		<code>ysep page top (option)</code>	11
		<code>ysep page top bottom margin (option)</code> .	13
		<code>ysep page top margin (option)</code>	13
		<code>yshift (option)</code>	11