

# marginalia — Marginal content anywhere with automatic adjustment for Lua $\text{\LaTeX}$ <sup>1</sup>

<sup>1</sup> This document describes v0.83.23, last revised 2026-07-08.

<sup>2</sup> a.j.cain (AT) gmail.com

Alan J. Cain<sup>2</sup>

Released 2026-07-08

## Abstract

This Lua $\text{\LaTeX}$  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.

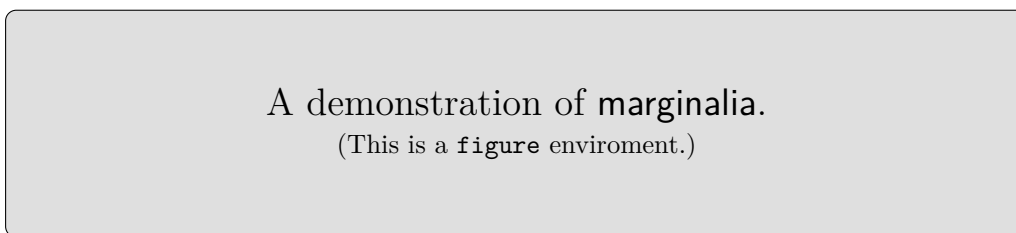
(Like this one.)

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.

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.

Notes can be placed on both sides of the paragraph.

**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.)



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.

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

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.

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).

# Contents

<b>Demonstration</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
<b>2 Requirements</b>	<b>4</b>
<b>3 Installation</b>	<b>4</b>
<b>4 Getting started</b>	<b>5</b>
<b>5 User commands</b>	<b>6</b>
5.1 Access to page and column . . . . .	6
<b>6 Options</b>	<b>7</b>
6.1 Type . . . . .	7
6.2 Horizontal placement . . . . .	8
6.3 Vertical placement . . . . .	9
6.4 Appearance . . . . .	11
<b>7 Placement</b>	<b>14</b>
7.1 Horizontal placement . . . . .	14
7.2 Vertical placement . . . . .	16
<b>8 Usage notes</b>	<b>17</b>
<b>9 Incompatibilities</b>	<b>18</b>
<b>10 Limitations</b>	<b>18</b>
<b>11 Implementation (L<sup>A</sup>T<sub>E</sub>X package)</b>	<b>19</b>
11.1 Initial set-up . . . . .	19
11.2 Tagging set-up . . . . .	19
11.3 Auxiliary macro for dimension setting . . . . .	19
11.4 Auxiliary macros for setting options . . . . .	20
11.5 Options . . . . .	20
11.5.1 Type . . . . .	21
11.5.2 Horizontal placement . . . . .	21
11.5.3 Vertical placement . . . . .	22
11.5.4 Appearance . . . . .	24
11.6 Lua backend and interface . . . . .	26
11.7 Processing data from the .aux file . . . . .	27
11.8 Writing version data to the .aux file . . . . .	29
11.9 Writing page data to the .aux file . . . . .	30
11.10 Marginal content item processing . . . . .	31
11.10.1 Variables . . . . .	31
Variables set by L <sup>A</sup> T <sub>E</sub> X. . . . .	31
Variables set by Lua. . . . .	31
11.10.2 Core macro . . . . .	32
11.10.3 Horizontal separation, width, style, mark selection . . . . .	35

11.10.4	Auxiliary box macros	36
11.10.5	Placement macros	37
11.11	User commands	38
<b>12</b>	<b>Implementation (Lua backend)</b>	<b>38</b>
12.1	Global variables	38
12.2	Constants	39
12.3	Keys for tables	39
12.3.1	Keys for both page and item data tables	39
12.3.2	Keys for page data tables, layout etc.	39
12.3.3	Keys for item data tables	40
12.4	Utility functions	41
12.5	Generic page/item data functions	42
12.6	Processing of page data from <code>.aux</code> file	43
12.7	Processing of item data from <code>.aux</code> file	45
12.8	Writing reports	46
12.9	Computing horizontal positions	48
12.10	Computing vertical positions	52
12.10.1	Auxiliary functions	52
12.10.2	Computing <code>optfixed</code> enabled	53
12.10.3	Computing vertical adjustment	54
12.10.4	Checking vertical adjustment	55
12.10.5	Core vertical position computation	57
12.11	Passing <code>item_data</code> back to $\LaTeX$	59
12.12	Export public functions	60

## Index 61

### 1 Introduction

The  $\LaTeX$  `\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  $\LaTeX$  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. (Bringhurst 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,  $\LaTeX$  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  $\TeX$ ’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`<sup>3</sup> (Tom Sgouros & Stefan Ulrich), `marginnote`<sup>4</sup> (Markus Kohm), `marginfix`<sup>5</sup> (Stephen Hicks) and `marginfit`<sup>6</sup> (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

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

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

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

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

## Installation

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 Lua $\LaTeX$  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 Lua $\LaTeX$  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].<sup>7</sup> Thus the basic functionality has been tested extensively, and it has performed correctly.

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

URL: [https://archive.org/details/cain\\_formand\\_number\\_ebook\\_large](https://archive.org/details/cain_formand_number_ebook_large)

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

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

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

**Licence.** `marginalia` is released under the  $\LaTeX$  Project Public Licence v1.3c or later.<sup>8</sup>

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

**Translation.** A French translation of the documentation has been made by members of GUTenberg (Le Groupe francophone des Utilisateurs de  $\TeX$ ).<sup>9</sup>

**Feature requests and bug reports** The development code and issue tracker are hosted at Codeberg.<sup>10</sup>

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

## 2 Requirements

`marginalia` requires

- (1) Lua $\LaTeX$ ,
- (2) a recent  $\LaTeX$  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 $\LaTeX$  can find them.

## Getting started

```
\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}
```

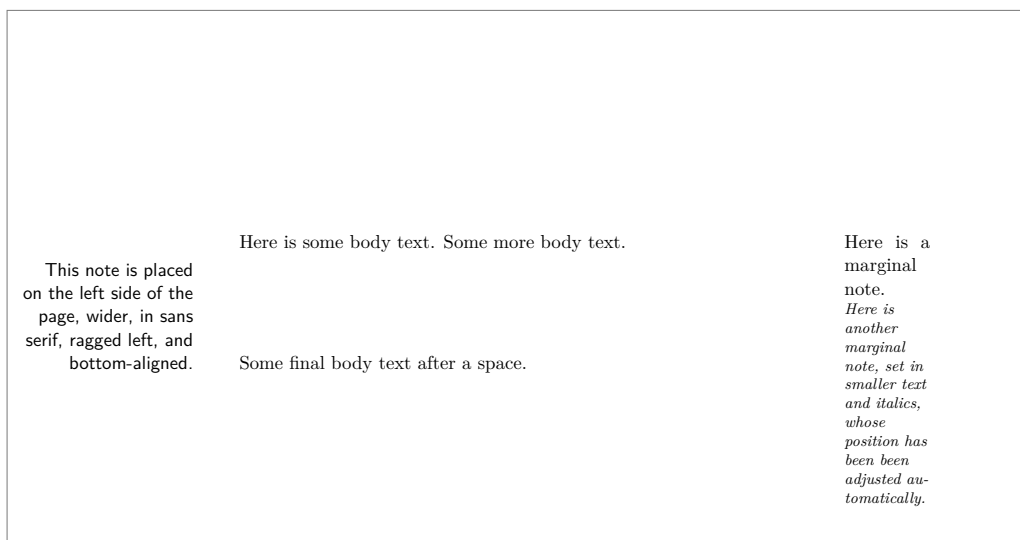


Figure 4.1: A small demonstration of marginalia.

## 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 `LuaATEX` for the marginal content to be placed correctly.* (See [Section 8](#) 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

---

`\marginalia` `\marginalia[options]{content}`

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](#).

---

`\marginaliasetup` `\marginaliasetup{options}`

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.

---

`\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,<sup>11</sup> or by using the  $\text{\LaTeX}$  command `\twocolumn` 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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  run and will default to 0.

## 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</code>	Dimension	<code>\marginparsep</code>
<code>xsep outer</code>	Dimension	<code>\marginparsep</code>
<code>xsep inner</code>	Dimension	<code>\marginparsep</code>
<code>xsep 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 right between</code>	Dimension	<code>\marginparsep</code>
<code>xsep left between</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]	—

## 6 Options

The description of keys in this section, which are summarized in Tables 1 and 2 (on pages 7 and 13 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 Type

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

## Options

**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`)

## 6.2 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 four 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`.

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

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

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

(Default: `auto`)

`column` 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 overflow `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`)

### Options

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](#).

<code>xsep</code>	<b>xsep recto outer:</b> used for an item in the outer margin of a recto page.
<code>xsep outer</code>	<b>xsep recto inner:</b> used for an item in the inner margin of a recto page.
<code>xsep inner</code>	<b>xsep verso outer:</b> used for an item in the outer margin of a verso page.
<code>xsep between</code>	<b>xsep verso inner:</b> used for an item in the inner margin of a verso page.
<code>xsep recto outer</code>	<b>xsep right between:</b> used for an item set from the right column between the columns.
<code>xsep recto inner</code>	
<code>xsep verso outer</code>	<b>xsep left between:</b> used for an item set from the left column between the columns.
<code>xsep verso inner</code>	<b>xsep outer:</b> a shorthand for setting the keys <code>xsep recto outer</code> and <code>xsep verso outer</code> simultaneously to the same value.
<code>xsep right between</code>	<b>xsep inner:</b> a shorthand for setting the keys <code>xsep recto inner</code> and <code>xsep verso inner</code> simultaneously to the same value.
<code>xsep left between</code>	<b>xsep between:</b> a shorthand for setting the keys <code>xsep right between</code> and <code>xsep left between</code> simultaneously to the same value.
	<b>xsep:</b> a shorthand for setting all of these keys simultaneously.

(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 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}`)

### 6.3 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.

## 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}
```

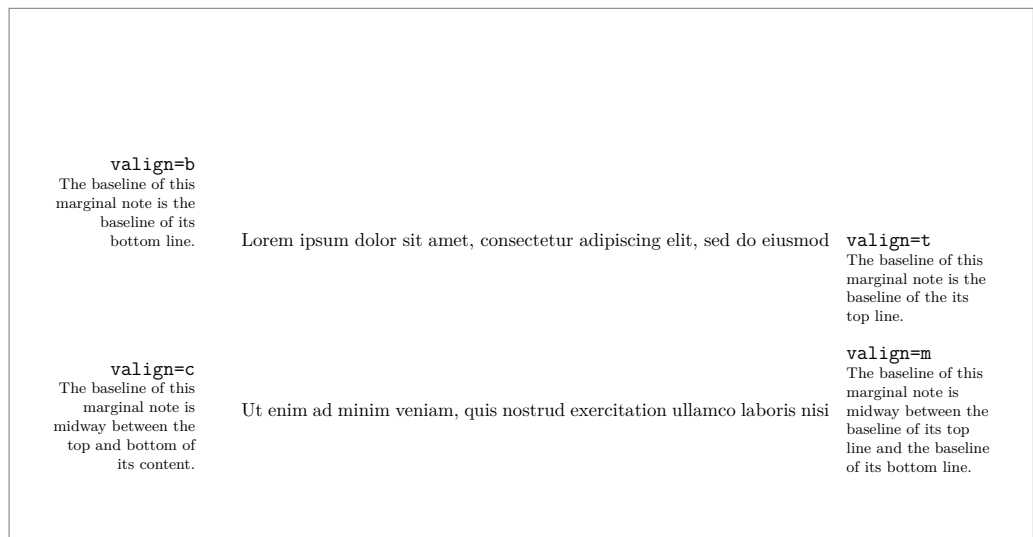


Figure 6.1: A demonstration of the various values of `valign`.

### Options

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.

`yshift` 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.

`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` **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 top bottom margin` **ysep page bottom margin:** Automatically set `ysep page bottom` to match the margins below 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.

## 6.4 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.

## Options

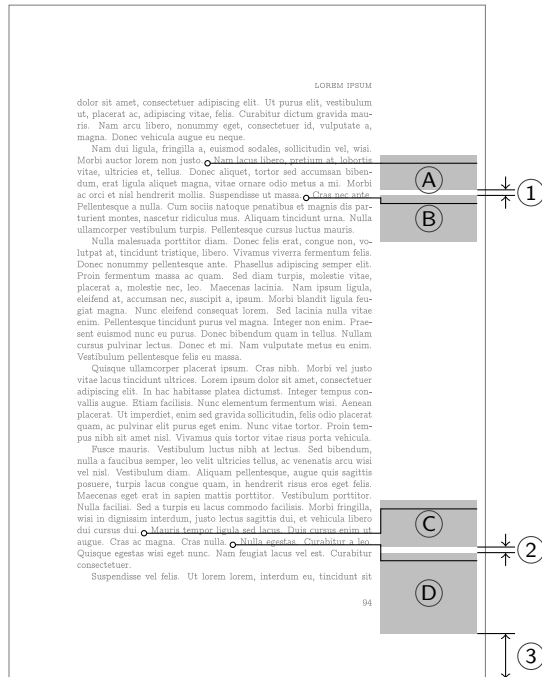


Figure 6.2: (Illustration of `ysep`) The length ① is at least the value of `ysep` below specified (locally or globally) for marginal content item (A) and at least the value of `ysep` above specified for item (B). In this example diagram, (B) has been automatically moved down from its natural position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

<code>width</code>	These keys specify the width of the n item of marginal content (or, more precisely,
<code>width outer</code>	the <code>\hsize</code> of the box into which the item is typeset). Which width is chosen will depend
<code>width inner</code>	on the where the item is typeset. The terminology is as in <a href="#">Figure 7.1</a> .
<code>width between</code>	<b>width recto outer:</b> used for an item in the outer margin of a recto page.
<code>width recto outer</code>	<b>width recto inner:</b> used for an item in the inner margin of a recto page.
<code>width recto inner</code>	<b>width verso outer:</b> used for an item in the outer margin of a verso page.
<code>width verso outer</code>	<b>width verso inner:</b> used for an item in the inner margin of a verso page.
<code>width verso inner</code>	<b>width right between:</b> used for an item set from the right column and placed be-
<code>width right between</code>	tween the columns.
<code>width left between</code>	<b>width left between:</b> used for an item set from the right column and placed between
	the columns.
	<b>width outer:</b> a shorthand for setting the keys <code>width recto outer</code> and <code>width verso</code>
	<code>outer</code> simultaneously to the same value.
	<b>width inner:</b> a shorthand for setting the keys <code>width recto inner</code> and <code>width verso</code>
	<code>inner</code> simultaneously to the same value.
	<b>width between:</b> a shorthand for setting the keys <code>width right between</code> and <code>width</code>
	<code>left between</code> simultaneously to the same value.
	<b>width:</b> a shorthand for setting all of these keys simultaneously.

## 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</code>	Dimension	<code>\marginparwidth</code>
<code>width outer</code>	Dimension	<code>\marginparwidth</code>
<code>width inner</code>	Dimension	<code>\marginparwidth</code>
<code>width 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 right between</code>	Dimension	<code>\marginparwidth</code>
<code>width left between</code>	Dimension	<code>\marginparwidth</code>
<code>style</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>style recto outer</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>style recto inner</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>style verso outer</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>style verso inner</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>style right between</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>style left between</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark recto outer</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark recto inner</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark verso outer</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark verso inner</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark right between</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]
<code>mark left between</code>	L <sup>A</sup> T <sub>E</sub> X code	[Empty]

(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. (*Default:* value of `\marginparwidth` at `\begin{document}`})

`style` These keys specify the style with which an item of marginal content is typeset.

`style recto outer` Which style is chosen will depend on where the item is typeset. The terminology is as in [Figure 7.1.](#)

`style recto inner`

`style verso outer` **style recto outer:** used for an item in the outer margin of a recto page.

`style verso inner` **style recto inner:** used for an item in the inner margin of a recto page.

`style right between` **style verso outer:** used for an item in the outer margin of a verso page.

`style left between` **style verso inner:** used for an item in the inner margin of a verso page.

**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:** a shorthand for setting all of these keys simultaneously.

*Placement* Each of these keys should be set to L<sup>A</sup>T<sub>E</sub>X code that specifies the style. (*Default:* [Empty])

These keys specify code to typeset something alongside the line where the call to `\marginalia` appears, on the same side on which the marginal content is placed. Roughly speaking, they can be set to any L<sup>A</sup>T<sub>E</sub>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 `\rlap` 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 `\llap` 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](#)

The terminology for the various `mark` options is as in [Figure 7.1](#).

**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 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:** a shorthand for setting all of these keys simultaneously.

Each of these keys should be set to L<sup>A</sup>T<sub>E</sub>X code. (*Default:* [Empty])

## 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 one-column mode, marginal content is placed by default in the outer margin: right on recto pages, left on verso pages. If `pos=reverse` is applied, it is placed in the inner margin: left on recto pages, right on verso pages.

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: on a recto page, the inner margin, on a verso page, the outer margin. 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: on a recto page, the outer margin, on a verso page, the inner margin. If `pos=reverse` is applied, it is placed between the two columns, adjacent to the right column.

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

`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`.

## Placement

```

\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.)} consecetur 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}

```

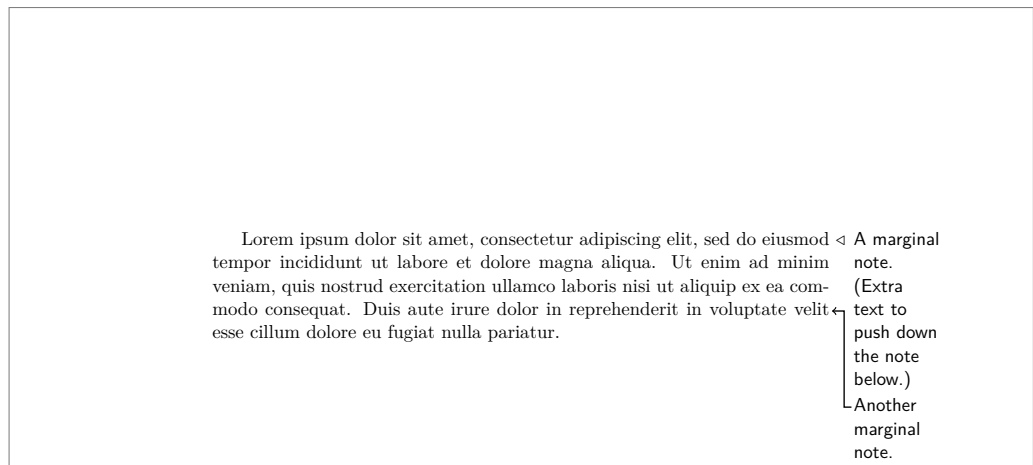


Figure 6.3: A demonstration of the `mark` option, in particular showing how it can be used to draw a TikZ arrow from a (moved) note to the relevant line. Notes: (1) The code in the marginal content is processed *before* the mark, so the `mark` TikZ code refers to a coordinate defined in the marginal 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

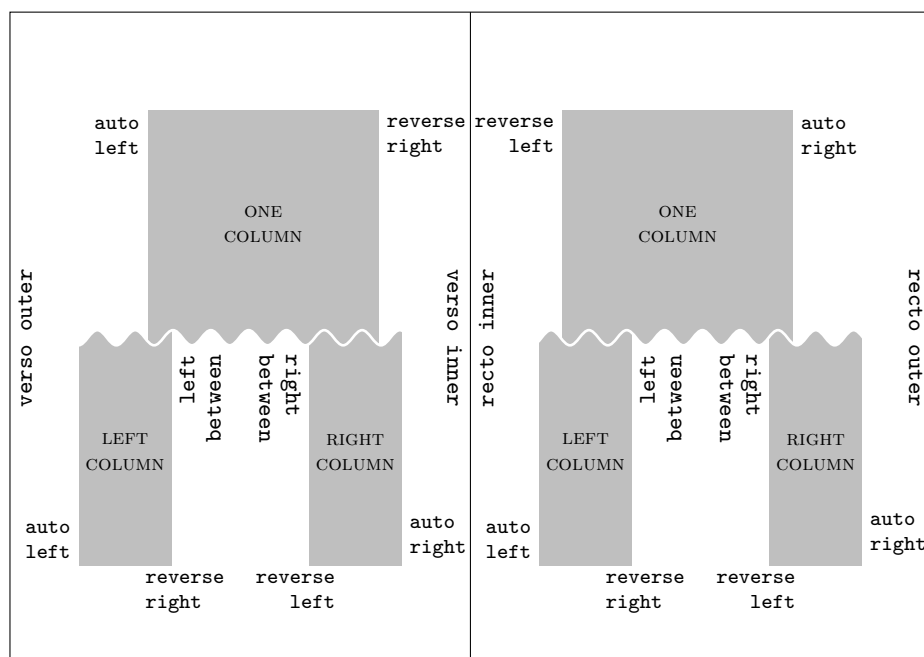


Figure 7.1: Summary of the positioning of marginal content using `pos`, and terminology used in `width` and `style` keys, on recto and verso pages, in both one-column and two-column mode.

`marginalia` determines in which column the call to `\marginalia` was made using its horizontal position. As discussed in the description of 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 the 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.

## Usage notes

- 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.)
- 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.)
- 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 $\LaTeX$  run.

## 8 Usage notes

`marginalia` requires a minimum of two Lua $\LaTeX$  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.

At the end of the Lua $\LaTeX$  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 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.

## 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 `\marginpar`.

## 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](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<sup>A</sup>T<sub>E</sub>X package)

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

### 11.1 Initial set-up

Package identification/version information.

```
3 \NeedsTeXFormat{LaTeX2e}[2020-02-02]
4 \ProvidesExplPackage{marginalia}{2026-07-08}{0.83.23}
5 {Non-floating marginal content for LuaLaTeX}
```

Check that Lua<sub>T</sub>E<sub>X</sub> 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<sup>A</sup>T<sub>E</sub>X 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 Auxiliary macro for dimension setting

`\__marginalia_set_dim:Nn` Set the dimension variable passed as the first parameter to value specified in second parameter at `begindocument` if used in the preamble, or immediately (since `begindocument` is a one-time hook) in the document.

```
31 \cs_new:Nn \__marginalia_set_dim:Nn
32 {
33   \hook_gput_code:nnn { begindocument } { marginalia/dim }
34   {
35     \dim_set:Nn #1 { #2 }
36   }
37 }
```

```

36     }
37 }

```

*(End of definition for `\_marginalia_set_dim:Nn`.)*

## 11.4 Auxiliary macros for setting options

`\_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 (via `\_marginalia_set_dim:Nn`) should have no effect. And since `\marginalia` cannot be used in the preamble, setting other options inside a group in the preamble is pointless.

```

38 \cs_new:Npn \_marginalia_setup_preamble:n #1
39 {
40   \int_if_zero:N\T{ \currentgrouplevel }
41   {
42     \keys_set:nn{marginalia}{ #1 }
43   }
44 }

```

*(End of definition for `\_marginalia_setup_preamble:n`.)*

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

```

45 \cs_new:Npn \_marginalia_setup_body:n #1
46 {
47   \keys_set:nn{marginalia}{ #1 }
48 }

```

*(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`.

```

49 \cs_set_eq:NN\_marginalia_setup:n\_marginalia_setup_preamble:n
50 \hook_gput_code:nnn{ begindocument/end }{ marginalia/setup }
51 {
52   \cs_set_eq:NN\_marginalia_setup:n\_marginalia_setup_body:n
53 }

```

*(End of definition for `\_marginalia_setup:n`.)*

## 11.5 Options

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

### 11.5.1 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.

```
54 \int_new:N\l__marginalia_type_int
55 \keys_define:nn { marginalia }
56 {
57   type .choices:nn = {normal,fixed,optfixed}{
58     \int_set:Nn\l__marginalia_type_int{\l_keys_choice_int}
59   },
60   type .initial:n = normal,
61 }
```

*(End of definition for \l\_\_marginalia\_type\_int.)*

### 11.5.2 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 = left, 4 = right, 5 = nearest.

```
62 \int_new:N\l__marginalia_pos_int
63 \keys_define:nn { marginalia }
64 {
65   pos .choices:nn = {auto,reverse,left,right,nearest}{
66     \int_set:Nn\l__marginalia_pos_int{\l_keys_choice_int}
67   },
68   pos .initial:n = auto
69 }
```

*(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).

```
70 \int_new:N\l__marginalia_column_int
71 \keys_define:nn { marginalia }
72 {
73   column .choices:nn = {auto,one,left,right}{
74     \int_set:Nn\l__marginalia_column_int{\l_keys_choice_int-2}
75   },
76   column .initial:n = auto,
77 }
```

*(End of definition for \l\_\_marginalia\_column\_int.)*

`\l__marginalia_xsep_recto_outer_dim`  
`\l__marginalia_xsep_recto_inner_dim`  
`\l__marginalia_xsep_verso_outer_dim`  
`\l__marginalia_xsep_verso_inner_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.

```
78 \dim_new:N\l__marginalia_xsep_recto_outer_dim
79 \dim_new:N\l__marginalia_xsep_recto_inner_dim
80 \dim_new:N\l__marginalia_xsep_verso_outer_dim
81 \dim_new:N\l__marginalia_xsep_verso_inner_dim
```

```

82 \dim_new:N\l__marginalia_xsep_right_between_dim
83 \dim_new:N\l__marginalia_xsep_left_between_dim
84 \keys_define:nn { marginalia }
85 {
86   xsep~recto~outer .code:n
87     = \__marginalia_set_dim:Nn\l__marginalia_xsep_recto_outer_dim{#1},
88   xsep~recto~inner .code:n
89     = \__marginalia_set_dim:Nn\l__marginalia_xsep_recto_inner_dim{#1},
90   xsep~verso~outer .code:n
91     = \__marginalia_set_dim:Nn\l__marginalia_xsep_verso_outer_dim{#1},
92   xsep~verso~inner .code:n
93     = \__marginalia_set_dim:Nn\l__marginalia_xsep_verso_inner_dim{#1},
94   xsep~right~between .code:n
95     = \__marginalia_set_dim:Nn\l__marginalia_xsep_right_between_dim{#1},
96   xsep~left~between .code:n
97     = \__marginalia_set_dim:Nn\l__marginalia_xsep_left_between_dim{#1},
98   xsep .code:n = {
99     \keys_set:nn{ marginalia }{
100       xsep~recto~outer=#1,
101       xsep~recto~inner=#1,
102       xsep~verso~outer=#1,
103       xsep~verso~inner=#1,
104       xsep~right~between=#1,
105       xsep~left~between=#1,
106     }
107   },
108   xsep~outer .code:n = {
109     \keys_set:nn{ marginalia }{
110       xsep~recto~outer=#1,
111       xsep~verso~outer=#1,
112     }
113   },
114   xsep~inner .code:n = {
115     \keys_set:nn{ marginalia }{
116       xsep~recto~inner=#1,
117       xsep~verso~inner=#1,
118     }
119   },
120   xsep~between .code:n = {
121     \keys_set:nn{ marginalia }{
122       xsep~right~between=#1,
123       xsep~left~between=#1,
124     }
125   },
126   xsep .initial:n = \marginparsep,
127 }

```

*(End of definition for \l\_\_marginalia\_xsep\_recto\_outer\_dim and others.)*

### 11.5.3 Vertical placement

`\l__marginalia_valign_int` A key to store the vertical alignment of the marginal content item. The setting is held in an 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).

```

128 \int_new:N\l__marginalia_valign_int
129 \keys_define:nn { marginalia }
130 {
131   valign .choices:nn = {t,b,c,m}{
132     \int_set_eq:NN\l__marginalia_valign_int\l_keys_choice_int
133   },
134   valign .initial:n = t,
135 }

```

*(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.

```

136 \keys_define:nn { marginalia }
137 {
138   yshift .dim_set:N = \l__marginalia_default_yshift_dim,
139   yshift .initial:n = 0pt,
140 }

```

*(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.

```

141 \cs_new:Npn \__marginalia_margin_top:
142   {
143     1in + \voffset + \topmargin + \headheight + \headsep
144   }
145 \cs_new:Npn \__marginalia_margin_bottom:
146   {
147     \pageheight - 1in - \voffset - \topmargin - \headheight - \headsep
148     - \textheight
149   }

```

*(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.

```

150 \dim_new:N\l__marginalia_ysep_above_dim
151 \dim_new:N\l__marginalia_ysep_below_dim
152 \dim_new:N\l__marginalia_ysep_page_top_dim
153 \dim_new:N\l__marginalia_ysep_page_bottom_dim
154 \keys_define:nn { marginalia }
155 {
156   ysep~above .code:n
157     = \__marginalia_set_dim:Nn\l__marginalia_ysep_above_dim{#1},
158   ysep~below .code:n
159     = \__marginalia_set_dim:Nn\l__marginalia_ysep_below_dim{#1},
160   ysep~page~top .code:n
161     = \__marginalia_set_dim:Nn\l__marginalia_ysep_page_top_dim{#1},
162   ysep~page~bottom .code:n
163     = \__marginalia_set_dim:Nn\l__marginalia_ysep_page_bottom_dim{#1},
164   ysep~above~below .code:n = {

```

```

165     \keys_set:nn{ marginalia }{
166       ysep-below=#1,
167       ysep-above=#1,
168     }
169   },
170   ysep .code:n = {
171     \keys_set:nn{ marginalia }{
172       ysep-below=#1,
173       ysep-above=#1,
174       ysep-page-top=#1,
175       ysep-page-bottom=#1,
176     }
177   },
178   ysep-page-top-margin .code:n = {
179     \keys_set:nn{ marginalia }{
180       ysep-page-top
181       = \_marginalia_margin_top:
182     }
183   },
184   ysep-page-bottom-margin .code:n = {
185     \keys_set:nn{ marginalia }{
186       ysep-page-bottom
187       = \_marginalia_margin_bottom:
188     }
189   },
190   ysep-page-top-bottom-margin .code:n = {
191     \keys_set:nn{ marginalia }{
192       ysep-page-top-margin,
193       ysep-page-bottom-margin,
194     }
195   },
196   ysep-above-below .initial:n = \marginparpush,
197   ysep-page-top .initial:n = \_marginalia_margin_top:,
198   ysep-page-bottom .initial:n = \_marginalia_margin_bottom:,
199 }

```

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

#### 11.5.4 Appearance

`\l__marginalia_width_recto_outer_dim` 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_inner_dim
\l__marginalia_width_verso_outer_dim
\l__marginalia_width_verso_inner_dim
\l__marginalia_width_right_between_dim
\l__marginalia_width_left_between_dim
200 \dim_new:N\l__marginalia_width_recto_outer_dim
201 \dim_new:N\l__marginalia_width_recto_inner_dim
202 \dim_new:N\l__marginalia_width_verso_outer_dim
203 \dim_new:N\l__marginalia_width_verso_inner_dim
204 \dim_new:N\l__marginalia_width_right_between_dim
205 \dim_new:N\l__marginalia_width_left_between_dim
206 \keys_define:nn { marginalia }
207 {
208   width-recto-outer .code:n
209     = \_marginalia_set_dim:Nn\l__marginalia_width_recto_outer_dim{#1},
210   width-recto-inner .code:n
211     = \_marginalia_set_dim:Nn\l__marginalia_width_recto_inner_dim{#1},

```

```

212 width~verso~outer .code:n
213   = \__marginalia_set_dim:Nn\l__marginalia_width_verso_outer_dim{#1},
214 width~verso~inner .code:n
215   = \__marginalia_set_dim:Nn\l__marginalia_width_verso_inner_dim{#1},
216 width~right~between .code:n
217   = \__marginalia_set_dim:Nn\l__marginalia_width_right_between_dim{#1},
218 width~left~between .code:n
219   = \__marginalia_set_dim:Nn\l__marginalia_width_left_between_dim{#1},
220 width .code:n = {
221   \keys_set:nn{ marginalia }{
222     width~recto~outer=#1,
223     width~recto~inner=#1,
224     width~verso~outer=#1,
225     width~verso~inner=#1,
226     width~right~between=#1,
227     width~left~between=#1,
228   }
229 },
230 width~outer .code:n = {
231   \keys_set:nn{ marginalia }{
232     width~recto~outer=#1,
233     width~verso~outer=#1,
234   }
235 },
236 width~inner .code:n = {
237   \keys_set:nn{ marginalia }{
238     width~recto~inner=#1,
239     width~verso~inner=#1,
240   }
241 },
242 width~between .code:n = {
243   \keys_set:nn{ marginalia }{
244     width~right~between=#1,
245     width~left~between=#1,
246   }
247 },
248 width .initial:n = \marginparwidth,
249 }

```

(End of definition for \l\_\_marginalia\_width\_recto\_outer\_dim and others.)

```

\l__marginalia_style_recto_outer_tl
\l__marginalia_style_recto_inner_tl
\l__marginalia_style_verso_outer_tl
\l__marginalia_style_verso_inner_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.

```

250 \keys_define:nn { marginalia }
251 {
252   style~recto~outer .tl_set:N = \l__marginalia_style_recto_outer_tl,
253   style~recto~inner .tl_set:N = \l__marginalia_style_recto_inner_tl,
254   style~verso~outer .tl_set:N = \l__marginalia_style_verso_outer_tl,
255   style~verso~inner .tl_set:N = \l__marginalia_style_verso_inner_tl,
256   style~right~between .tl_set:N = \l__marginalia_style_right_between_tl,
257   style~left~between .tl_set:N = \l__marginalia_style_left_between_tl,
258   style .code:n = {
259     \keys_set:nn{ marginalia }{
260       style~recto~outer=#1,

```

```

261     style~recto~inner=#1,
262     style~verso~outer=#1,
263     style~verso~inner=#1,
264     style~right~between=#1,
265     style~left~between=#1,
266   }
267 },
268 style .initial:n = {},
269 }

```

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

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

```

270 \keys_define:nn { marginalia }
271 {
272   mark~recto~outer .tl_set:N = \l__marginalia_mark_recto_outer_tl,
273   mark~recto~inner .tl_set:N = \l__marginalia_mark_recto_inner_tl,
274   mark~verso~outer .tl_set:N = \l__marginalia_mark_verso_outer_tl,
275   mark~verso~inner .tl_set:N = \l__marginalia_mark_verso_inner_tl,
276   mark~right~between .tl_set:N = \l__marginalia_mark_right_between_tl,
277   mark~left~between .tl_set:N = \l__marginalia_mark_left_between_tl,
278   mark .code:n = {
279     \keys_set:nn{ marginalia }{
280       mark~recto~outer=#1,
281       mark~recto~inner=#1,
282       mark~verso~outer=#1,
283       mark~verso~inner=#1,
284       mark~right~between=#1,
285       mark~left~between=#1,
286     }
287   },
288   mark .initial:n = {},
289 }

```

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

## 11.6 Lua backend and interface

Load the Lua backend.

```

290 \lua_now:n{
291   marginalia = require('marginalia')
292 }

```

The following 9 macros interface between L<sup>A</sup>T<sub>E</sub>X 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  
`\__marginalia_lua_store_page_data:n` process data not containing control sequences (read from the `.aux` file); hence `\lua_`  
`\__marginalia_lua_check_page_data:n` `now:n` is used.  
`\__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:`

```

293 \cs_new:Npn\__marginalia_lua_store_default_page_data:
294 {
295   \lua_now:n{ marginalia.store_default_page_data() }
296 }

```

```

297 \cs_new:Npn\__marginalia_lua_store_page_data:n #1
298 {
299   \lua_now:n{ marginalia.store_page_data('#1') }
300 }
301 \cs_new:Npn\__marginalia_lua_check_page_data:n #1
302 {
303   \lua_now:n{ marginalia.check_page_data('#1') }
304 }
305 \cs_new:Npn\__marginalia_lua_write_page_change_report:
306 {
307   \lua_now:n{ marginalia.write_page_change_report() }
308 }
309 \cs_new:Npn\__marginalia_lua_store_item_data:n #1
310 {
311   \lua_now:n{ marginalia.store_item_data('#1') }
312 }
313 \cs_new:Npn\__marginalia_lua_check_item_data:n #1
314 {
315   \lua_now:n{ marginalia.check_item_data('#1') }
316 }
317 \cs_new:Npn\__marginalia_lua_compute_items:
318 {
319   \lua_now:n{ marginalia.compute_items() }
320 }
321 \cs_new:Npn\__marginalia_lua_write_problem_report:
322 {
323   \lua_now:n{ marginalia.write_problem_report() }
324 }
325 \cs_new:Npn\__marginalia_lua_write_item_change_report:
326 {
327   \lua_now:n{ marginalia.write_item_change_report() }
328 }

```

*(End of definition for \\_\_marginalia\_lua\_store\_default\_page\_data: and others.)*

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

```

329 \cs_new:Npn\__marginalia_lua_load_item_data:n #1
330 {
331   \lua_now:e{ marginalia.load_item_data('#1') }
332 }

```

*(End of definition for \\_\_marginalia\_lua\_load\_item\_data:n.)*

## 11.7 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.

```

333 \cs_new:Npn \marginalia@version #1
334 {}

```

*(End of definition for \marginalia@pagedata.)*

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

```
335 \cs_new:Npn \marginalia@pagedata #1
336 {
337   \__marginalia_process_page_data:n{#1}
338 }
```

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.

```
339 \cs_set_eq:NN
340   \__marginalia_process_page_data:n
341   \__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.

```
342 \cs_new:Npn \marginalia@itemdata #1
343 {
344   \__marginalia_process_item_data:n{#1}
345 }
```

*(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.

```
346 \cs_set_eq:NN
347   \__marginalia_process_item_data:n
348   \__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.

```
349 \hook_gput_code:nnn{ begindocument }{ marginalia/prepare }{
350   \__marginalia_lua_store_default_page_data:
351   \__marginalia_lua_compute_items:
352   \cs_set_eq:NN\l__marginalia_aux_iow\@mainaux
353 }
```

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.

```
354 \hook_gput_code:nnn{ enddocument/afterlastpage }{ marginalia/check }{
355   \cs_set_eq:NN
356     \__marginalia_process_page_data:n
357     \__marginalia_lua_check_page_data:n
358   \cs_set_eq:NN
359     \__marginalia_process_item_data:n
360     \__marginalia_lua_check_item_data:n
361 }
```

`\_marginalia_write_reports:` All the reports of changes and/or problems are assembled in the Lua backend. This macro will write the reports as package warnings, using the following three messages, to which the Lua-assembled reports are passed as parameters:

```

362 \msg_new:nnn{marginalia}{placement_problem}
363   { Problems~in~placement.~#1 }
364 \msg_new:nnn{marginalia}{item_change}
365   { Changes~in~item~data.~Rerun~to~get~correct~placement.~#1 }
366 \msg_new:nnn{marginalia}{page_change}
367   { Changes~in~page~data.~Rerun~to~get~correct~placement.~#1 }
368 \cs_new:Npn\_marginalia_write_reports:
369   {
370     \group_begin:
371     \tl_set:N\l_tmpa_tl{\_marginalia_lua_write_problem_report:}
372     \tl_if_blank:VF\l_tmpa_tl
373     {
374       \msg_warning:nne{marginalia}{placement_problem}{\tl_use:N\l_tmpa_tl}
375     }
376     \tl_set:N\l_tmpa_tl{\_marginalia_lua_write_item_change_report:}
377     \tl_if_blank:VF\l_tmpa_tl
378     {
379       \msg_warning:nne{marginalia}{item_change}{\tl_use:N\l_tmpa_tl}
380     }
381     \tl_set:N\l_tmpa_tl{\_marginalia_lua_write_page_change_report:}
382     \tl_if_blank:VF\l_tmpa_tl
383     {
384       \msg_warning:nne{marginalia}{page_change}{\tl_use:N\l_tmpa_tl}
385     }
386     \group_end:
387   }

```

*(End of definition for `\_marginalia_write_reports:`.)*

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

```

388 \hook_gput_code:nnn{ enddocument/info }{ marginalia/report } {
389   \_marginalia_write_reports:
390 }

```

## 11.8 Writing version data to the .aux file

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

```

391 \cs_new:Npn\_marginalia_write_version:
392   {
393     \iow_now:N\l_marginalia_aux_iow{
394       \token_to_str:N\marginalia@version{
395         \use:c{ver@marginalia.sty}
396       }
397     }
398   }

```

*(End of definition for `\_marginalia_write_version:`.)*

## 11.9 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.

```
399 \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`).

```
400 \cs_set_eq:NN\_marginalia_write_page_data:\prg_do_nothing:
401 \cs_new:Npn\_marginalia_write_page_data_real:
402 {
403   \int_gincr:N\g_marginalia_pagedatano_int
404   \legacy_if:nTF{@twocolumn}
405     { \int_set:Nn\l_tmpa_int{2} }
406     { \int_set:Nn\l_tmpa_int{1} }
407   \iow_now:Ne\l_marginalia_aux_iow{
408     \token_to_str:N\marginalia@pagedata{
409       pagedatano=\int_value:w\g_marginalia_pagedatano_int,
410       abspageno=\int_eval:n{\g_shipout_readonly_int+1},
411       hoffset=\int_value:w\hoffset,
412       voffset=\int_value:w\voffset,
413       pageheight=\int_value:w\pageheight,
414       oddsidemargin=\int_value:w\oddsidemargin,
415       evensidemargin=\int_value:w\evensidemargin,
416       textwidth=\int_value:w\textwidth,
417       columncount=\int_value:w\l_tmpa_int,
418       columnwidth=\int_value:w\columnwidth,
419       columnsep=\int_value:w\columnsep,
420       twoside=\bool_to_str:n{\legacy_if_p:n{@twoside}},
421     }
422   }
423 }
```

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.

```
424 \hook_gput_code:nnn{ begindocument/end }{ marginalia/initial }
425 {
426   \_marginalia_write_version:
427   \cs_set_eq:NN
428     \_marginalia_write_page_data:
429     \_marginalia_write_page_data_real:
430   \_marginalia_write_page_data:
431 }
```

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

## 11.10 Marginal content item processing

### 11.10.1 Variables

#### Variables set by L<sup>A</sup>T<sub>E</sub>X.

`\g__marginalia_itemno_int` Global integer variable to index marginal content items.  
432 \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.  
433 \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`  
434 \dim\_new:N\l\_\_marginalia\_item\_height\_dim  
435 \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`.  
436 \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`.  
437 \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`  
438 \dim\_new:N\l\_\_marginalia\_xshift\_computed\_dim  
439 \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.  
440 \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.

```
441 \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.

```
442 \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`.)

### 11.10.2 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>`.

```
443 \cs_new:Npn\__marginalia_process_item:nn #1#2
```

```
444 {
```

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

```
445   \int_gincr:N\g__marginalia_itemno_int
```

```
446   \group_begin:
```

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

```
447     \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.

```
448     \__marginalia_lua_load_item_data:n
```

```
449     { \int_value:w\g__marginalia_itemno_int }
```

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

```
450     \mode_if_math:TF
```

```
451     {
```

```
452       \cs_set_eq:NN
```

```
453       \__marginalia_typeset:n
```

```
454       \__marginalia_typeset_mmode:n
```

```
455     }
```

```
456     {
```

```
457       \legacy_if:nT{@inlabel}
```

```
458       { \leavevmode }
```

```
459       \mode_if_horizontal:TF
```

```
460       {
```

```
461         \cs_set_eq:NN
```

```

462         \_margin_ia_ttypeset:n
463         \_margin_ia_ttypeset_hmode:n
464     }
465     {
466         \cs_set_eq:NN
467         \_margin_ia_ttypeset:n
468         \_margin_ia_ttypeset_vmode:n
469     }
470 }

```

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

```

471 \int_case:nn{\l__margin_ia_valign_int}
472 {
473   {1}{ \cs_set_eq:NN\_margin_ia_item_box_set:Nn\ vbox_set_top:Nn }
474   {2}{ \cs_set_eq:NN\_margin_ia_item_box_set:Nn\ vbox_set:Nn }
475   {3}{ \cs_set_eq:NN\_margin_ia_item_box_set:Nn\ vbox_set_center:Nn }
476   {4}{ \cs_set_eq:NN\_margin_ia_item_box_set:Nn\ vbox_set_midway:Nn }
477 }

```

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

```

478 \_margin_ia_set_xsep_width_style_mark:

```

Typeset the `<content>` into `\l__margin_ia_item_box`. Use `\@parboxrestore` for brevity, even though `\hsize` and `\linewidth` are subsequently set to `\l__margin_ia_width_dim`. Make available `\margin_ia_page` and `\margin_ia_column`.

```

479 \_margin_ia_tagging_socket:n {marginpar/begin}
480 \_margin_ia_item_box_set:Nn\l__margin_ia_item_box{
481   \@parboxrestore
482   \_margin_ia_tagging_socket:n {para/restore}
483   \normalfont\normalsize
484
485   \tl_use:N\l__margin_ia_style_tl
486   \dim_set_eq:NN\hsize\l__margin_ia_width_dim
487   \dim_set_eq:NN\linewidth\hsize
488
489   \cs_set_eq:NN\margin_ia_page\l__margin_ia_page_int
490   \cs_set_eq:NN\margin_ia_column\l__margin_ia_column_computed_int
491
492   \group_begin:
493   \ignorespaces
494   #2
495   \par
496   \group_end:
497 }
498 \_margin_ia_tagging_socket:n{marginpar/end}

```

Measure `\l__margin_ia_item_box`.

```

499 \dim_set:Nn\l__margin_ia_item_height_dim
500   {\box_ht:N\l__margin_ia_item_box}
501 \dim_set:Nn\l__margin_ia_item_depth_dim
502   {\box_dp:N\l__margin_ia_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.

```

503     \_marginalia_typeset:n{
504         \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.

```

505     \iow_shipout_e:Ne\l__marginalia_aux_iow{
506         \token_to_str:N\marginalia@itemdata{
507             itemno=\int_value:w\g__marginalia_itemno_int,
508             abspageno=\exp_not:N\int_eval:n{g_shipout_readonly_int},
509             pageno=\exp_not:N\int_value:w\c@page,
510             type=\str_use:N\int_value:w\l__marginalia_type_int,
511             xpos=\exp_not:N\int_value:w\lastxpos,
512             ypos=\exp_not:N\int_value:w\lastypos,
513             height=\int_value:w\l__marginalia_item_height_dim,
514             depth=\int_value:w\l__marginalia_item_depth_dim,
515             pos=\int_value:w\l__marginalia_pos_int,
516             column=\int_value:w\l__marginalia_column_int,
517             yshift=\int_value:w\l__marginalia_default_yshift_dim,
518             ysep~above=\int_value:w\l__marginalia_ysep_above_dim,
519             ysep~below=\int_value:w\l__marginalia_ysep_below_dim,
520             ysep~page~top=\int_value:w\l__marginalia_ysep_page_top_dim,
521             ysep~page~bottom=\int_value:w\l__marginalia_ysep_page_bottom_dim,
522         }
523     }

```

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.

```

524     \int_if_zero:nF{\l__marginalia_enabled_computed_int}
525     {
526         \int_if_zero:nTF{\l__marginalia_side_computed_int}
527         {
528             \hbox_overlap_right:n{
529                 \kern\l__marginalia_xshift_computed_dim
530                 \tl_if_empty:NF\l__marginalia_mark_tl
531                 {
532                     \hbox_overlap_right:n{ \tl_use:N\l__marginalia_mark_tl }
533                 }
534                 \kern\l__marginalia_xsep_dim
535                 \_marginalia_place_item_box:
536             }
537         }
538         {
539             \hbox_overlap_left:n{
540                 \_marginalia_place_item_box:
541                 \kern\l__marginalia_xsep_dim
542                 \tl_if_empty:NF\l__marginalia_mark_tl
543                 {
544                     \hbox_overlap_left:n{ \tl_use:N\l__marginalia_mark_tl }
545                 }
546                 \kern-\l__marginalia_xshift_computed_dim

```

```

547         }
548     }
549 }
550 }

```

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

```

551 \group_end:
552 }

```

*(End of definition for `\_marginolia_process_item:nn`.)*

### 11.10.3 Horizontal separation, width, style, mark selection

`\_marginolia_set_xsep_width_style_mark:` Set `\l__marginolia_xsep_dim`, `\l__marginolia_width_dim`, and `\l__marginolia_style_tl`, based on `\l__marginolia_marginno_computed_int`.

```

553 \cs_new:Npn\_marginolia_set_xsep_width_style_mark:
554 {
555   \int_case:nn{\l__marginolia_marginno_computed_int}
556   {
557     {0}
558     {
559       \cs_set_eq:NN\l__marginolia_xsep_dim
560         \l__marginolia_xsep_recto_outer_dim
561       \cs_set_eq:NN\l__marginolia_width_dim
562         \l__marginolia_width_recto_outer_dim
563       \cs_set_eq:NN\l__marginolia_style_tl
564         \l__marginolia_style_recto_outer_tl
565       \cs_set_eq:NN\l__marginolia_mark_tl
566         \l__marginolia_mark_recto_outer_tl
567     }
568     {1}
569     {
570       \cs_set_eq:NN\l__marginolia_xsep_dim
571         \l__marginolia_xsep_recto_inner_dim
572       \cs_set_eq:NN\l__marginolia_width_dim
573         \l__marginolia_width_recto_inner_dim
574       \cs_set_eq:NN\l__marginolia_style_tl
575         \l__marginolia_style_recto_inner_tl
576       \cs_set_eq:NN\l__marginolia_mark_tl
577         \l__marginolia_mark_recto_inner_tl
578     }
579     {2}
580     {
581       \cs_set_eq:NN\l__marginolia_xsep_dim
582         \l__marginolia_xsep_verso_outer_dim
583       \cs_set_eq:NN\l__marginolia_width_dim
584         \l__marginolia_width_verso_outer_dim
585       \cs_set_eq:NN\l__marginolia_style_tl
586         \l__marginolia_style_verso_outer_tl
587       \cs_set_eq:NN\l__marginolia_mark_tl
588         \l__marginolia_mark_verso_outer_tl
589     }
590     {3}
591     {

```

```

592     \cs_set_eq:NN\l__marginalia_xsep_dim
593         \l__marginalia_xsep_verso_inner_dim
594     \cs_set_eq:NN\l__marginalia_width_dim
595         \l__marginalia_width_verso_inner_dim
596     \cs_set_eq:NN\l__marginalia_style_tl
597         \l__marginalia_style_verso_inner_tl
598     \cs_set_eq:NN\l__marginalia_mark_tl
599         \l__marginalia_mark_verso_inner_tl
600 }
601 {4}
602 {
603     \cs_set_eq:NN\l__marginalia_xsep_dim
604         \l__marginalia_xsep_right_between_dim
605     \cs_set_eq:NN\l__marginalia_width_dim
606         \l__marginalia_width_right_between_dim
607     \cs_set_eq:NN\l__marginalia_style_tl
608         \l__marginalia_style_right_between_tl
609     \cs_set_eq:NN\l__marginalia_mark_tl
610         \l__marginalia_mark_right_between_tl
611 }
612 {5}
613 {
614     \cs_set_eq:NN\l__marginalia_xsep_dim
615         \l__marginalia_xsep_left_between_dim
616     \cs_set_eq:NN\l__marginalia_width_dim
617         \l__marginalia_width_left_between_dim
618     \cs_set_eq:NN\l__marginalia_style_tl
619         \l__marginalia_style_left_between_tl
620     \cs_set_eq:NN\l__marginalia_mark_tl
621         \l__marginalia_mark_left_between_tl
622 }
623 }
624 }

```

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

#### 11.10.4 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.

```

625 \cs_new:Npn \vbox_set_center:Nn #1#2
626 {
627     \vbox_set_top:Nn #1{#2}
628     \dim_set:Nn \l_tmpa_dim{ \box_ht:N#1 + \box_dp:N#1 }
629     \box_set_ht:Nn #1 { .5\l_tmpa_dim }
630     \box_set_dp:Nn #1 { .5\l_tmpa_dim }
631 }

```

(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.

```

632 \cs_new:Npn \vbox_set_midway:Nn #1#2
633 {

```

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.

```

634   \vbox_set_top:Nn #1{#2}
635   \tag_suspend:n{ marginalia }
636   \vbox_set:Nn \l_tmpa_box {#2}
637   \tag_resume:n{ marginalia }
638   \dim_set:Nn \l_tmpa_dim{ \box_dp:N#1 - \box_dp:N\l_tmpa_box}
639   \box_set_ht:Nn #1 { \box_ht:N#1 + .5\l_tmpa_dim }
640   \box_set_dp:Nn #1 { \box_dp:N#1 - .5\l_tmpa_dim }
641 }

```

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

### 11.10.5 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.

```

642 \cs_new:Npn \__marginalia_place_item_box:
643 {
644   \smash
645   {
646     \box_move_up:nn{\l__marginalia_yshift_computed_dim}
647     {
648       \box_use:N\l__marginalia_item_box
649     }
650   }
651 }

```

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

`\__marginalia_typeset_mmode:n`  
`\__marginalia_typeset_hmode:n`  
`\__marginalia_typeset_vmode: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.

```

652 \cs_new:Npn \__marginalia_typeset_mmode:n #1
653 {
654   #1
655 }
656 \cs_new:Npn \__marginalia_typeset_hmode:n #1
657 {
658   \@bsphack
659   #1
660   \@esphack
661 }
662 \bool_new:N\l__marginalia_nobreak_bool
663 \cs_new:Npn \__marginalia_typeset_vmode:n #1

```

```

664 {
665   \bool_set:Nn\l__marginalia_nobreak_bool{ \legacy_if_p:n{@nobreak} }
666   \nobreak\noindent #1\par
667   \skip_vertical:n{-\baselineskip}
668   \legacy_if_gset:nn{ @nobreak }{ \l__marginalia_nobreak_bool }
669 }

```

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

## 11.11 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`.

```

670 \NewDocumentCommand{\marginalia}{ 0{} +m }
671 {
672   \__marginalia_process_item:nn{#1}{#2}
673 }

```

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

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

```

674 \NewDocumentCommand{\marginaliasetup}{ m }
675 {
676   \__marginalia_setup:n{ #1 }
677 }

```

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

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

```

678 \NewDocumentCommand{\marginalianewgeometry}{}
679 {
680   \__marginalia_write_page_data:
681 }

```

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

```

682 </package>

```

## 12 Implementation (Lua backend)

```

683 <*lua>

```

### 12.1 Global variables

Global tables for `page_data` and `item_data`.

```

684 local PAGE_DATA_MAIN_TABLE = {}
685 local ITEM_DATA_MAIN_TABLE = {}

```

Global tables for compiling reports.

```

686 local PROBLEM_REPORT_TABLE = {}
687 local PAGE_CHANGE_REPORT_TABLE = {}
688 local ITEM_CHANGE_REPORT_TABLE = {}

```

Global configuration for reports.

```
689 local PROBLEM_REPORT_MAX_LENGTH = 40
690 local PAGE_CHANGE_REPORT_MAX_LENGTH = 10
691 local ITEM_CHANGE_REPORT_MAX_LENGTH = 10
```

## 12.2 Constants

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

```
692 local TYPE_NORMAL = 1
693 local TYPE_FIXED = 2
694 local TYPE_OPTFIXED = 3
```

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

```
695 local POS_AUTO = 1
696 local POS_REVERSE = 2
697 local POS_LEFT = 3
698 local POS_RIGHT = 4
699 local POS_NEAREST = 5
```

## 12.3 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 T<sub>E</sub>X scaled points (sp)

### 12.3.1 Keys for both page and item data tables

Integer: Absolute page number in output file (not on-page number), used in both page\_<sub>data</sub> and item\_<sub>data</sub> tables

```
700 local KEY_ABSPAGENO = 'abspageno'
```

Boolean: Used to mark page\_<sub>data</sub> or item\_<sub>data</sub> as checked when the .aux file is read back at the end of the document

```
701 local KEY_CHECKED = 'checked'
```

### 12.3.2 Keys for page data tables, layout etc.

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

```
702 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.

```
703 local KEY_HOFFSETORIGIN = 'hoffsetorigin'
704 local KEY_VOFFSETORIGIN = 'voffsetorigin'
```

Dimensions: corresponding to obvious L<sup>A</sup>T<sub>E</sub>X dimensions

```
705 local KEY_HOFFSET = 'hoffset'
706 local KEY_VOFFSET = 'voffset'
707 local KEY_PAGEHEIGHT = 'pageheight'
708 local KEY_ODDSIDEMARGIN = 'oddsidemargin'
709 local KEY_EVENSIDEMARGIN = 'evensidemargin'
710 local KEY_TEXTWIDTH = 'textwidth'
711 local KEY_COLUMNWIDTH = 'columnwidth'
712 local KEY_COLUMNSEP = 'columnsep'
```

Integer: either 1 or 2, depending on whether L<sup>A</sup>T<sub>E</sub>X was in one- or two-column mode

```
713 local KEY_COLUMNCOUNT = 'columncount'
```

Boolean: true iff L<sup>A</sup>T<sub>E</sub>X is in twoside mode

```
714 local KEY_TWOSIDE = 'twoside'
```

### 12.3.3 Keys for item data tables

Integer: Used to identify data with item

```
715 local KEY_ITEMNO = 'itemno'
```

Integer: On-page number

```
716 local KEY_PAGENO = 'pageno'
```

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

```
717 local KEY_XPOS = 'xpos'
```

```
718 local KEY_YPOS = 'ypos'
```

Dimensions: Height and depth of typeset item

```
719 local KEY_HEIGHT = 'height'
```

```
720 local KEY_DEPTH = 'depth'
```

Integer: Specified type, following `TYPE_*`

```
721 local KEY_TYPE = 'type'
```

Integer: corresponds to value of `pos` key: 0 = auto, 1 = reverse, 2 = left, 3 = right, 4 = nearest

```
722 local KEY_POS = 'pos'
```

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

```
723 local KEY_COLUMN = 'column'
```

Dimension: specified vertical shift

```
724 local KEY_YSHIFT = 'yshift'
```

Dimensions: specified vertical separations

```
725 local KEY_YSEP_ABOVE = 'ysep above'
```

```
726 local KEY_YSEP_BELOW = 'ysep below'
```

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

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

The preceding keys refer to values that will be supplied from L<sup>A</sup>T<sub>E</sub>X. The remaining values will be computed in Lua and passed back to L<sup>A</sup>T<sub>E</sub>X.

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

```
729 local KEY_COLNO_COMPUTED = 'colno computed'
```

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

```
730 local KEY_XSHIFT_COMPUTED = 'xshift computed'
```

Dimension: Computed vertical shift

```
731 local KEY_YSHIFT_COMPUTED = 'yshift computed'
```

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

```
732 local KEY_SIDE_COMPUTED = 'side computed'
```

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

```
733 local KEY_MARGINNO_COMPUTED = 'marginno computed'
```

Boolean: Whether the item will actually appear on the page

```
734 local KEY_ENABLED_COMPUTED = 'enabled computed'
```

## 12.4 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.)<sup>12</sup>

```
735 local function list_filter(t, f)
736   local j = 1
737   local n = #t
738
739   for i=1,n do
740     if (f(t[i])) then
741       if (i ~= j) then
742         t[j] = t[i]
743         t[i] = nil
744       end
745       j = j + 1
746     else
747       t[i] = nil
748     end
749   end
750
751 end
```

*(End of definition for list\_filter.)*

list\_filter Return boolean true iff `s` is exactly the string 'true'.

```
752 local function toboolean(s)
753   return s == "true"
754 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.

```
755 local function get_data_page_number(data)
756   local pageno = data[KEY_PAGENO]
757   if pageno ~= nil then
758     return 'p' .. pageno .. ' (' .. data[KEY_ABSPAGENO] .. ')'
759   else
760     return data[KEY_ABSPAGENO]
761   end
762 end
```

*(End of definition for get\_data\_page\_number.)*

## 12.5 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`.

```
763 local function parse_data(keyvalue_string,conversion_table,defaults_table)
764
765   local key
766   local value
767   local result = {}
768
769   for s in string.gmatch(keyvalue_string,'([^\,]+)') do
770
771     key,value = string.match(s,'^(.+)=(.+)$')
772     local conv = conversion_table[key]
773     if conv ~= nil then
774       result[key] = conv(value)
775     end
776
777   end
778
779   for key,value in pairs(defaults_table) do
780     if not(result[key] ~= nil) then
781       result[key] = value
782     end
783   end
784
785   return result
786
787 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`.

```
788 local function check_data(keyvalue_string,conversion_table,defaults_table,
789                           data_table,data_table_key_field,report_table)
790
791   local new_data = parse_data(keyvalue_string,
792                               conversion_table,defaults_table)
793
```

```

794 local data_table_key = new_data[data_table_key_field]
795
796 local stored_data = data_table[data_table_key]
797 if stored_data == nil then
798     table.insert(
799         report_table,
800         get_data_page_number(new_data) .. ' New'
801     )
802 else
803     local change_report = ''
804     for k,_ in pairs(conversion_table) do
805         if stored_data[k] ~= new_data[k] then
806             change_report = change_report
807                 .. ' ' .. k .. ':' ..
808                 tostring(stored_data[k]) .. '->' .. tostring(new_data[k])
809         end
810     end
811     if change_report ~= '' then
812         table.insert(
813             report_table,
814             get_data_page_number(new_data) .. ' ' .. change_report
815         )
816     end
817     stored_data[KEY_CHECKED] = true
818 end
819
820 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`.

```

821 local function check_removed_data(data_table,report_table)
822     for _,data in pairs(data_table) do
823         if not data[KEY_CHECKED] then
824             table.insert(
825                 report_table,
826                 ' Removed'
827             )
828             break
829         end
830     end
831 end

```

*(End of definition for check\_removed\_data.)*

## 12.6 Processing of page data from .aux file

Conversion and default tables.

```

832 local PAGE_DATA_CONVERSION_TABLE = {
833     [KEY_PAGEDATANO] = tonumber,
834     [KEY_ABSPAGENO] = tonumber,
835     [KEY_HOFFSETORIGIN] = tonumber,

```

```

836 [KEY_VOFFSETORIGIN] = tonumber,
837 [KEY_HOFFSET] = tonumber,
838 [KEY_VOFFSET] = tonumber,
839 [KEY_PAGEHEIGHT] = tonumber,
840 [KEY_ODDSIDEMARGIN] = tonumber,
841 [KEY_EVENSIDEMARGIN] = tonumber,
842 [KEY_COLUMNCOUNT] = tonumber,
843 [KEY_COLUMNWIDTH] = tonumber,
844 [KEY_COLUMNSEP] = tonumber,
845 [KEY_TEXTWIDTH] = tonumber,
846 [KEY_TWOSIDE] = toboolean,
847 }
848 local PAGE_DATA_DEFAULT_TABLE = {
849   [KEY_PAGEDATANO] = 0,
850   [KEY_A BSPAGENO] = 0,
851   [KEY_HOFFSETORIGIN] = tex.sp('1in'),
852   [KEY_VOFFSETORIGIN] = tex.sp('1in'),
853   [KEY_HOFFSET] = tex.dimen['hoffset'],
854   [KEY_VOFFSET] = tex.dimen['voffset'],
855   [KEY_PAGEHEIGHT] = tex.dimen['pageheight'],
856   [KEY_ODDSIDEMARGIN] = tex.dimen['oddsidemargin'],
857   [KEY_EVENSIDEMARGIN] = tex.dimen['evensidemargin'],
858   [KEY_TEXTWIDTH] = tex.dimen['textwidth'],
859   [KEY_COLUMNWIDTH] = tex.dimen['columnwidth'],
860   [KEY_COLUMNSEP] = tex.dimen['columnsep'],
861   [KEY_COLUMNCOUNT] = 1,
862   [KEY_TWOSIDE] = false,
863   [KEY_CHECKED] = false,
864 }

```

store\_page\_data Store page data supplied by keyvalue\_string in PAGE\_DATA\_MAIN\_TABLE.

```

865 local function store_page_data(keyvalue_string)
866
867   local page_data = parse_data(keyvalue_string,
868                               PAGE_DATA_CONVERSION_TABLE,
869                               PAGE_DATA_DEFAULT_TABLE)
870
871   PAGE_DATA_MAIN_TABLE[page_data[KEY_PAGEDATANO]] = page_data
872
873 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.

```

874 local function store_default_page_data()
875
876   default_page_data = parse_data('',
877                                   PAGE_DATA_CONVERSION_TABLE,
878                                   PAGE_DATA_DEFAULT_TABLE)
879
880   default_page_data[KEY_A BSPAGENO] = 1
881   default_page_data[KEY_CHECKED] = true

```

```

882
883 PAGE_DATA_MAIN_TABLE[0] = default_page_data
884
885 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.

```

886 local function check_page_data(keyvalue_string)
887
888   check_data(keyvalue_string,
889             PAGE_DATA_CONVERSION_TABLE,PAGE_DATA_DEFAULT_TABLE,
890             PAGE_DATA_MAIN_TABLE,KEY_PAGEDATANO,
891             PAGE_CHANGE_REPORT_TABLE)
892
893 end

```

*(End of definition for check\_page\_data.)*

## 12.7 Processing of item data from .aux file

Conversion and default tables.

```

894 local ITEM_DATA_CONVERSIONS = {
895   [KEY_ITEMNO] = tonumber,
896   [KEY_ABSPAGENO] = tonumber,
897   [KEY_PAGENO] = tonumber,
898   [KEY_XPOS] = tonumber,
899   [KEY_YPOS] = tonumber,
900   [KEY_HEIGHT] = tonumber,
901   [KEY_DEPTH] = tonumber,
902   [KEY_TYPE] = tonumber,
903   [KEY_POS] = tonumber,
904   [KEY_COLUMN] = tonumber,
905   [KEY_YSHIFT] = tonumber,
906   [KEY_YSEP_ABOVE] = tonumber,
907   [KEY_YSEP_BELOW] = tonumber,
908   [KEY_YSEP_PAGE_TOP] = tonumber,
909   [KEY_YSEP_PAGE_BOTTOM] = tonumber,
910   [KEY_CHECKED] = toboolean,
911 }
912 local ITEM_DATA_DEFAULTS = {
913   [KEY_ITEMNO] = 0,
914   [KEY_ABSPAGENO] = 1,
915   [KEY_PAGENO] = 1,
916   [KEY_XPOS] = 0,
917   [KEY_YPOS] = 0,
918   [KEY_HEIGHT] = 0,
919   [KEY_DEPTH] = 0,
920   [KEY_TYPE] = 0,
921   [KEY_POS] = 0,
922   [KEY_COLUMN] = -1,
923   [KEY_YSHIFT] = 0,
924   [KEY_YSEP_ABOVE] = tex.dimen['marginparpush'],

```

```

925 [KEY_YSEP_BELOW] = tex.dimen['marginparpush'],
926 [KEY_YSEP_PAGE_TOP] = tex.dimen['marginparpush'],
927 [KEY_YSEP_PAGE_BOTTOM] = tex.dimen['marginparpush'],
928 [KEY_COLNO_COMPUTED] = 0,
929 [KEY_XSHIFT_COMPUTED] = 0,
930 [KEY_YSHIFT_COMPUTED] = 0,
931 [KEY_SIDE_COMPUTED] = 0,
932 [KEY_MARGINNO_COMPUTED] = 0,
933 [KEY_ENABLED_COMPUTED] = true,
934 [KEY_CHECKED] = false,
935 }

```

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.

```

936 local function store_item_data(keyvalue_string)
937
938   local item = parse_data(keyvalue_string,
939                           ITEM_DATA_CONVERSIONS,
940                           ITEM_DATA_DEFAULTS)
941
942   ITEM_DATA_MAIN_TABLE[item[KEY_ITEMNO]] = item
943
944 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.

```

945 local function check_item_data(keyvalue_string)
946
947   check_data(keyvalue_string,
948              ITEM_DATA_CONVERSIONS, ITEM_DATA_DEFAULTS,
949              ITEM_DATA_MAIN_TABLE, KEY_ITEMNO,
950              ITEM_CHANGE_REPORT_TABLE)
951
952 end

```

*(End of definition for check\_item\_data.)*

## 12.8 Writing reports

`write_report` Write the data contained in report\_table to T<sub>E</sub>X in a format suitable for a package warning. The written text will contain at most max\_length items.

```

953 local function write_report(report_table,max_length,noun)
954
955   if #report_table > 0 then
956     local report_text
957     local report_length
958
959     if #report_table <= max_length then
960       report_length = #report_table
961       report_text = ' Here are the ' .. noun .. ':\n'

```

```

962     else
963         report_length = max_length
964         report_text = ' Here are the first ' .. report_length .. ' ' .. noun .. ':\n'
965     end
966
967     for i=1,report_length do
968         report_text = report_text .. report_table[i]
969         if i < report_length then
970             report_text = report_text .. '\n'
971         end
972     end
973
974     tex.print(report_text)
975 end
976
977 end

```

*(End of definition for write\_report.)*

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

```

978 local function write_problem_report()
979
980     write_report(PROBLEM_REPORT_TABLE,PROBLEM_REPORT_MAX_LENGTH, 'problems')
981
982 end

```

*(End of definition for write\_problem\_report.)*

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

```

983 local function write_item_change_report()
984
985     check_removed_data(ITEM_DATA_MAIN_TABLE,ITEM_CHANGE_REPORT_TABLE)
986     write_report(ITEM_CHANGE_REPORT_TABLE,ITEM_CHANGE_REPORT_MAX_LENGTH, 'changes')
987
988 end

```

*(End of definition for write\_item\_change\_report.)*

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

```

989 local function write_page_change_report()
990
991     check_removed_data(PAGE_DATA_MAIN_TABLE,PAGE_CHANGE_REPORT_TABLE)
992     write_report(PAGE_CHANGE_REPORT_TABLE,PAGE_CHANGE_REPORT_MAX_LENGTH, 'changes')
993
994 end

```

*(End of definition for write\_page\_change\_report.)*

## 12.9 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 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`.

```
995 local RIGHTSIDE_LOOKUP_TABLE = {
996   [true] = {
997     [0] = {
998       [POS_AUTO] = true,
999       [POS_REVERSE] = false,
1000     },
1001     [1] = {
1002       [POS_AUTO] = false,
1003       [POS_REVERSE] = true,
1004     },
1005     [2] = {
1006       [POS_AUTO] = true,
1007       [POS_REVERSE] = false,
1008     },
1009   },
1010   [false] = {
1011     [0] = {
1012       [POS_AUTO] = false,
1013       [POS_REVERSE] = true,
1014     },
1015     [1] = {
1016       [POS_AUTO] = false,
1017       [POS_REVERSE] = true,
1018     },
1019     [2] = {
1020       [POS_AUTO] = true,
1021       [POS_REVERSE] = false,
1022     },
1023   },
1024 }
```

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`).

```
1025 local MARGINNO_LOOKUP_TABLE = {
1026   [true] = {
1027     [0] = {
1028       [false] = 1,
1029       [true] = 0,
1030     },
1031     [1] = {
1032       [false] = 1,
1033       [true] = 5,
1034     },
```

```

1035     [2] = {
1036         [false] = 4,
1037         [true] = 0,
1038     },
1039 },
1040 [false] = {
1041     [0] = {
1042         [false] = 2,
1043         [true] = 3,
1044     },
1045     [1] = {
1046         [false] = 2,
1047         [true] = 5,
1048     },
1049     [2] = {
1050         [false] = 4,
1051         [true] = 3,
1052     },
1053 },
1054 }

```

`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.

```

1055 local function compute_items_horizontal(item_data_list,page_data)
Immediately return if item_data_list is empty, to avoid edge cases.
1056 if #item_data_list == 0 then
1057     return
1058 end

```

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

```

1059 local pageno = item_data_list[1][KEY_PAGENO]
1060 local twoside = page_data[KEY_TWOSIDE]
1061 local recto = ((pageno % 2) == 1) or (not twoside)
1062 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).

```

1063 local x_textleft = {}
1064 local x_textright = {}
1065 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.

```

1066 if recto then
1067     x_textleft[0] = (
1068         page_data[KEY_HOFFSETORIGIN]
1069         + page_data[KEY_HOFFSET]
1070         + page_data[KEY_ODDSIDEMARGIN]
1071     )

```

```

1072     x_textright[0] = (
1073         x_textleft[0]
1074         + page_data[KEY_TEXTWIDTH]
1075     )
1076 else
1077     x_textleft[0] = (
1078         page_data[KEY_HOFFSETORIGIN]
1079         + page_data[KEY_HOFFSET]
1080         + page_data[KEY_EVENSIDEMARGIN]
1081     )
1082     x_textright[0] = (
1083         x_textleft[0]
1084         + page_data[KEY_TEXTWIDTH]
1085     )
1086 end
1087 x_textmiddle[0] = (x_textleft[0] + x_textright[0])/2
1088
1089
1090 if columncount == 1 then

```

If the page is one-column, the field KEY\_COLNO\_COMPUTED can be set immediately for every item\_data.

```

1091     for i=1,#item_data_list do
1092         item_data_list[i][KEY_COLNO_COMPUTED] = 0
1093     end
1094 else

```

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

```

1095     x_textleft[1] = x_textleft[0]
1096     x_textright[1] = (
1097         x_textleft[1]
1098         + page_data[KEY_COLUMNWIDTH]
1099     )
1100     x_textmiddle[1] = (x_textleft[1] + x_textright[1])/2
1101
1102     x_textleft[2] = (
1103         x_textright[1]
1104         + page_data[KEY_COLUMNSEP]
1105     )
1106     x_textright[2] = (
1107         x_textleft[2]
1108         + page_data[KEY_COLUMNWIDTH]
1109     )
1110     x_textmiddle[2] = (x_textleft[2] + x_textright[2])/2
1111

```

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

```

1112     local left_column_x_limit = (
1113         x_textright[1]
1114         + .5*page_data[KEY_COLUMNSEP]
1115     )

```

Now set the field KEY\_COLNO\_COMPUTED for each item.

```

1116     for i=1,#item_data_list do
1117         local item_data = item_data_list[i]
1118
1119         if item_data[KEY_COLUMN] >= 0 then
1120             item_data[KEY_COLNO_COMPUTED] = item_data[KEY_COLUMN]
1121         else
1122             if item_data[KEY_XPOS] <= left_column_x_limit then
1123                 item_data[KEY_COLNO_COMPUTED] = 1
1124             else
1125                 item_data[KEY_COLNO_COMPUTED] = 2
1126             end
1127         end
1128     end
1129 end
1130 end

```

For every item\_data in item\_data\_list, compute and set the fields KEY\_SIDE\_COMPUTED, KEY\_XSHIFT\_COMPUTED, and KEY\_MARGINNO\_COMPUTED.

```

1131     for i=1,#item_data_list do
1132         local item = item_data_list[i]
1133
1134         local pos = item[KEY_POS]
1135         local colnocomputed = item[KEY_COLNO_COMPUTED]
1136
1137         if pos == POS_LEFT then
1138             rightside = false
1139         elseif pos == POS_RIGHT then
1140             rightside = true
1141         elseif pos == POS_NEAREST then
1142             rightside = (item[KEY_XPOS] >= x_textmiddle[colnocomputed])
1143         else

```

(In this case, pos must be POS\_AUTO or POS\_REVERSE.)

```

1144             rightside = RIGHTSIDE_LOOKUP_TABLE[recto][colnocomputed][pos]
1145         end
1146
1147         local marginno = MARGINNO_LOOKUP_TABLE[recto][colnocomputed][rightside]
1148
1149         if rightside then
1150             item[KEY_SIDE_COMPUTED] = 0
1151             item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
1152                                     + x_textright[colnocomputed]
1153         else
1154             item[KEY_SIDE_COMPUTED] = 1
1155             item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
1156                                     + x_textleft[colnocomputed]
1157         end
1158         item[KEY_MARGINNO_COMPUTED] = marginno
1159
1160     end
1161 end
1162 end

```

(End of definition for compute\_items\_horizontal.)

## 12.10 Computing vertical positions

### 12.10.1 Auxiliary functions

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

```
1163 local function get_y_item_top(item_data)
1164   return item_data[KEY_YPOS]
1165         + item_data[KEY_YSHIFT_COMPUTED]
1166         + item_data[KEY_HEIGHT]
1167 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`.

```
1168 local function get_y_item_bottom(item_data)
1169   return item_data[KEY_YPOS]
1170         - item_data[KEY_DEPTH]
1171         + item_data[KEY_YSHIFT_COMPUTED]
1172 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).

```
1173 local function get_ysep_list(item_data_list)
1174
1175   local ysep_list = {}
1176
1177   ysep_list[0] = 0
1178   for i=1,#item_data_list-1 do
1179     ysep_list[i] = math.max(
1180       item_data_list[i][KEY_YSEP_BELOW],
1181       item_data_list[i+1][KEY_YSEP_ABOVE]
1182     )
1183   end
1184   ysep_list[#item_data_list] = 0
1185
1186   return ysep_list
1187
1188 end
```

*(End of definition for `get_ysep_list`.)*

## 12.10.2 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.

```
1189 local function compute_items_vertical_optfixed_enabled(item_data_list)
1190
1191   local optfixed_item_data_list = {}
1192   local fixed_item_data_list = {}
1193
1194   for _,item_data in pairs(item_data_list) do
1195     if item_data[KEY_TYPE] == TYPE_OPTFIXED then
1196       optfixed_item_data_list[#optfixed_item_data_list+1] = item_data
1197     elseif item_data[KEY_TYPE] == TYPE_FIXED then
1198       fixed_item_data_list[#fixed_item_data_list+1] = item_data
1199     end
1200   end
1201
1202   for _,optfixed_item_data in pairs(optfixed_item_data_list) do
1203     local optfixed_y_item_top = get_y_item_top(optfixed_item_data)
1204     local optfixed_y_item_bottom = get_y_item_bottom(optfixed_item_data)
1205
1206     for _,fixed_item_data in pairs(fixed_item_data_list) do
1207       local fixed_y_item_top = get_y_item_top(fixed_item_data)
1208       local fixed_y_item_bottom = get_y_item_bottom(fixed_item_data)
1209
1210       if (
1211         (
1212           (fixed_y_item_bottom - optfixed_y_item_top)
1213           <
1214           math.max(
1215             fixed_item_data[KEY_YSEP_BELOW],
1216             optfixed_item_data[KEY_YSEP_ABOVE]
1217           )
1218         )
1219         and
1220         (
1221           (optfixed_y_item_bottom - fixed_y_item_top)
1222           <
1223           math.max(
1224             optfixed_item_data[KEY_YSEP_BELOW],
1225             fixed_item_data[KEY_YSEP_ABOVE]
1226           )
1227         )
1228       ) then
1229         optfixed_item_data[KEY_ENABLED_COMPUTED] = false
1230         break
1231       end
1232     end
1233   end
1234
1235 end
```

(End of definition for `compute_items_vertical_optfixed_enabled`.)

### 12.10.3 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.

```
1236 local function compute_items_vertical_adjustment(item_data_list,page_data)
```

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

```
1237   if #item_data_list == 0 then
1238     return
1239   end
```

```
1240
1241   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.

```
1242   local y_limit_above = (
1243     page_data[KEY_VOFFSET]
1244     + page_data[KEY_PAGEHEIGHT]
1245     - item_data_list[1][KEY_YSEP_PAGE_TOP]
1246   )
1247
1248   for i=1,#item_data_list do
1249     local item_data = item_data_list[i]
1250
1251     local y_item_top = get_y_item_top(item_data)
1252
1253     if y_item_top > y_limit_above then
1254       if item_data[KEY_TYPE] == TYPE_NORMAL then
1255         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1256                                           + (y_limit_above - y_item_top)
1257       end
1258     end
1259
1260     y_limit_above = get_y_item_bottom(item_data) - ysep_list[i]
1261   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.

```
1262   local y_limit_below = (
1263     page_data[KEY_VOFFSET]
1264     + item_data_list[#item_data_list][KEY_YSEP_PAGE_BOTTOM]
1265   )
1266
1267   for i=#item_data_list,1,-1 do
1268     local item_data = item_data_list[i]
1269
1270     local y_item_bottom = get_y_item_bottom(item_data)
1271
1272     if y_item_bottom < y_limit_below then
```

```

1273     if item_data[KEY_TYPE] == TYPE_NORMAL then
1274         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1275                                         + (y_limit_below - y_item_bottom)
1276     end
1277 end
1278
1279 y_limit_below = get_y_item_top(item_data) + ysep_list[i-1]
1280 end
1281
1282 end

```

(End of definition for compute\_items\_vertical\_adjustment.)

#### 12.10.4 Checking vertical adjustment

Messages to use when checking results of vertical adjustment.

```

1283 local ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES = {
1284     [TYPE_NORMAL] = 'Moveable item > ysep page top',
1285     [TYPE_FIXED] = 'Topmost fixed item > ysep page top',
1286     [TYPE_OPTFIXED] = 'Topmost optfixed item > ysep page top',
1287 }
1288 local ITEM_CLASH_MESSAGES = {
1289     [TYPE_NORMAL] = {
1290         [TYPE_NORMAL] = 'moveable items'
1291         .. ' (this shouldn\'t happen)',
1292         [TYPE_FIXED] = 'moveable item above fixed item',
1293         [TYPE_OPTFIXED] = 'moveable item above optfixed item',
1294     },
1295     [TYPE_FIXED] = {
1296         [TYPE_NORMAL] = 'moveable item below fixed item',
1297         [TYPE_FIXED] = 'fixed items',
1298         [TYPE_OPTFIXED] = 'fixed item above optfixed item '
1299         .. ' (this shouldn\'t happen)',
1300     },
1301     [TYPE_OPTFIXED] = {
1302         [TYPE_NORMAL] = 'moveable items below optfixed item',
1303         [TYPE_FIXED] = 'fixed item below optfixed item '
1304         .. ' (this shouldn\'t happen)',
1305         [TYPE_OPTFIXED] = 'optfixed items '
1306         .. ' (this shouldn\'t happen)',
1307     },
1308 }
1309 local ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE = {
1310     [TYPE_NORMAL] = 'Moveable item < ysep page bottom',
1311     [TYPE_FIXED] = 'Bottommost fixed item < ysep page bottom',
1312     [TYPE_OPTFIXED] = 'Bottommost optfixed item < ysep page bottom',
1313 }

```

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.

```

1314 local function check_items_vertical(item_data_list,page_data)

```

Immediately return if item\_data\_list is empty, to avoid edge cases

```
1315  if (#item_data_list) == 0 then
1316      return
1317  end
1318
1319  local ysep_list = get_ysep_list(item_data_list)
1320
1321  local item_data
1322
```

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

```
1323  item_data = item_data_list[1]
1324  if (
1325      get_y_item_top(item_data) > page_data[KEY_VOFFSET]
1326      + page_data[KEY_PAGEHEIGHT]
1327      - item_data[KEY_YSEP_PAGE_TOP]
1328  ) then
1329      table.insert(
1330          PROBLEM_REPORT_TABLE,
1331          get_data_page_number(item_data)
1332          .. ' ' .. ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES[item_data[KEY_TYPE]]
1333      )
1334  end
1335
1336  for i=2,#item_data_list do
1337      local item_data = item_data_list[i]
1338      local prev_item_data = item_data_list[i-1]
1339      if (
1340          get_y_item_top(item_data) > get_y_item_bottom(prev_item_data)
1341          - ysep_list[i-1]
1342      ) then
1343          table.insert(
1344              PROBLEM_REPORT_TABLE,
1345              get_data_page_number(item_data)
1346              .. ' Clash: ' ..
1347              ITEM_CLASH_MESSAGES[prev_item_data[KEY_TYPE]][item_data[KEY_TYPE]]
1348          )
1349      end
1350  end
```

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

```
1351  item_data = item_data_list[#item_data_list]
1352  if (
1353      get_y_item_bottom(item_data) < page_data[KEY_VOFFSET]
1354      + item_data[KEY_YSEP_PAGE_BOTTOM]
1355  ) then
1356      table.insert(
1357          PROBLEM_REPORT_TABLE,
1358          get_data_page_number(item_data)
1359          .. ' ' .. ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE[item_data[KEY_TYPE]]
1360      )
1361  end
1362
1363  end
```

(End of definition for `check_items_vertical`.)

### 12.10.5 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`.

```
1364 local function compute_items_vertical(item_data_list,page_data)
```

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

```
1365   for i=1,#item_data_list do
1366     local item_data = item_data_list[i]
1367
1368     item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT]
1369   end
```

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

```
1370   compute_items_vertical_optfixed_enabled(item_data_list)
```

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

```
1371   list_filter(item_data_list,function(item_data)
1372     return item_data[KEY_ENABLED_COMPUTED]
1373   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`.

```
1374   table.sort(
1375     item_data_list,
1376     function(left,right)
1377       local y_diff = left[KEY_YPOS] - right[KEY_YPOS]
1378
1379       if y_diff > 0 then
1380         return true
1381       elseif y_diff < 0 then
1382         return false
1383       end
1384
1385       local x_diff = left[KEY_XPOS] - right[KEY_XPOS]
1386
1387       if x_diff < 0 then
1388         return true
1389       elseif x_diff > 0 then
1390         return false
1391       end
1392
1393       return (left[KEY_ITEMNO] < right[KEY_ITEMNO])
1394     end
1395   )
```

```
1396   compute_items_vertical_adjustment(item_data_list,page_data)
```

```
1397   check_items_vertical(item_data_list,page_data)
```

```
1400
```

```
1401 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`.

```
1402 local function compute_items()
```

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

```
1403   local max_abspageno = 0
```

```
1404
```

```
1405   for k,v in pairs(ITEM_DATA_MAIN_TABLE) do
```

```
1406     max_abspageno = math.max(v[KEY_ABSPAGENO],max_abspageno)
```

```
1407   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.

```
1408   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`.

```
1409   for i=1,max_abspageno do
```

```
1410     per_abspage_item_data_list[i] = {}
```

```
1411   end
```

```
1412   for _,item_data in pairs(ITEM_DATA_MAIN_TABLE) do
```

```
1413     local temp_table = per_abspage_item_data_list[item_data[KEY_ABSPAGENO]]
```

```
1414     temp_table[#temp_table+1] = item_data
```

```
1415   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 `\marginianewgeometry` 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`.

```
1416   local per_abspage_page_data_list = {}
```

```
1417   local pagedatano = 0
```

```
1418   for abspageno = 1,max_abspageno do
```

```
1419     while (
```

```
1420       PAGE_DATA_MAIN_TABLE[pagedatano+1] ~= nil
```

```
1421       and
```

```
1422       PAGE_DATA_MAIN_TABLE[pagedatano+1][KEY_ABSPAGENO] == abspageno
```

```
1423     ) do
```

```
1424       pagedatano = pagedatano+1
```

```
1425     end
```

```
1426     per_abspage_page_data_list[abspageno] = PAGE_DATA_MAIN_TABLE[pagedatano]
```

```
1427   end
```

Iterate through all pages and perform the necessary computations.

```

1428   for abspageno=1,#per_abspage_item_data_list do
1429     local current_page_data = per_abspage_page_data_list[abspageno]
1430     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.

```

1431     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.

```

1432     local current_page_item_data_sublists = {}
1433
1434     for i=0,5 do
1435       current_page_item_data_sublists[i] = {}
1436     end
1437
1438     for _,item_data in pairs(current_page_item_data_list) do
1439       table.insert(
1440         current_page_item_data_sublists[item_data[KEY_MARGINNO_COMPUTED]],
1441         item_data
1442       )
1443     end

```

Compute vertical positions for each sublist.

```

1444     for i=0,5 do
1445       compute_items_vertical(
1446         current_page_item_data_sublists[i],
1447         current_page_data
1448       )
1449     end
1450   end
1451 end

```

*(End of definition for compute\_items.)*

## 12.11 Passing item\_data back to L<sup>A</sup>T<sub>E</sub>X

load\_item\_data Set the relevant L<sup>A</sup>T<sub>E</sub>X counter and dimension variables to the values computed for itemno.

```

1452 local function load_item_data(itemno)
1453
1454   item = ITEM_DATA_MAIN_TABLE[tonumber(itemno)]
1455   if item == nil then
1456     item = ITEM_DATA_DEFAULTS
1457   end
1458
1459   tex.count['l__marginalia_page_int'] = item[KEY_PAGENO]
1460   tex.count['l__marginalia_column_computed_int'] = item[KEY_COLNO_COMPUTED]
1461   tex.dimen['l__marginalia_xshift_computed_dim'] = item[KEY_XSHIFT_COMPUTED]
1462   tex.dimen['l__marginalia_yshift_computed_dim'] = item[KEY_YSHIFT_COMPUTED]
1463   tex.count['l__marginalia_side_computed_int'] = item[KEY_SIDE_COMPUTED]
1464   tex.count['l__marginalia_marginno_computed_int']
1465     = item[KEY_MARGINNO_COMPUTED]
1466   if item[KEY_ENABLED_COMPUTED] then
1467     tex.count['l__marginalia_enabled_computed_int'] = 1

```

```

1468 else
1469     tex.count['l__marginalia_enabled_computed_int'] = 0
1470 end
1471
1472 end

```

*(End of definition for load\_item\_data.)*

## 12.12 Export public functions

Finally, make available the functions that will be called from L<sup>A</sup>T<sub>E</sub>X using `\lua_now:n` and `\lua_now:e`.

```

1473 return {
1474     store_default_page_data = store_default_page_data,
1475     store_page_data = store_page_data,
1476     check_page_data = check_page_data,
1477
1478     store_item_data = store_item_data,
1479     check_item_data = check_item_data,
1480
1481     compute_items = compute_items,
1482
1483     load_item_data = load_item_data,
1484
1485     write_problem_report = write_problem_report,
1486
1487     write_page_change_report = write_page_change_report,
1488     write_item_change_report = write_item_change_report,
1489 }
1490  $\langle$ /lua $\rangle$ 

```

# 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.

	<b>Symbols</b>	
\'	1291, 1299, 1304, 1306	474, 475, 476, 489, 490, 559, 561, 563, 565, 570, 572, 574, 576, 581, 583, 585, 587, 592, 594, 596, 598, 603, 605, 607, 609, 614, 616, 618, 620
	<b>B</b>	
\baselineskip	37, 667	\currentgrouplevel
\begin	7, 9, 11, 13	40
bool commands:		<b>D</b>
\bool_new:N	662	dim commands:
\bool_set:Nn	665	\dim_new:N
\bool_to_str:n	420	78, 79, 80, 81, 82, 83, 150, 151, 152, 153, 200, 201, 202, 203, 204, 205, 434, 435, 438, 439
box commands:		\dim_set:Nn
\box_dp:N	502, 628, 638, 640	35, 499, 501, 628, 638
\box_ht:N	500, 628, 639	\dim_set_eq:NN
\box_move_up:nn	646	486, 487
\box_new:N	433	\l_tmpa_dim
\box_set_dp:Nn	630, 640	628, 629, 630, 638, 639, 640
\box_set_ht:Nn	629, 639	
\box_use:N	648	<b>E</b>
\l_tmpa_box	636, 638	\evensidemargin
		49, 415
		exp commands:
		\exp_not:N
		508, 509, 511, 512
	<b>C</b>	<b>G</b>
check commands:		get commands:
check_data	788	get_data_page_number
check_item_data	945	755
check_items_vertical	1314	get_y_item_bottom
check_page_data	886	1168
check_removed_data	821	get_y_item_top
column (option)	8	1163
\columnsep	419	get_ysep_list
\columnwidth	418	1173
compute commands:		group commands:
compute_items	1402	\group_begin:
compute_items_horizontal	1055	370, 446, 492
compute_items_vertical	1364	\group_end:
compute_items_vertical_adjustment	1236	386, 496, 551
compute_items_vertical_optfixed_		
enabled	1189	<b>H</b>
cs commands:		\hbox
\cs_new:Nn	31	34
\cs_new:Npn		hbox commands:
14, 26, 38, 45, 141, 145, 293, 297, 301, 305, 309, 313, 317, 321, 325, 329, 333, 335, 342, 368, 391, 401, 443, 553, 625, 632, 642, 652, 656, 663		\hbox_overlap_left:n
\cs_set_eq:NN		539, 544
49, 52, 339, 346, 352, 355, 358, 400, 427, 452, 461, 466, 473,		\hbox_overlap_right:n
		528, 532
		\headheight
		11, 143, 147
		\headsep
		11, 143, 147
		\hoffset
		49, 411
		hook commands:
		\hook_gput_code:nnn
		33, 50, 349, 354, 388, 424
		\hsize
		12, 33, 486, 487
	<b>I</b>	
		\ignorespaces
		493
		int commands:
		\int_case:nn
		471, 555
		\int_eval:n
		410, 508
		\int_gincr:N
		403, 445

<code>\int_if_zero:nTF</code>	40, 524, 526	<code>\l_marginalia_item_box</code>	33, 37, 433, 480, 500, 502, 648
<code>\int_new:N</code>	54, 62, 70, 128, 399, 432, 436, 437, 440, 441, 442	<code>\_marginalia_item_box_set:Nn</code>	473, 474, 475, 476, 480
<code>\int_set:Nn</code>	58, 66, 74, 405, 406	<code>\l_marginalia_item_depth_dim</code>	434, 501, 514
<code>\int_set_eq:NN</code>	132	<code>\l_marginalia_item_height_dim</code>	434, 499, 513
<code>\int_value:w</code>	409, 411, 412, 413, 414, 415, 416, 417, 418, 419, 449, 507, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521	<code>\g_marginalia_itemno_int</code>	432, 445, 449, 507
<code>\l_tmpa_int</code>	405, 406, 417	<code>\_marginalia_lua_check_item_data:n</code>	28, 293, 313, 360
iow commands:		<code>\_marginalia_lua_check_page_data:n</code>	28, 293, 301, 357
<code>\iow_now:Nn</code>	393, 407	<code>\_marginalia_lua_compute_items</code>	293, 317, 351
<code>\iow_shipout_e:Nn</code>	505	<code>\_marginalia_lua_load_item_data:n</code>	31, 329, 329, 448
		<code>\_marginalia_lua_store_default_page_data:</code>	293, 293, 350
		<code>\_marginalia_lua_store_item_data:n</code>	28, 293, 309, 348
		<code>\_marginalia_lua_store_page_data:n</code>	28, 293, 297, 341
		<code>\_marginalia_lua_write_item_change_report:</code>	293, 325, 376
		<code>\_marginalia_lua_write_page_change_report:</code>	305, 381
		<code>\_marginalia_lua_write_problem_report:</code>	293, 321, 371
		<code>\_marginalia_margin_bottom:</code>	141, 145, 187, 198
		<code>\_marginalia_margin_top:</code>	141, 141, 181, 197
		<code>\l_marginalia_marginno_computed_int</code>	35, 441, 555
		<code>\l_marginalia_mark_left_between_tl</code>	270, 621
		<code>\l_marginalia_mark_recto_inner_tl</code>	270, 577
		<code>\l_marginalia_mark_recto_outer_tl</code>	270, 566
		<code>\l_marginalia_mark_right_between_tl</code>	270, 610
		<code>\l_marginalia_mark_tl</code>	530, 532, 542, 544, 565, 576, 587, 598, 609, 620
		<code>\l_marginalia_mark_verso_inner_tl</code>	270, 599
		<code>\l_marginalia_mark_verso_outer_tl</code>	270, 588
		<code>\l_marginalia_nobreak_bool</code>	662, 665, 668
		<code>\l_marginalia_page_int</code>	32, 436, 489

**K**

<code>\kern</code>	529, 534, 541, 546
keys commands:	
<code>\l_keys_choice_int</code>	58, 66, 74, 132
<code>\keys_define:mn</code>	55, 63, 71, 84, 129, 136, 154, 206, 250, 270
<code>\keys_set:mn</code>	42, 47, 99, 109, 115, 121, 165, 171, 179, 185, 191, 221, 231, 237, 243, 259, 279, 447

**L**

<code>\lastxpos</code>	511
<code>\lastypos</code>	512
<code>\leavevmode</code>	458
legacy commands:	
<code>\legacy_if:nTF</code>	404, 457
<code>\legacy_if_gset:nn</code>	668
<code>\legacy_if_p:n</code>	420, 665
<code>\linewidth</code>	33, 487
list commands:	
<code>list_filter</code>	735, 752
<code>\llap</code>	4, 14
load commands:	
<code>load_item_data</code>	1452
lua commands:	
<code>\lua_now:n</code>	26, 27, 60, 290, 295, 299, 303, 307, 311, 315, 319, 323, 327, 331

**M**

<code>\marginalia</code>	4-9, 13, 14, 16-18, 20, 26, 31, 40, 670
marginalia internal commands:	
<code>\l_marginalia_aux_iow</code>	352, 393, 407, 505
<code>\l_marginalia_column_computed_int</code>	32, 437, 490
<code>\l_marginalia_column_int</code>	70, 516
<code>\l_marginalia_default_yshift_dim</code>	136, 517
<code>\l_marginalia_enabled_computed_int</code>	32, 442, 524

<code>\g__marginalia_pagedatano_int</code> ..	<code>\l__marginalia_width_dim</code> ....
..... 399, 403, 409	33,
<code>\__marginalia_place_item_box:</code> ..	35, 486, 561, 572, 583, 594, 605, 616
..... 34, 535, 540, 642, 642	<code>\l__marginalia_width_left_-</code>
<code>\l__marginalia_pos_int</code> .... 62, 515	<code>between_dim</code> ..... 200, 617
<code>\__marginalia_process_item:nn</code> ..	<code>\l__marginalia_width_recto_-</code>
..... 35, 38, 443, 443, 672	<code>inner_dim</code> ..... 200, 573
<code>\__marginalia_process_item_-</code>	<code>\l__marginalia_width_recto_-</code>
<code>data:n</code> ..... 28, 344, 347, 359	<code>outer_dim</code> ..... 200, 562
<code>\__marginalia_process_page_-</code>	<code>\l__marginalia_width_right_-</code>
<code>data:n</code> ..... 28, 337, 340, 356	<code>between_dim</code> ..... 200, 606
<code>\__marginalia_set_dim:Nn</code> . 20, 31,	<code>\l__marginalia_width_verso_-</code>
31, 87, 89, 91, 93, 95, 97, 157, 159,	<code>inner_dim</code> ..... 200, 595
161, 163, 209, 211, 213, 215, 217, 219	<code>\l__marginalia_width_verso_-</code>
<code>\__marginalia_set_xsep_width_-</code>	<code>outer_dim</code> ..... 200, 584
<code>style_mark:</code> ..... 478, 553, 553	<code>\__marginalia_write_page_data:</code> .
<code>\__marginalia_setup:n</code> .....	..... 30, 400, 400, 428, 430, 680
..... 20, 38, 49, 49, 52, 676	<code>\__marginalia_write_page_data_-</code>
<code>\__marginalia_setup_body:n</code> ....	<code>real:</code> ..... 401, 429
..... 20, 45, 45, 52	<code>\__marginalia_write_reports:</code> ...
<code>\__marginalia_setup_preamble:n</code> .	..... 362, 368, 389
..... 20, 38, 38, 49	<code>\__marginalia_write_version:</code> ...
<code>\l__marginalia_side_computed_int</code>	..... 391, 391, 426
..... 32, 34, 440, 526	<code>\l__marginalia_xsep_dim</code> .. 34, 35,
<code>\l__marginalia_style_left_-</code>	534, 541, 559, 570, 581, 592, 603, 614
<code>between_tl</code> ..... 250, 619	<code>\l__marginalia_xsep_left_-</code>
<code>\l__marginalia_style_recto_-</code>	<code>between_dim</code> ..... 78, 615
<code>inner_tl</code> ..... 250, 575	<code>\l__marginalia_xsep_recto_inner_-</code>
<code>\l__marginalia_style_recto_-</code>	<code>dim</code> ..... 78, 571
<code>outer_tl</code> ..... 250, 564	<code>\l__marginalia_xsep_recto_outer_-</code>
<code>\l__marginalia_style_right_-</code>	<code>dim</code> ..... 78, 560
<code>between_tl</code> ..... 250, 608	<code>\l__marginalia_xsep_right_-</code>
<code>\l__marginalia_style_tl</code> .....	<code>between_dim</code> ..... 78, 604
. 35, 485, 563, 574, 585, 596, 607, 618	<code>\l__marginalia_xsep_verso_inner_-</code>
<code>\l__marginalia_style_verso_-</code>	<code>dim</code> ..... 78, 593
<code>inner_tl</code> ..... 250, 597	<code>\l__marginalia_xsep_verso_outer_-</code>
<code>\l__marginalia_style_verso_-</code>	<code>dim</code> ..... 78, 582
<code>outer_tl</code> ..... 250, 586	<code>\l__marginalia_xshift_computed_-</code>
<code>\__marginalia_tagging_socket:n</code> .	<code>dim</code> ..... 32, 34, 438, 529, 546
..... 19, 14, 26, 479, 482, 498	<code>\l__marginalia_ysep_above_dim</code> ..
<code>\l__marginalia_type_int</code> .... 54, 510	..... 150, 518
<code>\__marginalia_typeset:n</code> .....	<code>\l__marginalia_ysep_below_dim</code> ..
..... 453, 462, 467, 503	..... 150, 519
<code>\__marginalia_typeset_hmmode:n</code> . 652	<code>\l__marginalia_ysep_page_bottom_-</code>
<code>\__marginalia_typeset_hmode:n</code> ..	<code>dim</code> ..... 150, 521
..... 463, 656	<code>\l__marginalia_ysep_page_top_dim</code>
<code>\__marginalia_typeset_mmode:n</code> ..	..... 150, 520
..... 454, 652, 652	<code>\l__marginalia_yshift_computed_-</code>
<code>\__marginalia_typeset_vmode:n</code> ..	<code>dim</code> ..... 32, 37, 438, 646
..... 468, 652, 663	<code>\marginaliacolumn</code> ..... 6, 31, 33, 490
<code>\l__marginalia_valign_int</code> 33, 128, 471	<code>\marginalianewgeometry</code> .... 6, 30, 58, 678
<code>\__marginalia_vbox_set_center:Nn</code> 625	<code>\marginaliapage</code> ..... 6, 31, 33, 489
<code>\__marginalia_vbox_set_midway:Nn</code> 632	<code>\marginaliasetup</code> ..... 6, 7, 13, 674
	<code>\marginpar</code> ..... 1, 3, 18
	<code>\marginparpush</code> ..... 7, 11, 196



