

# marginalia — Non-floating marginal content with automatic placement for Lua<sup>A</sup>T<sub>E</sub>X\*

Alan J. Cain

Released 2025-07-14

## Abstract

This Lua<sup>A</sup>T<sub>E</sub>X 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. In short, it tries to combine the best features from the packages `marginnote`, `marginfix` and `marginfit` with key–value settings that allow fine-grained customization.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Requirements</b>	<b>4</b>
<b>3</b>	<b>Installation</b>	<b>4</b>
<b>4</b>	<b>Getting started</b>	<b>4</b>
<b>5</b>	<b>User commands</b>	<b>4</b>
5.1	Access to page and column . . . . .	5
<b>6</b>	<b>Options</b>	<b>6</b>
6.1	Type . . . . .	6
6.2	Horizontal placement . . . . .	6
6.3	Vertical placement . . . . .	8
6.4	Appearance . . . . .	10
<b>7</b>	<b>Placement</b>	<b>10</b>
7.1	Horizontal placement . . . . .	11
7.2	Vertical placement . . . . .	12
<b>8</b>	<b>Usage notes</b>	<b>12</b>
<b>9</b>	<b>Incompatibilities</b>	<b>13</b>
<b>10</b>	<b>Limitations</b>	<b>13</b>

---

\*This file describes v0.81.2, last revised 2025-07-14.

<b>11</b>	<b>Implementation (L<sup>A</sup>T<sub>E</sub>X package)</b>	<b>15</b>
11.1	Initial set-up . . . . .	15
11.2	Tagging set-up . . . . .	15
11.3	Options . . . . .	15
11.3.1	Type . . . . .	16
11.3.2	Horizontal placement . . . . .	16
11.3.3	Vertical placement . . . . .	17
11.3.4	Appearance . . . . .	18
11.4	Lua backend and interface . . . . .	19
11.5	Processing data from the .aux file . . . . .	20
11.6	Writing page data to the .aux file . . . . .	22
11.7	Marginal content item processing . . . . .	23
11.7.1	Variables . . . . .	23
	Variables set by L <sup>A</sup> T <sub>E</sub> X. . . . .	23
	Variables set by Lua. . . . .	24
11.7.2	Core macro . . . . .	25
11.7.3	Width and style selection . . . . .	27
11.7.4	Auxiliary placement macros . . . . .	29
11.8	User commands . . . . .	29
<b>12</b>	<b>Implementation (Lua backend)</b>	<b>30</b>
12.1	Global variables . . . . .	30
12.2	Constants . . . . .	30
12.3	Keys for tables . . . . .	31
12.3.1	Keys for both page and item data tables . . . . .	31
12.3.2	Keys for page data tables, layout etc. . . . .	31
12.3.3	Keys for item data tables . . . . .	31
12.4	Utility functions . . . . .	33
12.5	Generic page/item data functions . . . . .	33
12.6	Processing of page data from .aux file . . . . .	35
12.7	Processing of item data from .aux file . . . . .	37
12.8	Writing reports . . . . .	38
12.9	Computing horizontal positions . . . . .	39
12.10	Computing vertical positions . . . . .	44
12.10.1	Computing <code>optfixed</code> enabled . . . . .	44
12.10.2	Computing vertical adjustment . . . . .	45
12.10.3	Checking vertical adjustment . . . . .	46
12.10.4	Core vertical position computation . . . . .	48
12.11	Passing item_data back to L <sup>A</sup> T <sub>E</sub> X . . . . .	51
12.12	Export public functions . . . . .	51

# 1 Introduction

The  $\text{\LaTeX}$  `\marginpar` command is the basic method for placing content in the margin. For purposes such as drawing attention to particular points in the text, it functions well. Its main limitation is that `\marginpar` works via the  $\text{\LaTeX}$  float mechanism and so cannot be used to create marginal content next to a figure, table, or other float, or next to a footnote, or to place running heads in the margin, such as are found in the left-hand margin of this document except for the ‘implementation’ section. (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,  $\text{\LaTeX}$  shifts the second item downward. But the cumulative effect can lead to `\marginpars` being shifted downward off the bottom of the page. Further, the asynchronous nature of  $\text{\TeX}$ ’s page-breaking can cause: (1) a `\marginpar` to be placed in the wrong margin; (2) the topmost `\marginpar` on a page to be unnecessarily shifted downward because of a hypothetical clash that would have occurred with the previous `\marginpar`, had they been on the same page.

Packages like `mparhack`<sup>1</sup> (Tom Sgouros & Stefan Ulrich), `marginnote`<sup>2</sup> (Markus Kohm), `marginfix`<sup>3</sup> (Stephen Hicks) and `marginfit`<sup>4</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 anywhere, but does not adjust positions to avoid clashes. `marginfix` adjusts positions, but the unadjusted vertical positioning can be slightly off, and the package still uses floats. `marginfit` gets positions exactly right, but uses the insert mechanism and so marginal content cannot appear next to floats or footnotes.

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

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

*Note:* `marginalia` was written to typeset running heads in the margin, sidenote references, side-captions for floats, and small marginal figures in the author’s book *Form & Number: A History of Mathematical Beauty* [Cai24].<sup>5</sup> Thus the basic functionality has been tested extensively, and it has performed correctly.

**Acknowledgements.** The author thanks Ulrike Fischer for explaining how to add tagging support.

**Licence.** `marginalia` is released under the  $\text{\LaTeX}$  Project Public Licence v1.3c or later.<sup>1</sup>

<sup>1</sup>URL: <https://www.latex-project.org/lppl.txt>

1 URL: <https://ctan.org/pkg/mparhack>  
 2 URL: <https://ctan.org/pkg/marginnote>  
 3 URL: <https://ctan.org/pkg/marginfix>  
 4 URL: <https://ctan.org/pkg/marginfit>

5 *Form & Number* is freely available on the Internet Archive under a Creative Commons licence.  
 URL: [https://archive.org/details/cain\\_formandnumber\\_ebook\\_large](https://archive.org/details/cain_formandnumber_ebook_large)

## 2 Requirements

`marginalia` requires

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

It does not depend on any other packages.

## 3 Installation

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

## 4 Getting started

`marginalia` works ‘out of the box’. Load the package (there are no package options) and use the main `\marginalia` command to place marginal content. Figure 4.1 shows the source code for a small demonstration and the resulting document. *The source code must be processed twice by Lua $\text{\LaTeX}$  for the marginal content to be placed correctly.* (See 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>`) than 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 Subsection 6.

---

`\marginaliasetup`    `\marginaliasetup{<options>}`

This command is used to set options for all 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.

---

`\marginalianewgeometry`    `\marginalianewgeometry`

This command signals to `marginalia` that the page layout has been changed, for instance by using the `\newgeometry` command from the `geometry` package,<sup>6</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.

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

## User commands

```
\documentclass[11pt]{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.}

Some final body text.\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}
```

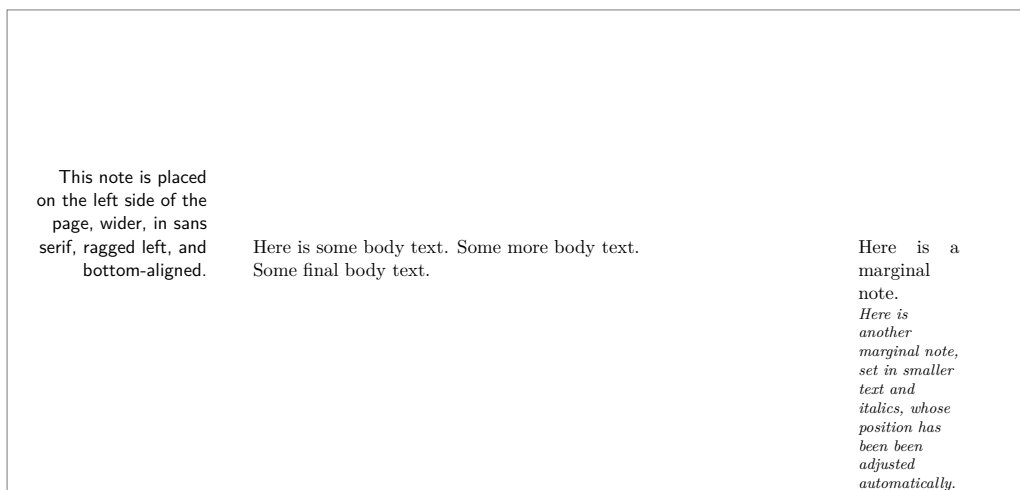


Figure 4.1: A small demonstration of marginalia.

## 5.1 Access to page and column

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

---

**`\marginaliapage`** A counter register, available within the `<content>` of `\marginalia`, that holds the actual page on which the marginal content appears. The value is based on the previous Lua<sup>A</sup>T<sub>E</sub>X run and will default to 1.

<hr/> <hr/> <code>\marginaliacolumn</code>	A counter register, available within the <code>\content</code> of <code>\marginalia</code> , 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 <code>column</code> is used to manually specify the column to which the content pertains, the value of <code>\marginaliacolumn</code> will change accordingly.) The value is based on the previous Lua <sup>A</sup> T <sub>E</sub> X run and will default to 0.
--	--

## 6 Options

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

### 6.1 Type

- type** The **type** of an item of marginal content can be set to one of the following three values:
- normal:** The vertical position of the item will be changed automatically if necessary to prevent a clash with another item of content.
  - fixed:** The vertical position of the item will *never* be changed automatically from the position specified by `yshift`, even if there is a clash with another item. (The type **fixed** was designed for setting float captions in the margin, since a caption should not move away from the float with which it is associated.)
  - optfixed:** The vertical position of the item will *never* be changed automatically from the position specified by `yshift`, even if there is a clash with another item. But an **optfixed** item will not appear in the document if it would clash with a **fixed** item. (The type **optfixed** was designed for setting running heads in the margin, which should not appear if they would clash with a figure caption set in the margin.)
- (Default: **normal**)

### 6.2 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**)

## Options

Table 1: Summary of keys that 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>\marginparwidth</code>
<code>xsep outer</code>	Dimension	<code>\marginparwidth</code>
<code>xsep inner</code>	Dimension	<code>\marginparwidth</code>
<code>xsep between</code>	Dimension	<code>\marginparwidth</code>
<code>xsep recto outer</code>	Dimension	<code>\marginparwidth</code>
<code>xsep recto inner</code>	Dimension	<code>\marginparwidth</code>
<code>xsep verso outer</code>	Dimension	<code>\marginparwidth</code>
<code>xsep verso inner</code>	Dimension	<code>\marginparwidth</code>
<code>xsep right between</code>	Dimension	<code>\marginparwidth</code>
<code>xsep left between</code>	Dimension	<code>\marginparwidth</code>
<code>valign</code>	<code>{t, b}</code>	<code>t</code>
<code>yshift</code>	Dimension	<code>0 pt</code>
<code>ysep</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	<code>\marginparpush</code>
<code>ysep page bot</code>	Dimension	<code>\marginparpush</code>
<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]

## Options

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

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

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

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

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

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

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

<code>xsep</code>	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 <a href="#">Figure 7.1</a> .
<code>xsep outer</code>	
<code>xsep inner</code>	
<code>xsep between</code>	<b>xsep recto outer:</b> used for an item in the outer margin of a recto page.
<code>xsep recto outer</code>	<b>xsep recto inner:</b> used for an item in the inner margin of a recto page.
<code>xsep recto inner</code>	<b>xsep verso outer:</b> used for an item in the outer margin of a verso page.
<code>xsep verso outer</code>	<b>xsep verso inner:</b> used for an item in the inner margin of a verso page.
<code>xsep verso inner</code>	<b>xsep right between:</b> used for an item set from the right column between the columns.
<code>xsep right between</code>	
<code>xsep left between</code>	<b>xsep left between:</b> used for an item set from the left column between the columns.
	<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.
	<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.
	<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` when the package is loaded)

## 6.3 Vertical placement

`valign` The option `valign` can be either `t` or `b`. In the former case, the baseline of the marginal content item is the baseline of the topmost box in its contents; in the latter case, its baseline is the baseline of the bottommost box in its contents. (Essentially, `\vtop` and `\vbox` are used to set the two options) (*Default: t*)



## Options

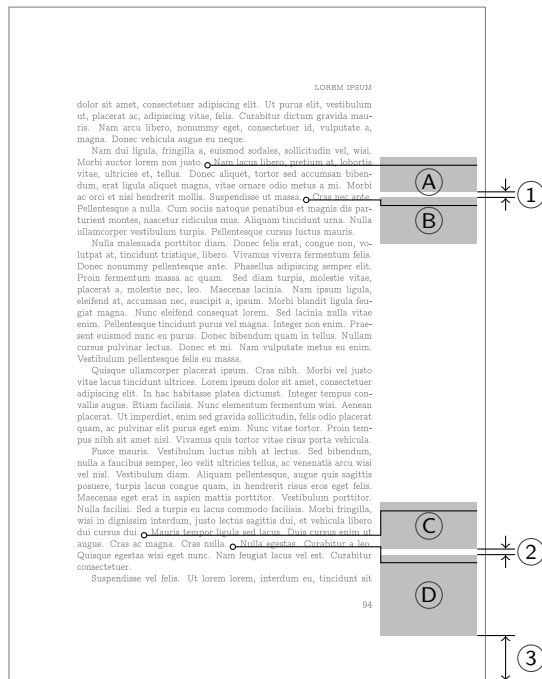


Figure 6.1: (Illustration of `ysep`) The length ① is at least the value of `ysep below` specified (locally or globally) for marginal content item ① and at least the value of `ysep above` specified for item ②. In this example diagram, ② has been automatically moved down from its natural position to maintain the required distance. Similarly, the length ② is at least the value of `ysep below` specified for ③ and at least the value of `ysep above` specified for ④, and the length ③ is at least the value of `ysep page bottom` specified for ④. In this example, to maintain the required distances, ③ and ④ have been automatically moved (respectively) up and down from their natural positions.

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

**ysep** These keys specify the minimum vertical separation above and below an item of marginal content

**ysep above** `ysep above`: the minimum vertical separation between an item and the one above.

**ysep below** `ysep below`: the minimum vertical separation between an item and the one below.

**ysep page top** `ysep page top`: the minimum vertical separation between an item and top of the page.

**ysep page bottom** `ysep page bottom`: the minimum vertical separation between an item and bottom of the page.

**ysep** `ysep` is a shorthand for setting all of these keys simultaneously to the same value. (See Figure 6.1.) Each of these keys must be set to a valid dimension. (*Default: value of `\marginparpush` when the package is loaded*).

## Placement 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.

<code>width</code> <code>width outer</code> <code>width inner</code> <code>width between</code> <code>width recto outer</code> <code>width recto inner</code> <code>width verso outer</code> <code>width verso inner</code> <code>width right between</code> <code>width left between</code>	<p>These keys specify the width of the an item of marginal content (or, more precisely, the <code>\hsize</code> of the box into which the item is typeset). Which width is chosen will depend on the where the item is typeset. The terminology is as in <a href="#">Figure 7.1</a>.</p> <p><b>width recto outer:</b> used for an item in the outer margin of a recto page.</p> <p><b>width recto inner:</b> used for an item in the inner margin of a recto page.</p> <p><b>width verso outer:</b> used for an item in the outer margin of a verso page.</p> <p><b>width verso inner:</b> used for an item in the inner margin of a verso page.</p> <p><b>width right between:</b> used for an item set from the right column and placed between the columns.</p> <p><b>width left between:</b> used for an item set from the right column and placed between the columns.</p> <p><b>width outer:</b> a shorthand for setting the keys <code>width recto outer</code> and <code>width verso outer</code> simultaneously to the same value.</p> <p><b>width inner:</b> a shorthand for setting the keys <code>width recto inner</code> and <code>width verso inner</code> simultaneously to the same value.</p> <p><b>width between:</b> a shorthand for setting the keys <code>width right between</code> and <code>width left between</code> simultaneously to the same value.</p> <p><b>width:</b> a shorthand for setting all of these keys simultaneously.</p> <p>(The shorthands <code>width outer</code> and <code>width inner</code> exist because page geometry is usually symmetrical between recto and verso pages as regards outer and inner margins. The shorthand <code>width between</code> exists because the space between columns, if used at all for marginal content, will often be shared equally.) Each of these keys must be set to a valid dimension. (<i>Default:</i> value of <code>\marginparwidth</code> when the package is loaded)</p>
---	--

<code>style</code> <code>style recto outer</code> <code>style recto inner</code> <code>style verso outer</code> <code>style verso inner</code> <code>style right between</code> <code>style left between</code>	<p>These keys specify the style with which an item of marginal content is typeset. Which style is chosen will depend on where the item is typeset. The terminology is as in <a href="#">Figure 7.1</a>.</p> <p><b>style recto outer:</b> used for an item in the outer margin of a recto page.</p> <p><b>style recto inner:</b> used for an item in the inner margin of a recto page.</p> <p><b>style verso outer:</b> used for an item in the outer margin of a verso page.</p> <p><b>style verso inner:</b> used for an item in the inner margin of a verso page.</p> <p><b>style right between:</b> used for an item set from the right column between the columns.</p> <p><b>style left between:</b> used for an item set from the right column between the columns.</p> <p><b>style:</b> a shorthand for setting all of these keys simultaneously.</p> <p>Each of these keys should be set to L<sup>A</sup>T<sub>E</sub>X code that specifies the style. (<i>Default:</i> [Empty])</p>
---	---

## 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 com-

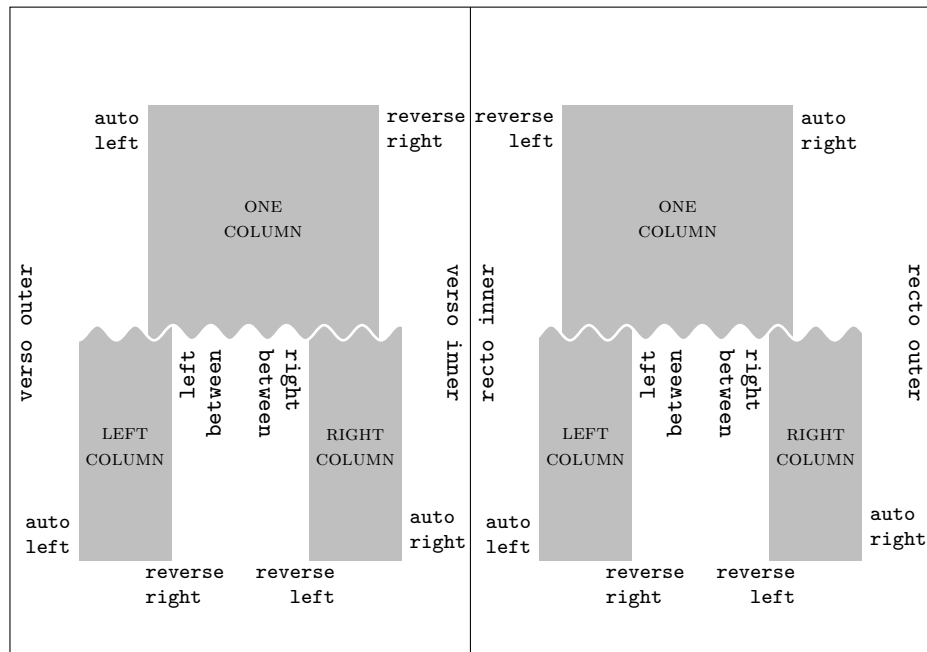


Figure 7.1: Summary of the positioning of marginal content using `pos`, and terminology used in `width` and `style` keys, on recto and verso pages, in both one-column and two-column mode.

plicated.

## 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`.

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

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

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

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

## 8 Usage notes

`marginalia` requires a minimum of two Lua<sup>A</sup>T<sub>E</sub>X runs, and often more, to place items of marginal content correctly. On the first pass, information about items, including their

## Limitations

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. Provided the main text stabilizes, the placement of items using `\marginalia` should be correct two runs later.

At the end of the Lua $\text{\LaTeX}$  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 run written to the `.aux` file on the previous run, and may not arise if item positions or sizes (i.e. height or depth) have changed. `marginalia` also reports any changes in positions or sizes compared to the previous run.

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

## 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 $\text{\LaTeX}$ 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  $\text{\LaTeX}$  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).

## 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}{2025-07-14}{0.81.2}
5 {Non-floating marginal content for LuaLaTeX}
```

Check that Lua<sub>T</sub><sub>E</sub>X 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 Options

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

### 11.3.1 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.

```

31 \int_new:N\l__marginalia_type_int
32 \keys_define:nn { marginalia }
33 {
34   type .choices:nn = {normal,fixed,optfixed}{
35     \int_set:Nn\l__marginalia_type_int{\l_keys_choice_int}
36   },
37   type .initial:n = normal,
38 }

```

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

### 11.3.2 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.

```

39 \int_new:N\l__marginalia_pos_int
40 \keys_define:nn { marginalia }
41 {
42   pos .choices:nn = {auto,reverse,left,right,nearest}{
43     \int_set:Nn\l__marginalia_pos_int{\l_keys_choice_int}
44   },
45   pos .initial:n = auto
46 }

```

*(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).

```

47 \int_new:N\l__marginalia_column_int
48 \keys_define:nn { marginalia }
49 {
50   column .choices:nn = {auto,one,left,right}{
51     \int_set:Nn\l__marginalia_column_int{\l_keys_choice_int-2}
52   },
53   column .initial:n = auto,
54 }

```

*(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.

```

55 \keys_define:nn { marginalia }
56 {
57   xsep~recto~outer .dim_set:N = \l__marginalia_xsep_recto_outer_dim,
58   xsep~recto~inner .dim_set:N = \l__marginalia_xsep_recto_inner_dim,

```



```

59 xsep~verso~outer .dim_set:N = \l__marginalia_xsep_verso_outer_dim,
60 xsep~verso~inner .dim_set:N = \l__marginalia_xsep_verso_inner_dim,
61 xsep~right~between .dim_set:N = \l__marginalia_xsep_right_between_dim,
62 xsep~left~between .dim_set:N = \l__marginalia_xsep_left_between_dim,
63 xsep .code:n = {
64   \keys_set:nn{ marginalia }{
65     xsep~recto~outer=#1,
66     xsep~recto~inner=#1,
67     xsep~verso~outer=#1,
68     xsep~verso~inner=#1,
69     xsep~right~between=#1,
70     xsep~left~between=#1,
71   }
72 },
73 xsep~outer .code:n = {
74   \keys_set:nn{ marginalia }{
75     xsep~recto~outer=#1,
76     xsep~verso~outer=#1,
77   }
78 },
79 xsep~inner .code:n = {
80   \keys_set:nn{ marginalia }{
81     xsep~recto~inner=#1,
82     xsep~verso~inner=#1,
83   }
84 },
85 xsep~between .code:n = {
86   \keys_set:nn{ marginalia }{
87     xsep~right~between=#1,
88     xsep~left~between=#1,
89   }
90 },
91 xsep .initial:n = \marginparsep,
92 }

```

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

### 11.3.3 Vertical placement

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

```

93 \int_new:N\l__marginalia_valign_int
94 \keys_define:nn { marginalia }
95 {
96   valign .choices:nn = {t,b}{
97     \int_set_eq:NN\l__marginalia_valign_int\l_keys_choice_int
98   },
99   valign .initial:n = t,
100 }

```

*(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.

```

101 \keys_define:nn { marginalia }
102 {
103   yshift .dim_set:N = \l__marginalia_default_yshift_dim,
104   yshift .initial:n = 0pt,
105 }

```

(End of definition for \l\_\_marginalia\_default\_yshift\_dim.)

\l\_\_marginalia\_ysep\_above\_dim    Dimension keys to hold the the minimum vertical spacing between a marginal content  
 \l\_\_marginalia\_ysep\_below\_dim    item and (respectively) the item above, the item below, the page top, and the page  
 \l\_\_marginalia\_ysep\_page\_top\_dim    bottom.  
 \l\_\_marginalia\_ysep\_page\_bottom\_dim

```

106 \keys_define:nn { marginalia }
107 {
108   ysep~above .dim_set:N = \l__marginalia_ysep_above_dim,
109   ysep~below .dim_set:N = \l__marginalia_ysep_below_dim,
110   ysep~page~top .dim_set:N = \l__marginalia_ysep_page_top_dim,
111   ysep~page~bottom .dim_set:N = \l__marginalia_ysep_page_bottom_dim,
112   ysep .code:n = {
113     \keys_set:nn{ marginalia }{
114       ysep~below=#1,
115       ysep~above=#1,
116       ysep~page~top=#1,
117       ysep~page~bottom=#1,
118     }
119   },
120   ysep .initial:n = \marginparpush,
121 }

```

(End of definition for \l\_\_marginalia\_ysep\_above\_dim and others.)

### 11.3.4 Appearance

\l\_\_marginalia\_width\_recto\_outer\_dim    Dimension keys to hold the width of the marginal content item, which can be dependent  
 \l\_\_marginalia\_width\_recto\_inner\_dim    on where it appears on the page.  
 \l\_\_marginalia\_width\_verso\_outer\_dim  
 \l\_\_marginalia\_width\_verso\_inner\_dim  
 \l\_\_marginalia\_width\_right\_between\_dim  
 \l\_\_marginalia\_width\_left\_between\_dim

```

122 \keys_define:nn { marginalia }
123 {
124   width~recto~outer .dim_set:N = \l__marginalia_width_recto_outer_dim,
125   width~recto~inner .dim_set:N = \l__marginalia_width_recto_inner_dim,
126   width~verso~outer .dim_set:N = \l__marginalia_width_verso_outer_dim,
127   width~verso~inner .dim_set:N = \l__marginalia_width_verso_inner_dim,
128   width~right~between .dim_set:N = \l__marginalia_width_right_between_dim,
129   width~left~between .dim_set:N = \l__marginalia_width_left_between_dim,
130   width .code:n = {
131     \keys_set:nn{ marginalia }{
132       width~recto~outer=#1,
133       width~recto~inner=#1,
134       width~verso~outer=#1,
135       width~verso~inner=#1,
136       width~right~between=#1,
137       width~left~between=#1,
138     }
139   },
140   width~outer .code:n = {
141     \keys_set:nn{ marginalia }{

```

```

142     width~recto~outer=#1,
143     width~verso~outer=#1,
144   }
145 },
146 width~inner .code:n = {
147   \keys_set:nn{ marginalia }{
148     width~recto~inner=#1,
149     width~verso~inner=#1,
150   }
151 },
152 width~between .code:n = {
153   \keys_set:nn{ marginalia }{
154     width~right~between=#1,
155     width~left~between=#1,
156   }
157 },
158 width .initial:n = \marginparwidth,
159 }

```

(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.

```

160 \keys_define:nn { marginalia }
161 {
162   style~recto~outer .tl_set:N = \l__marginalia_style_recto_outer_tl,
163   style~recto~inner .tl_set:N = \l__marginalia_style_recto_inner_tl,
164   style~verso~outer .tl_set:N = \l__marginalia_style_verso_outer_tl,
165   style~verso~inner .tl_set:N = \l__marginalia_style_verso_inner_tl,
166   style~right~between .tl_set:N = \l__marginalia_style_right_between_tl,
167   style~left~between .tl_set:N = \l__marginalia_style_left_between_tl,
168   style .code:n = {
169     \keys_set:nn{ marginalia }{
170       style~recto~outer=#1,
171       style~recto~inner=#1,
172       style~verso~outer=#1,
173       style~verso~inner=#1,
174       style~right~between=#1,
175       style~left~between=#1,
176     }
177   },
178   style .initial:n = {},
179 }

```

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

## 11.4 Lua backend and interface

Load the Lua backend.

```

180 \lua_now:n{
181   marginalia = require('marginalia')
182 }

```

The following 9 macros interface between  $\text{\LaTeX}$  and Lua code. Each control sequence `\__marginalia_lua_XYZ` simply calls the corresponding Lua function `marginalia.XYZ`.

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

```

\marginalia_lua_store_page_data:n
\marginalia_lua_check_page_data:n
\marginalia_lua_store_item_data:n
\marginalia_lua_check_item_data:n
\marginalia_lua_compute_items:
\marginalia_lua_write_problem_report:
\marginalia_lua_write_item_change_report:

183 \cs_new:Npn\_marginalia_lua_store_default_page_data:
184 {
185   \lua_now:n{ marginalia.store_default_page_data() }
186 }
187 \cs_new:Npn\_marginalia_lua_store_page_data:n #1
188 {
189   \lua_now:n{ marginalia.store_page_data('#1') }
190 }
191 \cs_new:Npn\_marginalia_lua_check_page_data:n #1
192 {
193   \lua_now:n{ marginalia.check_page_data('#1') }
194 }
195 \cs_new:Npn\_marginalia_lua_write_page_change_report:
196 {
197   \lua_now:n{ marginalia.write_page_change_report() }
198 }
199 \cs_new:Npn\_marginalia_lua_store_item_data:n #1
200 {
201   \lua_now:n{ marginalia.store_item_data('#1') }
202 }
203 \cs_new:Npn\_marginalia_lua_check_item_data:n #1
204 {
205   \lua_now:n{ marginalia.check_item_data('#1') }
206 }
207 \cs_new:Npn\_marginalia_lua_compute_items:
208 {
209   \lua_now:n{ marginalia.compute_items() }
210 }
211 \cs_new:Npn\_marginalia_lua_write_problem_report:
212 {
213   \lua_now:n{ marginalia.write_problem_report() }
214 }
215 \cs_new:Npn\_marginalia_lua_write_item_change_report:
216 {
217   \lua_now:n{ marginalia.write_item_change_report() }
218 }

```

*(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.

```

219 \cs_new:Npn\_marginalia_lua_load_item_data:n #1
220 {
221   \lua_now:e{ marginalia.load_item_data('#1') }
222 }

```

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

## 11.5 Processing data from the .aux file

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

```

223 \NewDocumentCommand{\marginalia@pagedata}{ m }{
224   \__marginalia_process_page_data:n{#1}
225 }

```

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.

```

226 \cs_set_eq:NN
227   \__marginalia_process_page_data:n
228   \__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.

```

229 \DeclareDocumentCommand{\marginalia@itemdata}{ m }{
230   \__marginalia_process_item_data:n{#1}
231 }

```

*(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.

```

232 \cs_set_eq:NN
233   \__marginalia_process_item_data:n
234   \__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.

```

235 \AddToHook{begindocument}{
236   \__marginalia_lua_store_default_page_data:
237   \__marginalia_lua_compute_items:
238   \cs_set_eq:NN\l__marginalia_aux_iow\@mainaux
239 }

```

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.

```

240 \AddToHook{enddocument/afterlastpage}{
241   \cs_set_eq:NN
242     \__marginalia_process_page_data:n
243     \__marginalia_lua_check_page_data:n
244   \cs_set_eq:NN
245     \__marginalia_process_item_data:n
246     \__marginalia_lua_check_item_data:n
247 }

```

`\__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:

```

248 \msg_new:nnn{marginalia}{placement_problem}

```

```

249 { Problems~in~placement.~#1 }
250 \msg_new:nnn{marginalia}{item_change}
251 { Changes~in~item~data.~#1 }
252 \msg_new:nnn{marginalia}{page_change}
253 { Changes~in~page~data.~#1 }
254 \cs_new:Npn\__marginalia_write_reports:
255 {
256   \group_begin:
257   \tl_set:N\l_tmpa_tl{\__marginalia_lua_write_problem_report:}
258   \tl_if_blank:VF\l_tmpa_tl
259   {
260     \msg_warning:nne{marginalia}{placement_problem}{\tl_use:N\l_tmpa_tl}
261   }
262   \tl_set:N\l_tmpa_tl{\__marginalia_lua_write_item_change_report:}
263   \tl_if_blank:VF\l_tmpa_tl
264   {
265     \msg_warning:nne{marginalia}{item_change}{\tl_use:N\l_tmpa_tl}
266   }
267   \tl_set:N\l_tmpa_tl{\__marginalia_lua_write_page_change_report:}
268   \tl_if_blank:VF\l_tmpa_tl
269   {
270     \msg_warning:nne{marginalia}{page_change}{\tl_use:N\l_tmpa_tl}
271   }
272   \group_end:
273 }

```

*(End of definition for \\_\_marginalia\_write\_reports:.)*

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

```

274 \AddToHook{enddocument/info}{
275   \__marginalia_write_reports:
276 }

```

## 11.6 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.

```

277 \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`).

```

278 \cs_set_eq:NN\__marginalia_write_page_data:\prg_do_nothing:
279 \cs_new:Npn\__marginalia_write_page_data_real:
280 {
281   \int_gincr:N\g__marginalia_pagedatano_int
282   \iow_now:N\l__marginalia_aux_iow{
283     \token_to_str:N\marginalia@pagedata{

```

```

284         pagedatano=\int_value:w\g__marginalia_pagedatano_int,
285         abspageno=\int_eval:n{\g_shipout_readonly_int+1},
286         hoffset=\int_value:w\hoffset,
287         voffset=\int_value:w\voffset,
288         paperheight=\int_value:w\paperheight,
289         oddsidemargin=\int_value:w\oddsidemargin,
290         evensidemargin=\int_value:w\evensidemargin,
291         textwidth=\int_value:w\textwidth,
292         columncount=\int_value:w\col@number,
293         columnwidth=\int_value:w\columnwidth,
294         columnsep=\int_value:w\columnsep,
295         twoside=\bool_to_str:n{\legacy_if_p:n{@twoside}},
296     }
297 }
298 }

```

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.

```

299 \AddToHook{begindocument/end}
300 {
301     \cs_set_eq:NN
302     \__marginalia_write_page_data:
303     \__marginalia_write_page_data_real:
304     \__marginalia_write_page_data:
305 }

```

*(End of definition for \\_\_marginalia\_write\_page\_data.)*

## 11.7 Marginal content item processing

### 11.7.1 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.

```

306 \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.

```

307 \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
308 \dim_new:N\l__marginalia_item_height_dim

```

```

309 \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.

<code>\l__marginalia_page_int</code>	Integer variable for the page on which the marginal content item appears. This variable will be made available via <code>\marginaliapage</code> within the <code>\content</code> of <code>\marginalia</code> . <code>310 \int_new:N\l__marginalia_page_int</code> <i>(End of definition for \l__marginalia_page_int.)</i>
<code>\l__marginalia_column_computed_int</code>	Integer variable for the column next to which the marginal content item appears. This variable will be made available via <code>\marginaliacolumn</code> within the <code>\content</code> of <code>\marginalia</code> . <code>311 \int_new:N\l__marginalia_column_computed_int</code> <i>(End of definition for \l__marginalia_column_computed_int.)</i>
<code>\l__marginalia_xshift_computed_dim</code> <code>\l__marginalia_yshift_computed_dim</code>	Dimension variables to hold the differences in $x$ and $y$ coordinates between the call to <code>\marginalia</code> and the position where the marginal content item should appear. <code>312 \dim_new:N\l__marginalia_xshift_computed_dim</code> <code>313 \dim_new:N\l__marginalia_yshift_computed_dim</code> <i>(End of definition for \l__marginalia_xshift_computed_dim and \l__marginalia_yshift_computed_dim.)</i>
<code>\l__marginalia_side_computed_int</code>	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. <code>314 \int_new:N\l__marginalia_side_computed_int</code> (This variable could be a boolean, but an integer is used because there is no canonical access to booleans from Lua.) <i>(End of definition for \l__marginalia_side_computed_int.)</i>
<code>\l__marginalia_marginno_computed_int</code>	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. <code>315 \int_new:N\l__marginalia_marginno_computed_int</code> <i>(End of definition for \l__marginalia_marginno_computed_int.)</i>
<code>\l__marginalia_enabled_computed_int</code>	Integer variable to indicate whether the marginal content item is enabled: 0 = disabled, 1 = enabled. <code>316 \int_new:N\l__marginalia_enabled_computed_int</code> (This variable could be a boolean, but an integer is used because there is no canonical access to booleans from Lua.) <i>(End of definition for \l__marginalia_enabled_computed_int.)</i>



### 11.7.2 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>`.

```
317 \cs_new:Npn\_marginalia_process_item:nn #1#2
318 {
```

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

```
319 \int_gincr:N\g__marginalia_itemno_int
320 \group_begin:
```

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

```
321 \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.

```
322 \__marginalia_lua_load_item_data:n
323 { \int_value:w\g__marginalia_itemno_int }
```

Choose the correct auxiliary function for typesetting, depending on which mode T<sub>E</sub>X is in.

```
324 \mode_if_math:TF
325 {
326   \cs_set_eq:NN
327     \__marginalia_typeset:n
328     \__marginalia_typeset_mmode:n
329 }
330 {
331   \legacy_if:nT{@inlabel}
332   { \leavevmode }
333   \mode_if_horizontal:TF
334   {
335     \cs_set_eq:NN
336       \__marginalia_typeset:n
337       \__marginalia_typeset_hmode:n
338   }
339   {
340     \cs_set_eq:NN
341       \__marginalia_typeset:n
342       \__marginalia_typeset_vmode:n
343   }
344 }
```

Choose the correct box in which to typeset the item. `\l__marginalia_valign_int` can only be 1 or 2, so take 2 to signify bottom-aligned, anything else signifies top-aligned.

```
345 \int_compare:nNnTF{\l__marginalia_valign_int}={2}
346 {
347   \cs_set_eq:NN\_marginalia_item_box_set:Nn\ vbox_set:Nn
348 }
```

```

349     {
350         \cs_set_eq:NN\__marginalia_item_box_set:Nn\ vbox_set_top:Nn
351     }

```

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

```

352     \__marginalia_set_xsep_width_style:

```

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

```

353     \__marginalia_tagging_socket:n {marginpar/begin}
354     \__marginalia_item_box_set:Nn\l__marginalia_item_box{
355         \@parboxrestore
356         \__marginalia_tagging_socket:n {para/restore}
357         \normalfont\normalsize
358
359         \tl_use:N\l__marginalia_style_tl
360         \dim_set_eq:NN\hsize\l__marginalia_width_dim
361         \dim_set_eq:NN\linewidth\hsize
362
363         \cs_set_eq:NN\marginaliapage\l__marginalia_page_int
364         \cs_set_eq:NN\marginaliacolumn\l__marginalia_column_computed_int
365
366         \group_begin:
367         \ignorespaces
368         #2
369         \par
370         \group_end:
371     }
372     \__marginalia_tagging_socket:n{marginpar/end}

```

Measure `\l__marginalia_item_box`.

```

373     \dim_set:Nn\l__marginalia_item_height_dim
374         {\box_ht:N\l__marginalia_item_box}
375     \dim_set:Nn\l__marginalia_item_depth_dim
376         {\box_dp:N\l__marginalia_item_box}

```

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

```

377     \__marginalia_typeset:n{
378         \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.

```

379     \iow_shipout_e:Ne\l__marginalia_aux_iow{
380         \token_to_str:N\marginalia@itemdata{
381             itemno=\int_value:w\g__marginalia_itemno_int,
382             abspageno=\exp_not:N\int_eval:n{\g_shipout_readonly_int},
383             pageno=\exp_not:N\int_value:w\c@page,
384             type=\str_use:N\int_value:w\l__marginalia_type_int,
385             xpos=\exp_not:N\int_value:w\lastxpos,
386             ypos=\exp_not:N\int_value:w\lastypos,
387             height=\int_value:w\l__marginalia_item_height_dim,

```

```

388         depth=\int_value:w\l__marginalia_item_depth_dim,
389         pos=\int_value:w\l__marginalia_pos_int,
390         column=\int_value:w\l__marginalia_column_int,
391         yshift=\int_value:w\l__marginalia_default_yshift_dim,
392         ysep~above=\int_value:w\l__marginalia_ysep_above_dim,
393         ysep~below=\int_value:w\l__marginalia_ysep_below_dim,
394         ysep~page~top=\int_value:w\l__marginalia_ysep_page_top_dim,
395         ysep~page~bottom=\int_value:w\l__marginalia_ysep_page_bottom_dim,
396     }
397 }

```

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

$$|\backslash l\_marginalia\_xshift\_computed\_dim| + |\backslash l\_marginalia\_xsep\_dim|$$

to the right in an `\rlap` or to the left in an `\llap`, depending on `\l__marginalia_side_computed_int`, then use `\__marginalia_place_item_box` for the vertical placement.

```

398     \int_if_zero:nF{\l__marginalia_enabled_computed_int}
399     {
400         \int_if_zero:nTF{\l__marginalia_side_computed_int}
401         {
402             \rlap{
403                 \kern\l__marginalia_xshift_computed_dim
404                 \kern\l__marginalia_xsep_dim
405                 \__marginalia_place_item_box:
406             }
407         }
408         {
409             \llap{
410                 \__marginalia_place_item_box:
411                 \kern\l__marginalia_xsep_dim
412                 \kern-\l__marginalia_xshift_computed_dim
413             }
414         }
415     }
416 }

```

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

```

417     \group_end:
418 }

```

*(End of definition for \\_\_marginalia\_process\_item:nn.)*

### 11.7.3 Width and style selection

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

```

419 \cs_new:Npn\__marginalia_set_xsep_width_style:
420 {
421     \int_case:nn{\l__marginalia_marginno_computed_int}
422     {
423         {0}
424         {
425             \cs_set_eq:NN\l__marginalia_xsep_dim
426             \l__marginalia_xsep_recto_outer_dim

```

```

427         \cs_set_eq:NN\l__marginalia_width_dim
428         \l__marginalia_width_recto_outer_dim
429         \cs_set_eq:NN\l__marginalia_style_tl
430         \l__marginalia_style_recto_outer_tl
431     }
432 {1}
433 {
434     \cs_set_eq:NN\l__marginalia_xsep_dim
435     \l__marginalia_xsep_recto_inner_dim
436     \cs_set_eq:NN\l__marginalia_width_dim
437     \l__marginalia_width_recto_inner_dim
438     \cs_set_eq:NN\l__marginalia_style_tl
439     \l__marginalia_style_recto_inner_tl
440 }
441 {2}
442 {
443     \cs_set_eq:NN\l__marginalia_xsep_dim
444     \l__marginalia_xsep_verso_outer_dim
445     \cs_set_eq:NN\l__marginalia_width_dim
446     \l__marginalia_width_verso_outer_dim
447     \cs_set_eq:NN\l__marginalia_style_tl
448     \l__marginalia_style_verso_outer_tl
449 }
450 {3}
451 {
452     \cs_set_eq:NN\l__marginalia_xsep_dim
453     \l__marginalia_xsep_verso_inner_dim
454     \cs_set_eq:NN\l__marginalia_width_dim
455     \l__marginalia_width_verso_inner_dim
456     \cs_set_eq:NN\l__marginalia_style_tl
457     \l__marginalia_style_verso_inner_tl
458 }
459 {4}
460 {
461     \cs_set_eq:NN\l__marginalia_xsep_dim
462     \l__marginalia_xsep_right_between_dim
463     \cs_set_eq:NN\l__marginalia_width_dim
464     \l__marginalia_width_right_between_dim
465     \cs_set_eq:NN\l__marginalia_style_tl
466     \l__marginalia_style_right_between_tl
467 }
468 {5}
469 {
470     \cs_set_eq:NN\l__marginalia_xsep_dim
471     \l__marginalia_xsep_left_between_dim
472     \cs_set_eq:NN\l__marginalia_width_dim
473     \l__marginalia_width_left_between_dim
474     \cs_set_eq:NN\l__marginalia_style_tl
475     \l__marginalia_style_left_between_tl
476 }
477 }
478 }

```

(End of definition for \\_\_marginalia\_set\_xsep\_width\_style.)

### 11.7.4 Auxiliary 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.

```

479 \cs_new:Npn\_marginalia_place_item_box:
480 {
481   \smash
482   {
483     \box_move_up:nn{\l\_marginalia_yshift_computed_dim}
484     {
485       \box_use:N\l\_marginalia_item_box
486     }
487   }
488 }
```

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

`\_marginalia_typeset_mmode:n` These three macros handle typesetting in math mode, horizontal mode, and vertical mode. Nothing special needs to be done in math mode. In horizontal mode, `\@bsphack...\@esphack` avoids double spacing. In vertical mode, a new paragraph containing only a `\strut` is started, the item is typeset, the paragraph is ended, and then a vertical skip of `-\baselineskip` should ‘hide’ that invisible paragraph.

`\_marginalia_typeset_hmode:n`  
`\_marginalia_typeset_vmode:n`

```

489 \cs_new:Npn\_marginalia_typeset_mmode:n #1
490 {
491   #1
492 }
493 \cs_new:Npn\_marginalia_typeset_hmode:n #1
494 {
495   \@bsphack
496   #1
497   \@esphack
498 }
499 \cs_new:Npn\_marginalia_typeset_vmode:n #1
500 {
501   \nobreak\noindent\strut #1\par
502   \skip_vertical:n{-\baselineskip}
503 }
```

*(End of definition for `\_marginalia_typeset_mmode:n`, `\_marginalia_typeset_hmode:n`, and `\_marginalia_typeset_vmode:n`.)*

## 11.8 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`.

```

504 \NewDocumentCommand{\marginalia}{0}{+m}
505 {
506   \_marginalia_process_item:nn{#1}{#2}
507 }
```

*(End of definition for `\marginalia`. This function is documented on page 4.)*

`\marginaliasetup` The user command to set the configuration.

```

508 \NewDocumentCommand{\marginaliasetup}{ m }
509 {
510   \keys_set:nn{marginalia}{ #1 }
511 }

```

*(End of definition for \marginaliasetup. This function is documented on page 4.)*

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

```

512 \NewDocumentCommand{\marginalianewgeometry}{}
513 {
514   \__marginalia_write_page_data:
515 }

```

*(End of definition for \marginalianewgeometry. This function is documented on page 4.)*

```

516 \endpackage

```

## 12 Implementation (Lua backend)

```

517 \lua

```

### 12.1 Global variables

Global tables for `page_data` and `item_data`.

```

518 local PAGE_DATA_MAIN_TABLE = {}
519 local ITEM_DATA_MAIN_TABLE = {}

```

Global tables for compiling reports.

```

520 local PROBLEM_REPORT_TABLE = {}
521 local PAGE_CHANGE_REPORT_TABLE = {}
522 local ITEM_CHANGE_REPORT_TABLE = {}

```

Global configuration for reports.

```

523 local PROBLEM_REPORT_MAX_LENGTH = 40
524 local PAGE_CHANGE_REPORT_MAX_LENGTH = 10
525 local ITEM_CHANGE_REPORT_MAX_LENGTH = 10

```

### 12.2 Constants

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

```

526 local TYPE_NORMAL = 1
527 local TYPE_FIXED = 2
528 local TYPE_OPTFIXED = 3

```

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

```

529 local POS_AUTO = 1
530 local POS_REVERSE = 2
531 local POS_LEFT = 3
532 local POS_RIGHT = 4
533 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 TeX 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_data` and `item_data` tables

```
534 local KEY_ABSPAGENO = 'abspageno'
```

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

```
535 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

```
536 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.

```
537 local KEY_HOFFSETORIGIN = 'hoffsetorigin'
```

```
538 local KEY_VOFFSETORIGIN = 'voffsetorigin'
```

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

```
539 local KEY_HOFFSET = 'hoffset'
```

```
540 local KEY_VOFFSET = 'voffset'
```

```
541 local KEY_PAPERHEIGHT = 'paperheight'
```

```
542 local KEY_ODDSIDEMARGIN = 'oddsidemargin'
```

```
543 local KEY_EVENSIDEMARGIN = 'evensidemargin'
```

```
544 local KEY_TEXTWIDTH = 'textwidth'
```

```
545 local KEY_COLUMNWIDTH = 'columnwidth'
```

```
546 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

```
547 local KEY_COLUMNCOUNT = 'columncount'
```

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

```
548 local KEY_TWOSIDE = 'twoside'
```

### 12.3.3 Keys for item data tables

Integer: Used to identify data with item

```
549 local KEY_ITEMNO = 'itemno'
```

Integer: On-page number

```
550 local KEY_PAGENO = 'pageno'
```

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

```
551 local KEY_XPOS = 'xpos'
```

```
552 local KEY_YPOS = 'ypos'
```

Dimensions: Height and depth of typeset item

```
553 local KEY_HEIGHT = 'height'
554 local KEY_DEPTH = 'depth'
```

Integer: Specified type, following TYPE\_\*

```
555 local KEY_TYPE = 'type'
```

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

```
556 local KEY_POS = 'pos'
```

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

```
557 local KEY_COLUMN = 'column'
```

Dimension: specified vertical shift

```
558 local KEY_YSHIFT = 'yshift'
```

Dimensions: specified vertical separations

```
559 local KEY_YSEP_ABOVE = 'ysep above'
560 local KEY_YSEP_BELOW = 'ysep below'
561 local KEY_YSEP_PAGE_TOP = 'ysep page top'
562 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

```
563 local KEY_COLNO_COMPUTED = 'colno computed'
```

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

```
564 local KEY_XSHIFT_COMPUTED = 'xshift computed'
```

Dimension: Computed vertical shift

```
565 local KEY_YSHIFT_COMPUTED = 'yshift computed'
```

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

```
566 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

```
567 local KEY_MARGINNO_COMPUTED = 'marginno computed'
```

Boolean: Whether the item will actually appear on the page

```
568 local KEY_ENABLED_COMPUTED = 'enabled computed'
```



## 12.4 Utility functions

`list_filter`  
 7 Code adapted from  
<https://stackoverflow.com/a/53038524/8990243>.

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>7</sup>

```
569 local function list_filter(t, f)
570   local j = 1
571   local n = #t
572
573   for i=1,n do
574     if (f(t[i])) then
575       if (i ~= j) then
576         t[j] = t[i]
577         t[i] = nil
578       end
579       j = j + 1
580     else
581       t[i] = nil
582     end
583   end
584
585 end
```

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

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

```
586 local function toboolean(s)
587   return s == "true"
588 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.

```
589 local function get_data_page_number(data)
590   local pageno = data[KEY_PAGENO]
591   if pageno ~= nil then
592     return 'p' .. pageno .. ' (' .. data[KEY_ABSPAGENO] .. ')'
593   else
594     return data[KEY_ABSPAGENO]
595   end
596 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`.

```

597 local function parse_data(keyvalue_string,conversion_table,defaults_table)
598
599     local key
600     local value
601     local result = {}
602
603     for s in string.gmatch(keyvalue_string,'([^\,]+)') do
604
605         key,value = string.match(s,'^(.+)=(.+)$')
606         local conv = conversion_table[key]
607         if conv ~= nil then
608             result[key] = conv(value)
609         end
610
611     end
612
613     for key,value in pairs(defaults_table) do
614         if not(result[key] ~= nil) then
615             result[key] = value
616         end
617     end
618
619     return result
620
621 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`.

```

622 local function check_data(keyvalue_string,conversion_table,defaults_table,
623                           data_table,data_table_key_field,report_table)
624
625     local new_data = parse_data(keyvalue_string,
626                                 conversion_table,defaults_table)
627
628     local data_table_key = new_data[data_table_key_field]
629
630     local stored_data = data_table[data_table_key]
631     if stored_data == nil then
632         table.insert(
633             report_table,
634             get_data_page_number(new_data) .. ' New'
635         )
636     else
637         local change_report = ''

```

```

638     for k,_ in pairs(conversion_table) do
639         if stored_data[k] ~= new_data[k] then
640             change_report = change_report
641             .. ' ' .. k .. ':' ..
642             tostring(stored_data[k]) .. '->' .. tostring(new_data[k])
643         end
644     end
645     if change_report ~= '' then
646         table.insert(
647             report_table,
648             get_data_page_number(new_data) .. ' ' .. change_report
649         )
650     end
651     stored_data[KEY_CHECKED] = true
652 end
653
654 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`.

```

655 local function check_removed_data(data_table,report_table)
656     for _,data in pairs(data_table) do
657         if not data[KEY_CHECKED] then
658             table.insert(
659                 report_table,
660                 ' Removed'
661             )
662             break
663         end
664     end
665 end

```

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

## 12.6 Processing of page data from .aux file

Conversion and default tables.

```

666 local PAGE_DATA_CONVERSION_TABLE = {
667     [KEY_PAGEDATANO] = tonumber,
668     [KEY_A BSPAGENO] = tonumber,
669     [KEY_HOFFSETORIGIN] = tonumber,
670     [KEY_VOFFSETORIGIN] = tonumber,
671     [KEY_HOFFSET] = tonumber,
672     [KEY_VOFFSET] = tonumber,
673     [KEY_PAPERHEIGHT] = tonumber,
674     [KEY_ODDSIDEMARGIN] = tonumber,
675     [KEY_EVENSIDEMARGIN] = tonumber,
676     [KEY_COLUMNCOUNT] = tonumber,
677     [KEY_COLUMNWIDTH] = tonumber,
678     [KEY_COLUMNSEP] = tonumber,
679     [KEY_TEXTWIDTH] = tonumber,

```

```

680 [KEY_TWOSIDE] = toboolean,
681 }
682 local PAGE_DATA_DEFAULT_TABLE = {
683 [KEY_PAGEDATANO] = 0,
684 [KEY_A BSPAGENO] = 0,
685 [KEY_HOFFSETORIGIN] = tex.sp('1in'),
686 [KEY_VOFFSETORIGIN] = tex.sp('1in'),
687 [KEY_HOFFSET] = tex.dimen['hoffset'],
688 [KEY_VOFFSET] = tex.dimen['voffset'],
689 [KEY_PAPERHEIGHT] = tex.dimen['paperheight'],
690 [KEY_ODDSIDEMARGIN] = tex.dimen['oddsidemargin'],
691 [KEY_EVENSIDEMARGIN] = tex.dimen['evensidemargin'],
692 [KEY_TEXTWIDTH] = tex.dimen['textwidth'],
693 [KEY_COLUMNWIDTH] = tex.dimen['columnwidth'],
694 [KEY_COLUMNSEP] = tex.dimen['columnsep'],
695 [KEY_COLUMNCOUNT] = 1,
696 [KEY_TWOSIDE] = false,
697 [KEY_CHECKED] = false,
698 }

```

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

```

699 local function store_page_data(keyvalue_string)
700
701   local page_data = parse_data(keyvalue_string,
702                                 PAGE_DATA_CONVERSION_TABLE,
703                                 PAGE_DATA_DEFAULT_TABLE)
704
705   PAGE_DATA_MAIN_TABLE[page_data[KEY_PAGEDATANO]] = page_data
706
707 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.

```

708 local function store_default_page_data()
709
710   default_page_data = parse_data('',
711                                   PAGE_DATA_CONVERSION_TABLE,
712                                   PAGE_DATA_DEFAULT_TABLE)
713
714   default_page_data[KEY_A BSPAGENO] = 1
715   default_page_data[KEY_CHECKED] = true
716
717   PAGE_DATA_MAIN_TABLE[0] = default_page_data
718
719 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.

```

720 local function check_page_data(keyvalue_string)

```

```

721
722     check_data(keyvalue_string,
723                PAGE_DATA_CONVERSION_TABLE,PAGE_DATA_DEFAULT_TABLE,
724                PAGE_DATA_MAIN_TABLE,KEY_PAGEDATANO,
725                PAGE_CHANGE_REPORT_TABLE)
726
727 end

```

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

## 12.7 Processing of item data from .aux file

Conversion and default tables.

```

728 local ITEM_DATA_CONVERSIONS = {
729     [KEY_ITEMNO] = tonumber,
730     [KEY_ABSPAGENO] = tonumber,
731     [KEY_PAGENO] = tonumber,
732     [KEY_XPOS] = tonumber,
733     [KEY_YPOS] = tonumber,
734     [KEY_HEIGHT] = tonumber,
735     [KEY_DEPTH] = tonumber,
736     [KEY_TYPE] = tonumber,
737     [KEY_POS] = tonumber,
738     [KEY_COLUMN] = tonumber,
739     [KEY_YSHIFT] = tonumber,
740     [KEY_YSEP_ABOVE] = tonumber,
741     [KEY_YSEP_BELOW] = tonumber,
742     [KEY_YSEP_PAGE_TOP] = tonumber,
743     [KEY_YSEP_PAGE_BOTTOM] = tonumber,
744     [KEY_CHECKED] = toboolean,
745 }
746 local ITEM_DATA_DEFAULTS = {
747     [KEY_ITEMNO] = 0,
748     [KEY_ABSPAGENO] = 1,
749     [KEY_PAGENO] = 1,
750     [KEY_XPOS] = 0,
751     [KEY_YPOS] = 0,
752     [KEY_HEIGHT] = 0,
753     [KEY_DEPTH] = 0,
754     [KEY_TYPE] = 0,
755     [KEY_POS] = 0,
756     [KEY_COLUMN] = -1,
757     [KEY_YSHIFT] = 0,
758     [KEY_YSEP_ABOVE] = tex.dimen['marginparpush'],
759     [KEY_YSEP_BELOW] = tex.dimen['marginparpush'],
760     [KEY_YSEP_PAGE_TOP] = tex.dimen['marginparpush'],
761     [KEY_YSEP_PAGE_BOTTOM] = tex.dimen['marginparpush'],
762     [KEY_COLNO_COMPUTED] = 0,
763     [KEY_XSHIFT_COMPUTED] = 0,
764     [KEY_YSHIFT_COMPUTED] = 0,
765     [KEY_SIDE_COMPUTED] = 0,
766     [KEY_MARGINNO_COMPUTED] = 0,
767     [KEY_ENABLED_COMPUTED] = true,
768     [KEY_CHECKED] = false,

```

```
769 }
```

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.

```
770 local function store_item_data(keyvalue_string)
771
772   local item = parse_data(keyvalue_string,
773                           ITEM_DATA_CONVERSIONS,
774                           ITEM_DATA_DEFAULTS)
775
776   ITEM_DATA_MAIN_TABLE[item[KEY_ITEMNO]] = item
777
778 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.

```
779 local function check_item_data(keyvalue_string)
780
781   check_data(keyvalue_string,
782             ITEM_DATA_CONVERSIONS, ITEM_DATA_DEFAULTS,
783             ITEM_DATA_MAIN_TABLE, KEY_ITEMNO,
784             ITEM_CHANGE_REPORT_TABLE)
785
786 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.

```
787 local function write_report(report_table,max_length)
788
789   if #report_table > 0 then
790     local report_text
791     local report_length
792
793     if #report_table <= max_length then
794       report_length = #report_table
795       report_text = ' Here they are:\n'
796     else
797       report_length = max_length
798       report_text = ' Here are the first ' .. report_length .. ':\n'
799     end
800
801     for i=1,report_length do
802       report_text = report_text .. report_table[i]
803       if i < report_length then
804         report_text = report_text .. '\n'
805       end
806     end
807   end
```

```

806     end
807
808     tex.print(report_text)
809 end
810
811 end

```

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

`write_problem_report` Write a report about placement problems to T<sub>E</sub>X in a format suitable for a package warning.

```

812 local function write_problem_report()
813
814     write_report(PROBLEM_REPORT_TABLE, PROBLEM_REPORT_MAX_LENGTH)
815
816 end

```

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

`write_item_change_report` Write a report about changes in item data to T<sub>E</sub>X in a format suitable for a package warning.

```

817 local function write_item_change_report()
818
819     check_removed_data(ITEM_DATA_MAIN_TABLE, ITEM_CHANGE_REPORT_TABLE)
820     write_report(ITEM_CHANGE_REPORT_TABLE, ITEM_CHANGE_REPORT_MAX_LENGTH)
821
822 end

```

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

`write_page_change_report` Write a report about changes in page data to T<sub>E</sub>X in a format suitable for a package warning.

```

823 local function write_page_change_report()
824
825     check_removed_data(PAGE_DATA_MAIN_TABLE, PAGE_CHANGE_REPORT_TABLE)
826     write_report(PAGE_CHANGE_REPORT_TABLE, PAGE_CHANGE_REPORT_MAX_LENGTH)
827
828 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`.

```

829 local RIGHTSIDE_LOOKUP_TABLE = {
830     [true] = {
831         [0] = {
832             [POS_AUTO] = true,
833             [POS_REVERSE] = false,

```

```

834     },
835     [1] = {
836         [POS_AUTO] = false,
837         [POS_REVERSE] = true,
838     },
839     [2] = {
840         [POS_AUTO] = true,
841         [POS_REVERSE] = false,
842     },
843 },
844 [false] = {
845     [0] = {
846         [POS_AUTO] = false,
847         [POS_REVERSE] = true,
848     },
849     [1] = {
850         [POS_AUTO] = true,
851         [POS_REVERSE] = false,
852     },
853     [2] = {
854         [POS_AUTO] = false,
855         [POS_REVERSE] = true,
856     },
857 },
858 }

```

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`).

```

859 local MARGINNO_LOOKUP_TABLE = {
860     [true] = {
861         [0] = {
862             [false] = 1,
863             [true] = 0,
864         },
865         [1] = {
866             [false] = 1,
867             [true] = 5,
868         },
869         [2] = {
870             [false] = 4,
871             [true] = 0,
872         },
873     },
874     [false] = {
875         [0] = {
876             [false] = 2,
877             [true] = 3,
878         },
879         [1] = {
880             [false] = 2,
881             [true] = 5,

```



```

882     },
883     [2] = {
884         [false] = 4,
885         [true] = 3,
886     },
887 },
888 }

```

`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.

```

889 local function compute_items_horizontal(item_data_list,page_data)

```

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

```

890 if #item_data_list == 0 then
891     return
892 end

```

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

```

893 local pageno = item_data_list[1][KEY_PAGENO]
894 local twoside = page_data[KEY_TWOSIDE]
895 local recto = ((pageno % 2) == 1) or (not twoside)
896 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).

```

897 local x_textleft = {}
898 local x_textright = {}
899 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.

```

900 if recto then
901     x_textleft[0] = (
902         page_data[KEY_HOFFSETORIGIN]
903         + page_data[KEY_HOFFSET]
904         + page_data[KEY_ODDSIDEMARGIN]
905     )
906     x_textright[0] = (
907         x_textleft[0]
908         + page_data[KEY_TEXTWIDTH]
909     )
910 else
911     x_textleft[0] = (
912         page_data[KEY_HOFFSETORIGIN]
913         + page_data[KEY_HOFFSET]
914         + page_data[KEY_EVENSIDEMARGIN]
915     )
916     x_textright[0] = (
917         x_textleft[0]
918         + page_data[KEY_TEXTWIDTH]

```

```

919     )
920 end
921 x_textmiddle[0] = (x_textleft[0] + x_textright[0])/2
922
923
924 if columncount == 1 then

```

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

```

925     for i=1,#item_data_list do
926         item_data_list[i][KEY_COLNO_COMPUTED] = 0
927     end
928 else

```

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

```

929     x_textleft[1] = x_textleft[0]
930     x_textright[1] = (
931         x_textleft[1]
932         + page_data[KEY_COLUMNWIDTH]
933     )
934     x_textmiddle[1] = (x_textleft[1] + x_textright[1])/2
935
936     x_textleft[2] = (
937         x_textright[1]
938         + page_data[KEY_COLUMNSEP]
939     )
940     x_textright[2] = (
941         x_textleft[2]
942         + page_data[KEY_COLUMNWIDTH]
943     )
944     x_textmiddle[2] = (x_textleft[2] + x_textright[2])/2
945

```

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

```

946     local left_column_x_limit = (
947         x_textright[1]
948         + .5*page_data[KEY_COLUMNSEP]
949     )

```

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

```

950     for i=1,#item_data_list do
951         local item_data = item_data_list[i]
952
953         if item_data[KEY_COLUMN] >= 0 then
954             item_data[KEY_COLNO_COMPUTED] = item_data[KEY_COLUMN]
955         else
956             if item_data[KEY_XPOS] <= left_column_x_limit then
957                 item_data[KEY_COLNO_COMPUTED] = 1
958             else
959                 item_data[KEY_COLNO_COMPUTED] = 2
960             end
961         end
962     end

```

```

963
964     end

```

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

```

965     for i=1,#item_data_list do
966         local item = item_data_list[i]
967
968         local pos = item[KEY_POS]
969         local colnocomputed = item[KEY_COLNO_COMPUTED]
970
971         if pos == POS_LEFT then
972             rightside = false
973         elseif pos == POS_RIGHT then
974             rightside = true
975         elseif pos == POS_NEAREST then
976             rightside = (item[KEY_XPOS] >= x_textmiddle[colnocomputed])
977         else

```

pos must be POS\_AUTO or POS\_REVERSE

```

978             rightside = RIGHTSIDE_LOOKUP_TABLE[recto][colnocomputed][pos]
979         end
980
981         local marginno = MARGINNO_LOOKUP_TABLE[recto][colnocomputed][rightside]
982
983         if rightside then
984             item[KEY_SIDE_COMPUTED] = 0
985             item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
986                                     + x_textright[colnocomputed]
987         else
988             item[KEY_SIDE_COMPUTED] = 1
989             item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
990                                     + x_textleft[colnocomputed]
991         end
992         item[KEY_MARGINNO_COMPUTED] = marginno
993
994     end
995
996 end

```

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

get\_y\_item\_top Return the y-coordinate of the top of the item described by item\_data.

```

997 local function get_y_item_top(item_data)
998     return item_data[KEY_YPOS]
999         + item_data[KEY_YSHIFT_COMPUTED]
1000        + item_data[KEY_HEIGHT]
1001 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.

```

1002 local function get_y_item_bottom(item_data)
1003     return item_data[KEY_YPOS]
1004        - item_data[KEY_DEPTH]

```

```

1005         + item_data[KEY_YSHIFT_COMPUTED]
1006     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).

```

1007 local function get_ysep_list(item_data_list)
1008
1009     local ysep_list = {}
1010
1011     ysep_list[0] = 0
1012     for i=1,#item_data_list-1 do
1013         ysep_list[i] = math.max(
1014             item_data_list[i][KEY_YSEP_BELOW],
1015             item_data_list[i+1][KEY_YSEP_ABOVE]
1016         )
1017     end
1018     ysep_list[#item_data_list] = 0
1019
1020     return ysep_list
1021
1022 end

```

*(End of definition for get\_ysep\_list.)*

## 12.10 Computing vertical positions

### 12.10.1 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.

```

1023 local function compute_items_vertical_optfixed_enabled(item_data_list)
1024
1025     local optfixed_item_data_list = {}
1026     local fixed_item_data_list = {}
1027
1028     for _,item_data in pairs(item_data_list) do
1029         if item_data[KEY_TYPE] == TYPE_OPTFIXED then
1030             optfixed_item_data_list[#optfixed_item_data_list+1] = item_data
1031         elseif item_data[KEY_TYPE] == TYPE_FIXED then

```

```

1032     fixed_item_data_list[#fixed_item_data_list+1] = item_data
1033   end
1034 end
1035
1036 for _,optfixed_item_data in pairs(optfixed_item_data_list) do
1037   local optfixed_y_item_top = get_y_item_top(optfixed_item_data)
1038   local optfixed_y_item_bottom = get_y_item_bottom(optfixed_item_data)
1039
1040   for _,fixed_item_data in pairs(fixed_item_data_list) do
1041     local fixed_y_item_top = get_y_item_top(fixed_item_data)
1042     local fixed_y_item_bottom = get_y_item_bottom(fixed_item_data)
1043
1044     if (
1045       (
1046         (fixed_y_item_bottom - optfixed_y_item_top)
1047         <
1048         math.max(
1049           fixed_item_data[KEY_YSEP_BELOW],
1050           optfixed_item_data[KEY_YSEP_ABOVE]
1051         )
1052       )
1053       and
1054       (
1055         (optfixed_y_item_bottom - fixed_y_item_top)
1056         <
1057         math.max(
1058           optfixed_item_data[KEY_YSEP_BELOW],
1059           fixed_item_data[KEY_YSEP_ABOVE]
1060         )
1061       )
1062     ) then
1063       optfixed_item_data[KEY_ENABLED_COMPUTED] = false
1064       break
1065     end
1066   end
1067 end
1068
1069 end

```

*(End of definition for compute\_items\_vertical\_optfixed\_enabled.)*

## 12.10.2 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.

```

1070 local function compute_items_vertical_adjustment(item_data_list,page_data)

```

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

```

1071   if #item_data_list == 0 then
1072     return
1073   end

```

```

1074
1075 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.

```

1076 local y_limit_above = (
1077     page_data[KEY_VOFFSET]
1078     + page_data[KEY_PAPERHEIGHT]
1079     - item_data_list[1][KEY_YSEP_PAGE_TOP]
1080 )
1081
1082 for i=1,#item_data_list do
1083     local item_data = item_data_list[i]
1084
1085     local y_item_top = get_y_item_top(item_data)
1086
1087     if y_item_top > y_limit_above then
1088         if item_data[KEY_TYPE] == TYPE_NORMAL then
1089             item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1090                                             + (y_limit_above - y_item_top)
1091         end
1092     end
1093
1094     y_limit_above = get_y_item_bottom(item_data) - ysep_list[i]
1095 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.

```

1096 local y_limit_below = (
1097     page_data[KEY_VOFFSET]
1098     + item_data_list[#item_data_list][KEY_YSEP_PAGE_BOTTOM]
1099 )
1100
1101 for i=#item_data_list,1,-1 do
1102     local item_data = item_data_list[i]
1103
1104     local y_item_bottom = get_y_item_bottom(item_data)
1105
1106     if y_item_bottom < y_limit_below then
1107         if item_data[KEY_TYPE] == TYPE_NORMAL then
1108             item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1109                                             + (y_limit_below - y_item_bottom)
1110         end
1111     end
1112
1113     y_limit_below = get_y_item_top(item_data) + ysep_list[i-1]
1114 end
1115
1116 end

```

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

### 12.10.3 Checking vertical adjustment

Messages to use when checking results of vertical adjustment.

```

1117 local ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES = {
1118     [TYPE_NORMAL] = 'Moveable item > ysep page top',
1119     [TYPE_FIXED] = 'Topmost fixed item > ysep page top',
1120     [TYPE_OPTFIXED] = 'Topmost optfixed item > ysep page top',
1121 }
1122 local ITEM_CLASH_MESSAGES = {
1123     [TYPE_NORMAL] = {
1124         [TYPE_NORMAL] = 'moveable items'
1125         .. ' (this shouldn\'t happen)',
1126         [TYPE_FIXED] = 'moveable item above fixed item',
1127         [TYPE_OPTFIXED] = 'moveable item above optfixed item',
1128     },
1129     [TYPE_FIXED] = {
1130         [TYPE_NORMAL] = 'moveable item below fixed item',
1131         [TYPE_FIXED] = 'fixed items',
1132         [TYPE_OPTFIXED] = 'fixed item above optfixed item '
1133         .. ' (this shouldn\'t happen)',
1134     },
1135     [TYPE_OPTFIXED] = {
1136         [TYPE_NORMAL] = 'moveable items below optfixed item',
1137         [TYPE_FIXED] = 'fixed item below optfixed item '
1138         .. ' (this shouldn\'t happen)',
1139         [TYPE_OPTFIXED] = 'optfixed items '
1140         .. ' (this shouldn\'t happen)',
1141     },
1142 }
1143 local ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE = {
1144     [TYPE_NORMAL] = 'Moveable item < ysep page bottom',
1145     [TYPE_FIXED] = 'Bottommost fixed item < ysep page bottom',
1146     [TYPE_OPTFIXED] = 'Bottommost optfixed item < ysep page bottom',
1147 }

```

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.

```

1148 local function check_items_vertical(item_data_list,page_data)

```

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

```

1149     if (#item_data_list) == 0 then
1150         return
1151     end
1152
1153     local ysep_list = get_ysep_list(item_data_list)
1154
1155     local item_data
1156

```

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

```

1157     item_data = item_data_list[1]
1158     if (
1159         get_y_item_top(item_data) > page_data[KEY_VOFFSET]
1160         + page_data[KEY_PAPERHEIGHT]
1161         - item_data[KEY_YSEP_PAGE_TOP]
1162     ) then
1163         table.insert(

```

```

1164     PROBLEM_REPORT_TABLE,
1165     get_data_page_number(item_data)
1166     .. ' ' .. ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES[item_data[KEY_TYPE]]
1167 )
1168 end
1169
1170 for i=2,#item_data_list do
1171     local item_data = item_data_list[i]
1172     local prev_item_data = item_data_list[i-1]
1173     if (
1174         get_y_item_top(item_data) > get_y_item_bottom(prev_item_data)
1175                                     - ysep_list[i-1]
1176     ) then
1177         table.insert(
1178             PROBLEM_REPORT_TABLE,
1179             get_data_page_number(item_data)
1180             .. ' Clash: ' ..
1181             ITEM_CLASH_MESSAGES[prev_item_data[KEY_TYPE]][item_data[KEY_TYPE]]
1182         )
1183     end
1184 end

```

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

```

1185     item_data = item_data_list[#item_data_list]
1186     if (
1187         get_y_item_bottom(item_data) < page_data[KEY_VOFFSET]
1188                                     + item_data[KEY_YSEP_PAGE_BOTTOM]
1189     ) then
1190         table.insert(
1191             PROBLEM_REPORT_TABLE,
1192             get_data_page_number(item_data)
1193             .. ' ' .. ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE[item_data[KEY_TYPE]]
1194         )
1195     end
1196
1197 end

```

*(End of definition for check\_items\_vertical.)*

#### 12.10.4 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`.

```

1198 local function compute_items_vertical(item_data_list,page_data)

```

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

```

1199     for i=1,#item_data_list do
1200         local item_data = item_data_list[i]
1201
1202         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT]
1203     end

```



Decide which items of type ITEM\_DATA\_OPTFIXED are to be disabled.

```
1204 compute_items_vertical_optfixed_enabled(item_data_list)
```

Strip any item\_data with KEY\_ENABLED\_COMPUTED set to false from item\_data\_list.

```
1205 list_filter(item_data_list,function(item_data)
1206     return item_data[KEY_ENABLED_COMPUTED]
1207 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.

```
1208 table.sort(
1209     item_data_list,
1210     function(left,right)
1211         local y_diff = left[KEY_YPOS] - right[KEY_YPOS]
1212
1213         if y_diff > 0 then
1214             return true
1215         elseif y_diff < 0 then
1216             return false
1217         end
1218
1219         local x_diff = left[KEY_XPOS] - right[KEY_XPOS]
1220
1221         if x_diff < 0 then
1222             return true
1223         elseif x_diff > 0 then
1224             return false
1225         end
1226
1227         return (left[KEY_ITEMNO] < right[KEY_ITEMNO])
1228     end
1229 )
1230
1231 compute_items_vertical_adjustment(item_data_list,page_data)
1232
1233 check_items_vertical(item_data_list,page_data)
1234
1235 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.

```
1236 local function compute_items()
```

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

```
1237     local max_abspageno = 0
1238
1239     for k,v in pairs(ITEM_DATA_MAIN_TABLE) do
1240         max_abspageno = math.max(v[KEY_ABSPAGENO],max_abspageno)
1241     end
```

list `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.

```
1242 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`.

```
1243 for i=1,max_abspageno do
1244     per_abspage_item_data_list[i] = {}
1245 end
1246 for _,item_data in pairs(ITEM_DATA_MAIN_TABLE) do
1247     local temp_table = per_abspage_item_data_list[item_data[KEY_ABSPAGENO]]
1248     temp_table[#temp_table+1] = item_data
1249 end
```

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

```
1250 local per_abspage_page_data_list = {}
1251 local pagedatano = 0
1252 for abspageno = 1,max_abspageno do
1253     while (
1254         PAGE_DATA_MAIN_TABLE[pagedatano+1] ~= nil
1255         and
1256         PAGE_DATA_MAIN_TABLE[pagedatano+1][KEY_ABSPAGENO] == abspageno
1257     ) do
1258         pagedatano = pagedatano+1
1259     end
1260     per_abspage_page_data_list[abspageno] = PAGE_DATA_MAIN_TABLE[pagedatano]
1261 end
```

Iterate through all pages and perform the necessary computations.

```
1262 for abspageno=1,#per_abspage_item_data_list do
1263     local current_page_data = per_abspage_page_data_list[abspageno]
1264     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.

```
1265     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.

```
1266     local current_page_item_data_sublists = {}
1267
1268     for i=0,5 do
1269         current_page_item_data_sublists[i] = {}
1270     end
1271
1272     for _,item_data in pairs(current_page_item_data_list) do
1273         table.insert(
```

```

1274         current_page_item_data_sublists[item_data[KEY_MARGINNO_COMPUTED]],
1275         item_data
1276     )
1277 end

```

Compute vertical positons for each sublist.

```

1278     for i=0,5 do
1279         compute_items_vertical(
1280             current_page_item_data_sublists[i],
1281             current_page_data
1282         )
1283     end
1284 end
1285 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`.

```

1286 local function load_item_data(itemno)
1287
1288     item = ITEM_DATA_MAIN_TABLE[tonumber(itemno)]
1289     if item == nil then
1290         item = ITEM_DATA_DEFAULTS
1291     end
1292
1293     tex.count['l__marginalia_page_int'] = item[KEY_PAGENO]
1294     tex.count['l__marginalia_column_computed_int'] = item[KEY_COLNO_COMPUTED]
1295     tex.dimen['l__marginalia_xshift_computed_dim'] = item[KEY_XSHIFT_COMPUTED]
1296     tex.dimen['l__marginalia_yshift_computed_dim'] = item[KEY_YSHIFT_COMPUTED]
1297     tex.count['l__marginalia_side_computed_int'] = item[KEY_SIDE_COMPUTED]
1298     tex.count['l__marginalia_marginno_computed_int']
1299         = item[KEY_MARGINNO_COMPUTED]
1300     if item[KEY_ENABLED_COMPUTED] then
1301         tex.count['l__marginalia_enabled_computed_int'] = 1
1302     else
1303         tex.count['l__marginalia_enabled_computed_int'] = 0
1304     end
1305
1306 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`.

```

1307 return {
1308     store_default_page_data = store_default_page_data,
1309     store_page_data = store_page_data,
1310     check_page_data = check_page_data,
1311

```

```
1312 store_item_data = store_item_data,  
1313 check_item_data = check_item_data,  
1314  
1315 compute_items = compute_items,  
1316  
1317 load_item_data = load_item_data,  
1318  
1319 write_problem_report = write_problem_report,  
1320  
1321 write_page_change_report = write_page_change_report,  
1322 write_item_change_report = write_item_change_report,  
1323 }  
1324 </lua>
```