

marginalia — Non-floating marginal content with automatic placement for Lua^AT_EX^{*}

Alan J. Cain

Released 2025-02-17

Abstract

This Lua^AT_EX 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

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

^{*}This file describes v0.80.1, last revised 2025-02-17.

11	Implementation (\LaTeX package)	14
11.1	Initial set-up	14
11.2	Options	14
11.2.1	Type	14
11.2.2	Horizontal placement	14
11.2.3	Vertical placement	16
11.2.4	Appearance	17
11.3	Lua backend and interface	18
11.4	Processing data from the <code>.aux</code> file	19
11.5	Writing page data to the <code>.aux</code> file	21
11.6	Marginal content item processing	22
11.6.1	Variables	22
	Variables set by \LaTeX	22
	Variables set by Lua.	22
11.6.2	Core macro	23
11.6.3	Width and style selection	26
11.6.4	Auxiliary placement macros	27
11.7	User commands	28
12	Implementation (Lua backend)	28
12.1	Global variables	28
12.2	Constants	29
12.3	Keys for tables	29
12.3.1	Keys for both page and item data tables	29
12.3.2	Keys for page data tables, layout etc.	29
12.3.3	Keys for item data tables	30
12.4	Utility functions	31
12.5	Generic page/item data functions	31
12.6	Processing of page data from <code>.aux</code> file	33
12.7	Processing of item data from <code>.aux</code> file	35
12.8	Writing reports	36
12.9	Computing horizontal positions	37
12.10	Computing vertical positions	42
12.10.1	Computing <code>optfixed</code> enabled	42
12.10.2	Computing vertical adjustment	43
12.10.3	Checking vertical adjustment	44
12.10.4	Core vertical position computation	46
12.11	Passing item_data back to \LaTeX	49
12.12	Export public functions	49

Index	51
--------------	-----------

1 Introduction

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

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

Packages like `mparhack`¹ (Tom Sgouros & Stefan Ulrich), `marginnote`² (Markus Kohm), `marginfix`³ (Stephen Hicks) and `marginfit`⁴ (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 \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 \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.

Caveat: `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].⁵ Thus the basic functionality has been tested extensively, and it has performed correctly.

Licence. `marginalia` is released under the \LaTeX Project Public Licence v1.3c or later.¹

2 Requirements

`marginalia` requires (1) \LaTeX , (2) a recent \LaTeX kernel with `expl3` support (any kernel version since 2020-02-02 should suffice). It does not depend on any other packages.

¹URL: <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

Getting started

```
\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}
```

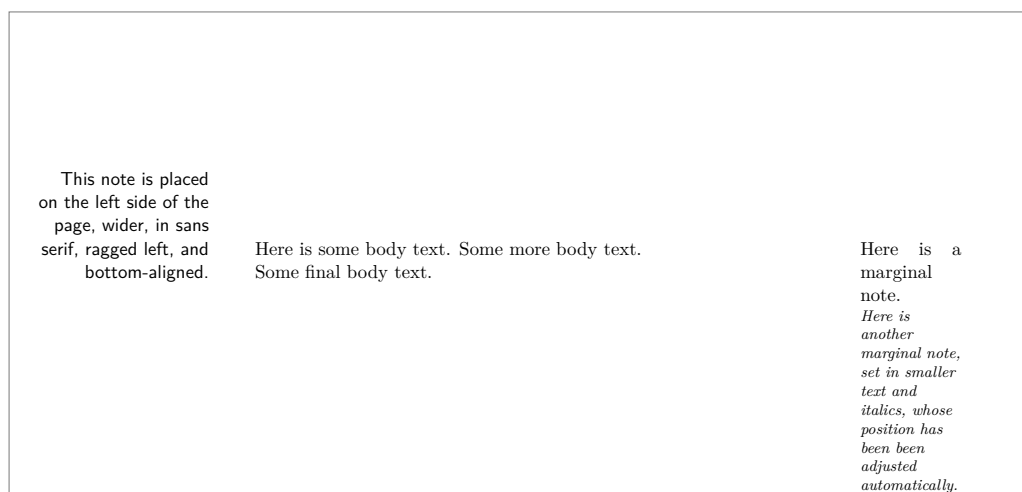


Figure 4.1: A small demonstration of marginalia.

3 Installation

To install `marginalia` manually, run `luatex marginalia.ins` and copy `marginalia.sty` and `marginalia.lua` to somewhere `LuaATEX` 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 `LuaATEX` for the marginal content to be placed correctly.* (See [Section 8](#) for discussion of the need for multiple runs.)

Turn to [Section 5](#) for a detailed description of the available user commands, and [Section 6](#) for the various options (such as `style=<code>`) than can be used to change the

User commands placement and formatting of the marginal content.

5 User commands

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

This is the basic command for placing marginal content. The *<content>* can, roughly speaking, be anything: text, mathematics, included graphics, TikZ. The optional argument *<options>* is a key-value list that specifies how the content is typeset. The keys are described in [Subsection 6](#).

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

This command is used to set options globally. The argument *<options>* is the same kind of key-value list as the *<options>* argument for the `\marginalia` command, and are described in [Subsection 6](#).

<code>\marginalianewgeometry</code>	<code>\marginalianewgeometry</code>
-------------------------------------	-------------------------------------

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

This command signals to `marginalia` that the page layout has been changed, for instance by using the `\newgeometry` command from the `geometry` package,⁶ or by using the \LaTeX command `\twocolumn` to switch to two-column mode. It should be issued immediately after such a change, and certainly before the first page with the new layout has been shipped out. There is no harm in using it unnecessarily.

5.1 Access to page and column

Two counters available within the *<content>* of `\marginalia` 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.

<code>\marginaliapage</code>

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 \LaTeX run and will default to 1.

<code>\marginaliacolumn</code>

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

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

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 ^A T _E X code	[Empty]
<code>style recto outer</code>	L ^A T _E X code	[Empty]
<code>style recto inner</code>	L ^A T _E X code	[Empty]
<code>style verso outer</code>	L ^A T _E X code	[Empty]
<code>style verso inner</code>	L ^A T _E X code	[Empty]
<code>style right between</code>	L ^A T _E X code	[Empty]
<code>style left between</code>	L ^A T _E X code	[Empty]

Options

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**)

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

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

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

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

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

xsep verso inner **xsep right between:** used for an item set from the right column between the columns.

xsep right between **xsep left between:** used for an item set from the left column between the columns.

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

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

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

xsep: 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**)

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: 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).

Options

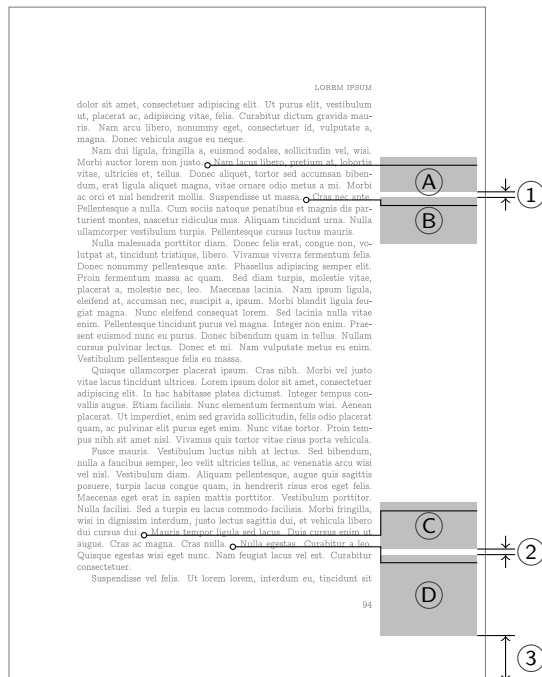


Figure 6.1: (Illustration of `ysep`) The length ① is at least the value of `ysep below` specified (locally or globally) for marginal content item ① and at least the value of `ysep above` specified for item ②. In this example diagram, ② has been automatically moved down from its natural position to maintain 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.

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

Placement

width left between: used for an item set from the right column and placed between the columns.

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

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

width between: a shorthand for setting the keys `width right between` and `width left between` simultaneously to the same value.

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

(The shorthands `width outer` and `width inner` exist because page geometry is usually symmetrical between recto and verso pages as regards outer and inner margins. The shorthand `width between` exists because the space between columns, if used at all for marginal content, will often be shared equally.) Each of these keys must be set to a valid dimension. (*Default:* value of `\marginparwidth` when the package is loaded)

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

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

style recto inner

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

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

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

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

style right between: used for an item set from the right column between the columns.

style left between: used for an item set from the right column between the columns.

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

Each of these keys should be set to L^AT_EX code that specifies the style. (*Default:* [Empty])

7 Placement

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

7.1 Horizontal placement

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

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

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

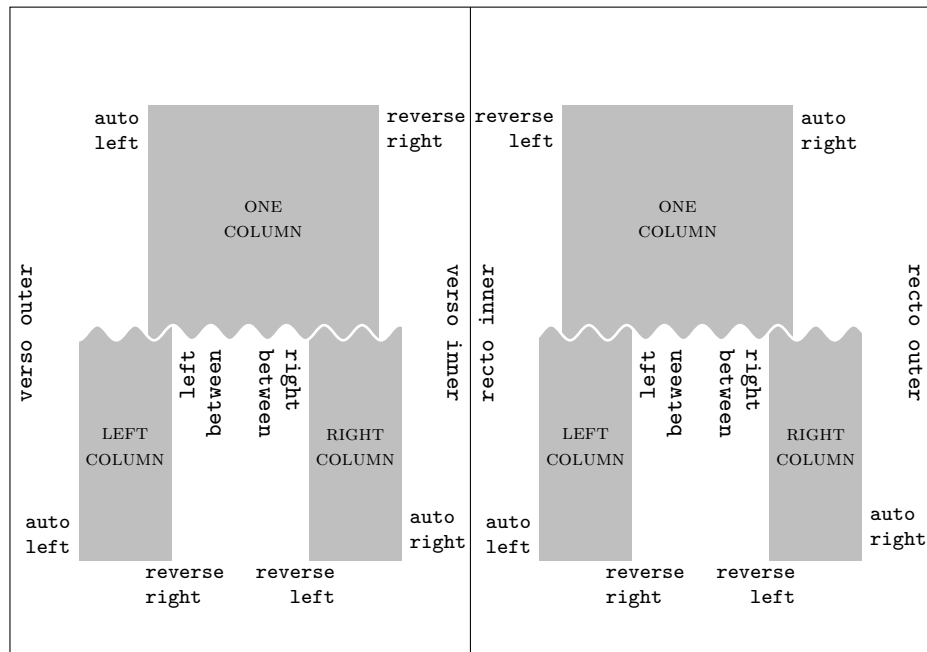


Figure 7.1: Summary of the positioning of marginal content using `pos`, and terminology used in `width` and `style` keys, on recto and verso pages, in both one-column and two-column mode.

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

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

Usage notes

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

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

8 Usage notes

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

More runs may be needed if the position of the call to `\marginalia` changes between runs. Provided the main text stabilizes, the placement of items using `\marginalia` should be correct two runs later.

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

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

9 Incompatibilities

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

10 Limitations

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

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

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

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

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

References

- [Bri04] R. Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, version 3.0, 2004.
- [Cai24] A. J. Cain. *Form & Number: A History of Mathematical Beauty*. Lisbon, 2024.
URL: https://archive.org/details/cain_formandnumber_ebook_large.

11 Implementation (L^AT_EX package)

```

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

```

11.1 Initial set-up

Package identification/version information.

```

3 \NeedsTeXFormat{LaTeX2e}[2020-02-02]
4 \ProvidesExplPackage{marginalia}{2025-02-17}{0.80.1}
5 {Non-floating marginal content for LuaLaTeX}

```

Check that Lua_TE_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 Options

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

11.2.1 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.

```

12 \int_new:N\l__marginalia_type_int
13 \keys_define:nn { marginalia }
14 {
15   type .choices:nn = {normal,fixed,optfixed}{
16     \int_set:Nn\l__marginalia_type_int{\l_keys_choice_int}
17   },
18   type .initial:n = normal,
19 }

```

(End of definition for \l__marginalia_type_int.)

11.2.2 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**.

```

20 \int_new:N\l__marginalia_pos_int
21 \keys_define:nn { marginalia }
22 {
23   pos .choices:nn = {auto,reverse,left,right,nearest}{
24     \int_set:Nn\l__marginalia_pos_int{\l_keys_choice_int}
25   },
26   pos .initial:n = auto
27 }

```

(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).

```

28 \int_new:N\l__marginalia_column_int
29 \keys_define:nn { marginalia }
30 {
31   column .choices:nn = {auto,one,left,right}{
32     \int_set:Nn\l__marginalia_column_int{\l_keys_choice_int-2}
33   },
34   column .initial:n = auto,
35 }

```

(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.

```

36 \keys_define:nn { marginalia }
37 {
38   xsep~recto~outer .dim_set:N = \l__marginalia_xsep_recto_outer_dim,
39   xsep~recto~inner .dim_set:N = \l__marginalia_xsep_recto_inner_dim,
40   xsep~verso~outer .dim_set:N = \l__marginalia_xsep_verso_outer_dim,
41   xsep~verso~inner .dim_set:N = \l__marginalia_xsep_verso_inner_dim,
42   xsep~right~between .dim_set:N = \l__marginalia_xsep_right_between_dim,
43   xsep~left~between .dim_set:N = \l__marginalia_xsep_left_between_dim,
44   xsep .code:n = {
45     \keys_set:nn{ marginalia }{
46       xsep~recto~outer=#1,
47       xsep~recto~inner=#1,
48       xsep~verso~outer=#1,
49       xsep~verso~inner=#1,
50       xsep~right~between=#1,
51       xsep~left~between=#1,
52     }
53   },
54   xsep~outer .code:n = {
55     \keys_set:nn{ marginalia }{
56       xsep~recto~outer=#1,
57       xsep~verso~outer=#1,
58     }
59   },
60   xsep~inner .code:n = {
61     \keys_set:nn{ marginalia }{
62       xsep~recto~inner=#1,
63       xsep~verso~inner=#1,
64     }
65   },
66   xsep~between .code:n = {
67     \keys_set:nn{ marginalia }{
68       xsep~right~between=#1,
69       xsep~left~between=#1,
70     }

```

```

71   },
72   xsep .initial:n = \marginparsep,
73 }

```

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

11.2.3 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).

```

74 \int_new:N\l__marginalia_valign_int
75 \keys_define:nn { marginalia }
76 {
77   valign .choices:nn = {t,b}{
78     \int_set_eq:NN\l__marginalia_valign_int\l_keys_choice_int
79   },
80   valign .initial:n = t,
81 }

```

(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.

```

82 \keys_define:nn { marginalia }
83 {
84   yshift .dim_set:N = \l__marginalia_default_yshift_dim,
85   yshift .initial:n = 0pt,
86 }

```

(End of definition for `\l__marginalia_default_yshift_dim`.)

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

```

87 \keys_define:nn { marginalia }
88 {
89   ysep~above .dim_set:N = \l__marginalia_ysep_above_dim,
90   ysep~below .dim_set:N = \l__marginalia_ysep_below_dim,
91   ysep~page~top .dim_set:N = \l__marginalia_ysep_page_top_dim,
92   ysep~page~bottom .dim_set:N = \l__marginalia_ysep_page_bottom_dim,
93   ysep .code:n = {
94     \keys_set:nn{ marginalia }{
95       ysep~below=#1,
96       ysep~above=#1,
97       ysep~page~top=#1,
98       ysep~page~bottom=#1,
99     }
100   },
101   ysep .initial:n = \marginparpush,
102 }

```

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

11.2.4 Appearance

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

```

\l_marginalia_width_recto_outer_dim
\l_marginalia_width_recto_inner_dim
\l_marginalia_width_verso_outer_dim
\l_marginalia_width_verso_inner_dim
\l_marginalia_width_right_between_dim
\l_marginalia_width_left_between_dim
103 \keys_define:nn { marginalia }
104 {
105   width~recto~outer .dim_set:N = \l_marginalia_width_recto_outer_dim,
106   width~recto~inner .dim_set:N = \l_marginalia_width_recto_inner_dim,
107   width~verso~outer .dim_set:N = \l_marginalia_width_verso_outer_dim,
108   width~verso~inner .dim_set:N = \l_marginalia_width_verso_inner_dim,
109   width~right~between .dim_set:N = \l_marginalia_width_right_between_dim,
110   width~left~between .dim_set:N = \l_marginalia_width_left_between_dim,
111   width .code:n = {
112     \keys_set:nn{ marginalia }{
113       width~recto~outer=#1,
114       width~recto~inner=#1,
115       width~verso~outer=#1,
116       width~verso~inner=#1,
117       width~right~between=#1,
118       width~left~between=#1,
119     }
120   },
121   width~outer .code:n = {
122     \keys_set:nn{ marginalia }{
123       width~recto~outer=#1,
124       width~verso~outer=#1,
125     }
126   },
127   width~inner .code:n = {
128     \keys_set:nn{ marginalia }{
129       width~recto~inner=#1,
130       width~verso~inner=#1,
131     }
132   },
133   width~between .code:n = {
134     \keys_set:nn{ marginalia }{
135       width~right~between=#1,
136       width~left~between=#1,
137     }
138   },
139   width .initial:n = \marginparwidth,
140 }

```

(End of definition for \l_marginalia_width_recto_outer_dim and others.)

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.

```

\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
141 \keys_define:nn { marginalia }
142 {
143   style~recto~outer .tl_set:N = \l_marginalia_style_recto_outer_tl,
144   style~recto~inner .tl_set:N = \l_marginalia_style_recto_inner_tl,
145   style~verso~outer .tl_set:N = \l_marginalia_style_verso_outer_tl,
146   style~verso~inner .tl_set:N = \l_marginalia_style_verso_inner_tl,
147   style~right~between .tl_set:N = \l_marginalia_style_right_between_tl,

```

```

148 style~left~between .tl_set:N = \l__marginalia_style_left_between_tl,
149 style .code:n = {
150   \keys_set:nn{ marginalia }{
151     style~recto~outer=#1,
152     style~recto~inner=#1,
153     style~verso~outer=#1,
154     style~verso~inner=#1,
155     style~right~between=#1,
156     style~left~between=#1,
157   }
158 },
159 style .initial:n = {},
160 }

```

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

11.3 Lua backend and interface

Load the Lua backend.

```

161 \lua_now:n{
162   marginalia = require('marginalia')
163 }

```

The following 9 macros interface between L^AT_EX and Lua code. Each control sequence `__marginalia_lua_XYZ` simply calls the corresponding Lua function `marginalia.XYZ`.

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

```

164 \cs_new:Npn\__marginalia_lua_store_default_page_data:
165 {
166   \lua_now:n{ marginalia.store_default_page_data() }
167 }
168 \cs_new:Npn\__marginalia_lua_store_page_data:n #1
169 {
170   \lua_now:n{ marginalia.store_page_data('#1') }
171 }
172 \cs_new:Npn\__marginalia_lua_check_page_data:n #1
173 {
174   \lua_now:n{ marginalia.check_page_data('#1') }
175 }
176 \cs_new:Npn\__marginalia_lua_write_page_change_report:
177 {
178   \lua_now:n{ marginalia.write_page_change_report() }
179 }
180 \cs_new:Npn\__marginalia_lua_store_item_data:n #1
181 {
182   \lua_now:n{ marginalia.store_item_data('#1') }
183 }
184 \cs_new:Npn\__marginalia_lua_check_item_data:n #1
185 {
186   \lua_now:n{ marginalia.check_item_data('#1') }
187 }
188 \cs_new:Npn\__marginalia_lua_compute_items:

```

```

189 {
190   \lua_now:n{ marginalia.compute_items() }
191 }
192 \cs_new:Npn\__marginalia_lua_write_problem_report:
193 {
194   \lua_now:n{ marginalia.write_problem_report() }
195 }
196 \cs_new:Npn\__marginalia_lua_write_item_change_report:
197 {
198   \lua_now:n{ marginalia.write_item_change_report() }
199 }

```

(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.

```

200 \cs_new:Npn\__marginalia_lua_load_item_data:n #1
201 {
202   \lua_now:e{ marginalia.load_item_data('#1') }
203 }

```

(End of definition for __marginalia_lua_load_item_data:n.)

11.4 Processing data from the .aux file

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

```

204 \NewDocumentCommand{\marginalia@pagedata}{ m }{
205   \__marginalia_process_page_data:n{#1}
206 }

```

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.

```

207 \cs_set_eq:NN
208   \__marginalia_process_page_data:n
209   \__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.

```

210 \DeclareDocumentCommand{\marginalia@itemdata}{ m }{
211   \__marginalia_process_item_data:n{#1}
212 }

```

(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.

```

213 \cs_set_eq:NN
214   \__marginalia_process_item_data:n
215   \__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.

```

216 \AddToHook{begindocument}{
217   \__marginalia_lua_store_default_page_data:
218   \__marginalia_lua_compute_items:
219   \cs_set_eq:NN\l__marginalia_aux_iow\@mainaux
220 }

```

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.

```

221 \AddToHook{enddocument/afterlastpage}{
222   \cs_set_eq:NN
223     \__marginalia_process_page_data:n
224     \__marginalia_lua_check_page_data:n
225   \cs_set_eq:NN
226     \__marginalia_process_item_data:n
227     \__marginalia_lua_check_item_data:n
228 }

```

`__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:

```

229 \msg_new:nnn{marginalia}{placement_problem}
230 { Problems~in~placement.~#1 }
231 \msg_new:nnn{marginalia}{item_change}
232 { Changes~in~item~data.~#1 }
233 \msg_new:nnn{marginalia}{page_change}
234 { Changes~in~page~data.~#1 }
235 \cs_new:Npn\__marginalia_write_reports:
236 {
237   \group_begin:
238   \tl_set:N\l_tmpa_tl{\__marginalia_lua_write_problem_report:}
239   \tl_if_blank:VF\l_tmpa_tl
240   {
241     \msg_warning:nne{marginalia}{placement_problem}{\tl_use:N\l_tmpa_tl}
242   }
243   \tl_set:N\l_tmpa_tl{\__marginalia_lua_write_item_change_report:}
244   \tl_if_blank:VF\l_tmpa_tl
245   {
246     \msg_warning:nne{marginalia}{item_change}{\tl_use:N\l_tmpa_tl}
247   }
248   \tl_set:N\l_tmpa_tl{\__marginalia_lua_write_page_change_report:}
249   \tl_if_blank:VF\l_tmpa_tl
250   {
251     \msg_warning:nne{marginalia}{page_change}{\tl_use:N\l_tmpa_tl}
252   }
253   \group_end:
254 }

```

(End of definition for `__marginalia_write_reports:`.)

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

```
255 \AddToHook{enddocument/info}{
256   \__marginalia_write_reports:
257 }
```

11.5 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.

```
258 \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`).

```
259 \cs_set_eq:NN\__marginalia_write_page_data:\prg_do_nothing:
260 \cs_new:Npn\__marginalia_write_page_data_real:
261 {
262   \int_gincr:N\g_marginalia_pagedatano_int
263   \iow_now:Ne\l__marginalia_aux_iow{
264     \token_to_str:N\marginalia@pagedata{
265       pagedatano=\int_value:w\g_marginalia_pagedatano_int,
266       abspageno=\int_eval:n{\g_shipout_readonly_int+1},
267       hoffset=\int_value:w\hoffset,
268       voffset=\int_value:w\voffset,
269       paperheight=\int_value:w\paperheight,
270       oddsidemargin=\int_value:w\oddsidemargin,
271       evensidemargin=\int_value:w\evensidemargin,
272       textwidth=\int_value:w\textwidth,
273       columncount=\int_value:w\col@number,
274       columnwidth=\int_value:w\columnwidth,
275       columnsep=\int_value:w\columnsep,
276       twoside=\bool_to_str:n{\legacy_if_p:n{@twoside}},
277     }
278   }
279 }
```

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.

```
280 \AddToHook{begindocument/end}
281 {
282   \cs_set_eq:NN
283     \__marginalia_write_page_data:
284     \__marginalia_write_page_data_real:
285   \__marginalia_write_page_data:
286 }
```

(End of definition for `__marginalia_write_page_data:`.)

11.6 Marginal content item processing

11.6.1 Variables

Variables set by **L^AT_EX**.

`\g__marginalia_itemno_int` Global integer variable to index marginal content items.

```
287 \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.

```
288 \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 289 \dim_new:N\l__marginalia_item_height_dim
```

```
290 \dim_new:N\l__marginalia_item_depth_dim
```

(End of definition for \l__marginalia_item_height_dim and \l__marginalia_item_depth_dim.)

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

`\l__marginalia_page_int` Integer variable for the page on which the marginal content item appears. This variable will be made available via `\marginaliapage` within the `\content` of `\marginalia`.

```
291 \int_new:N\l__marginalia_page_int
```

(End of definition for \l__marginalia_page_int.)

`\l__marginalia_column_computed_int` Integer variable for the column next to which the marginal content item appears. This variable will be made available via `\marginaliacolumn` within the `\content` of `\marginalia`.

```
292 \int_new:N\l__marginalia_column_computed_int
```

(End of definition for \l__marginalia_column_computed_int.)

`\l__marginalia_xshift_computed_dim` Dimension variables to hold the differences in x and y coordinates between the call to `\marginalia` and the position where the marginal content item should appear.

`\l__marginalia_yshift_computed_dim`

```
293 \dim_new:N\l__marginalia_xshift_computed_dim
```

```
294 \dim_new:N\l__marginalia_yshift_computed_dim
```

(End of definition for \l__marginalia_xshift_computed_dim and \l__marginalia_yshift_computed_dim.)

`\l__marginalia_side_computed_int` Integer variable to indicate the side of the text block or column on which the marginal content item should be placed: 0 = right and 1 = left.

```
295 \int_new:N\l__marginalia_side_computed_int
```

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

(End of definition for \l__marginalia_side_computed_int.)

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

```
296 \int_new:N\l_marginalia_marginno_computed_int
```

(End of definition for `\l_marginalia_marginno_computed_int`.)

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

```
297 \int_new:N\l_marginalia_enabled_computed_int
```

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

(End of definition for `\l_marginalia_enabled_computed_int`.)

11.6.2 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>`.

```
298 \cs_new:Npn\_marginalia_process_item:nn #1#2
299 {
```

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

```
300 \int_gincr:N\g_marginalia_itemno_int
301 \group_begin:
```

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

```
302 \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.

```
303 \__marginalia_lua_load_item_data:n
304 { \int_value:w\g_marginalia_itemno_int }
```

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

```
305 \mode_if_math:TF
306 {
307   \cs_set_eq:NN
308     \__marginalia_typeset:n
309     \__marginalia_typeset_mmode:n
310 }
311 {
312   \legacy_if:nT{@inlabel}
313   { \leavevmode }
314   \mode_if_horizontal:TF
315   {
316     \cs_set_eq:NN
```

```

317         \l__marginalia_typeset:n
318         \l__marginalia_typeset_hmode:n
319     }
320     {
321         \cs_set_eq:NN
322         \l__marginalia_typeset:n
323         \l__marginalia_typeset_vmode:n
324     }
325 }

```

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.

```

326     \int_compare:nNnTF{\l__marginalia_valign_int}={2}
327     {
328         \cs_set_eq:NN\l__marginalia_item_box_set:Nn\ vbox_set:Nn
329     }
330     {
331         \cs_set_eq:NN\l__marginalia_item_box_set:Nn\ vbox_set_top:Nn
332     }

```

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

```

333     \l__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`.

```

334     \l__marginalia_item_box_set:Nn\l__marginalia_item_box{
335         \@parboxrestore
336         \normalfont\normalsize
337
338         \tl_use:N\l__marginalia_style_tl
339         \dim_set_eq:NN\hsize\l__marginalia_width_dim
340         \dim_set_eq:NN\linewidth\hsize
341
342         \cs_set_eq:NN\marginaliapage\l__marginalia_page_int
343         \cs_set_eq:NN\marginaliacolumn\l__marginalia_column_computed_int
344
345         \group_begin:
346         \ignorespaces
347         #2
348         \par
349         \group_end:
350     }

```

Measure `\l__marginalia_item_box`.

```

351     \dim_set:Nn\l__marginalia_item_height_dim
352     {\box_ht:N\l__marginalia_item_box}
353     \dim_set:Nn\l__marginalia_item_depth_dim
354     {\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.

```

355     \l__marginalia_typeset:n{
356         \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.

```

357     \iow_shipout_e:Ne\l__marginalia_aux_iow{
358     \token_to_str:N\marginalia@itemdata{
359         itemno=\int_value:w\g__marginalia_itemno_int,
360         abspageno=\exp_not:N\int_eval:n{\g_shipout_readonly_int},
361         pageno=\exp_not:N\int_value:w\c@page,
362         type=\str_use:N\int_value:w\l__marginalia_type_int,
363         xpos=\exp_not:N\int_value:w\lastxpos,
364         ypos=\exp_not:N\int_value:w\lastypos,
365         height=\int_value:w\l__marginalia_item_height_dim,
366         depth=\int_value:w\l__marginalia_item_depth_dim,
367         pos=\int_value:w\l__marginalia_pos_int,
368         column=\int_value:w\l__marginalia_column_int,
369         yshift=\int_value:w\l__marginalia_default_yshift_dim,
370         ysep~above=\int_value:w\l__marginalia_ysep_above_dim,
371         ysep~below=\int_value:w\l__marginalia_ysep_below_dim,
372         ysep~page~top=\int_value:w\l__marginalia_ysep_page_top_dim,
373         ysep~page~bottom=\int_value:w\l__marginalia_ysep_page_bottom_dim,
374     }
375 }
```

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

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

to the right 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.

```

376     \int_if_zero:nF{\l__marginalia_enabled_computed_int}
377     {
378         \int_if_zero:nTF{\l__marginalia_side_computed_int}
379         {
380             \rlap{
381                 \kern\l__marginalia_xshift_computed_dim
382                 \kern\l__marginalia_xsep_dim
383                 \__marginalia_place_item_box:
384             }
385         }
386         {
387             \llap{
388                 \__marginalia_place_item_box:
389                 \kern\l__marginalia_xsep_dim
390                 \kern-\l__marginalia_xshift_computed_dim
391             }
392         }
393     }
394 }
```

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

```

395     \group_end:
396 }
```

(End of definition for __marginalia_process_item:nn.)

11.6.3 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`.

```

397 \cs_new:Npn\_marginalia_set_xsep_width_style:
398 {
399   \int_case:nn{\l\_marginalia_marginno_computed_int}
400   {
401     {0}
402     {
403       \cs_set_eq:NN\l\_marginalia_xsep_dim
404         \l\_marginalia_xsep_recto_outer_dim
405       \cs_set_eq:NN\l\_marginalia_width_dim
406         \l\_marginalia_width_recto_outer_dim
407       \cs_set_eq:NN\l\_marginalia_style_tl
408         \l\_marginalia_style_recto_outer_tl
409     }
410     {1}
411     {
412       \cs_set_eq:NN\l\_marginalia_xsep_dim
413         \l\_marginalia_xsep_recto_inner_dim
414       \cs_set_eq:NN\l\_marginalia_width_dim
415         \l\_marginalia_width_recto_inner_dim
416       \cs_set_eq:NN\l\_marginalia_style_tl
417         \l\_marginalia_style_recto_inner_tl
418     }
419     {2}
420     {
421       \cs_set_eq:NN\l\_marginalia_xsep_dim
422         \l\_marginalia_xsep_verso_outer_dim
423       \cs_set_eq:NN\l\_marginalia_width_dim
424         \l\_marginalia_width_verso_outer_dim
425       \cs_set_eq:NN\l\_marginalia_style_tl
426         \l\_marginalia_style_verso_outer_tl
427     }
428     {3}
429     {
430       \cs_set_eq:NN\l\_marginalia_xsep_dim
431         \l\_marginalia_xsep_verso_inner_dim
432       \cs_set_eq:NN\l\_marginalia_width_dim
433         \l\_marginalia_width_verso_inner_dim
434       \cs_set_eq:NN\l\_marginalia_style_tl
435         \l\_marginalia_style_verso_inner_tl
436     }
437     {4}
438     {
439       \cs_set_eq:NN\l\_marginalia_xsep_dim
440         \l\_marginalia_xsep_right_between_dim
441       \cs_set_eq:NN\l\_marginalia_width_dim
442         \l\_marginalia_width_right_between_dim
443       \cs_set_eq:NN\l\_marginalia_style_tl
444         \l\_marginalia_style_right_between_tl
445     }
446     {5}

```

```

447     {
448         \cs_set_eq:NN\l__marginalia_xsep_dim
449         \l__marginalia_xsep_left_between_dim
450         \cs_set_eq:NN\l__marginalia_width_dim
451         \l__marginalia_width_left_between_dim
452         \cs_set_eq:NN\l__marginalia_style_tl
453         \l__marginalia_style_left_between_tl
454     }
455 }
456 }

```

(End of definition for `__marginalia_set_xsep_width_style`.)

11.6.4 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.

```

457 \cs_new:Npn\__marginalia_place_item_box:
458 {
459     \smash
460     {
461         \box_move_up:nn{\l__marginalia_yshift_computed_dim}
462         {
463             \box_use:N\l__marginalia_item_box
464         }
465     }
466 }

```

(End of definition for `__marginalia_place_item_box:`.)

`__marginalia_typeset_mmode:n`
`__marginalia_typeset_hmode:n`
`__marginalia_typeset_vmode:n` These three macros handle typesetting in math mode, horizontal mode, and vertical mode. Nothing special needs to be done in math mode. In horizontal mode, `\@bsphack...\@esphack` avoids double spacing. In vertical mode, 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.

```

467 \cs_new:Npn\__marginalia_typeset_mmode:n #1
468 {
469     #1
470 }
471 \cs_new:Npn\__marginalia_typeset_hmode:n #1
472 {
473     \@bsphack
474     #1
475     \@esphack
476 }
477 \cs_new:Npn\__marginalia_typeset_vmode:n #1
478 {
479     \nobreak\noindent\strut #1\par
480     \skip_vertical:n{-\baselineskip}
481 }

```

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

11.7 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`.

```
482 \NewDocumentCommand{\marginalia}{ 0{} +m }  
483 {  
484   \__marginalia_process_item:nn{#1}{#2}  
485 }
```

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

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

```
486 \NewDocumentCommand{\marginaliasetup}{ m }  
487 {  
488   \keys_set:nn{marginalia}{ #1 }  
489 }
```

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

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

```
490 \NewDocumentCommand{\marginalianewgeometry}{}  
491 {  
492   \__marginalia_write_page_data:  
493 }
```

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

```
494 \</package>
```

12 Implementation (Lua backend)

```
495 \<lua>
```

12.1 Global variables

Global tables for `page_data` and `item_data`.

```
496 local PAGE_DATA_MAIN_TABLE = {}  
497 local ITEM_DATA_MAIN_TABLE = {}
```

Global tables for compiling reports.

```
498 local PROBLEM_REPORT_TABLE = {}  
499 local PAGE_CHANGE_REPORT_TABLE = {}  
500 local ITEM_CHANGE_REPORT_TABLE = {}
```

Global configuration for reports.

```
501 local PROBLEM_REPORT_MAX_LENGTH = 40  
502 local PAGE_CHANGE_REPORT_MAX_LENGTH = 10  
503 local ITEM_CHANGE_REPORT_MAX_LENGTH = 10
```

12.2 Constants

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

```
504 local TYPE_NORMAL = 1
505 local TYPE_FIXED = 2
506 local TYPE_OPTFIXED = 3
```

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

```
507 local POS_AUTO = 1
508 local POS_REVERSE = 2
509 local POS_LEFT = 3
510 local POS_RIGHT = 4
511 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

```
512 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

```
513 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

```
514 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.

```
515 local KEY_HOFFSETORIGIN = 'offsetorigin'
516 local KEY_VOFFSETORIGIN = 'voffsetorigin'
```

Dimensions: corresponding to obvious LaTeX dimensions

```
517 local KEY_HOFFSET = 'offset'
518 local KEY_VOFFSET = 'voffset'
519 local KEY_PAPERHEIGHT = 'paperheight'
520 local KEY_ODDSIDEMARGIN = 'oddsidemargin'
521 local KEY_EVENSIDEMARGIN = 'evensidemargin'
522 local KEY_TEXTWIDTH = 'textwidth'
523 local KEY_COLUMNWIDTH = 'columnwidth'
524 local KEY_COLUMNSEP = 'columnsep'
```

Integer: either 1 or 2, depending on whether LaTeX was in one- or two-column mode

```
525 local KEY_COLUMNCOUNT = 'columncount'
```

Boolean: true iff LaTeX is in twoside mode

```
526 local KEY_TWOSIDE = 'twoside'
```

12.3.3 Keys for item data tables

Integer: Used to identify data with item

```
527 local KEY_ITEMNO = 'itemno'
```

Integer: On-page number

```
528 local KEY_PAGENO = 'pageno'
```

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

```
529 local KEY_XPOS = 'xpos'
```

```
530 local KEY_YPOS = 'ypos'
```

Dimensions: Height and depth of typeset item

```
531 local KEY_HEIGHT = 'height'
```

```
532 local KEY_DEPTH = 'depth'
```

Integer: Specified type, following `TYPE_*`

```
533 local KEY_TYPE = 'type'
```

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

```
534 local KEY_POS = 'pos'
```

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

```
535 local KEY_COLUMN = 'column'
```

Dimension: specified vertical shift

```
536 local KEY_YSHIFT = 'yshift'
```

Dimensions: specified vertical separations

```
537 local KEY_YSEP_ABOVE = 'ysep above'
```

```
538 local KEY_YSEP_BELOW = 'ysep below'
```

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

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

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

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

```
541 local KEY_COLNO_COMPUTED = 'colno computed'
```

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

```
542 local KEY_XSHIFT_COMPUTED = 'xshift computed'
```

Dimension: Computed vertical shift

```
543 local KEY_YSHIFT_COMPUTED = 'yshift computed'
```

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

```
544 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

```
545 local KEY_MARGINNO_COMPUTED = 'marginno computed'
```

Boolean: Whether the item will actually appear on the page

```
546 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.)⁷

```
547 local function list_filter(t, f)
548   local j = 1
549   local n = #t
550
551   for i=1,n do
552     if (f(t[i])) then
553       if (i ~= j) then
554         t[j] = t[i]
555         t[i] = nil
556       end
557       j = j + 1
558     else
559       t[i] = nil
560     end
561   end
562
563 end
```

(End of definition for list_filter.)

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

```
564 local function toboolean(s)
565   return s == "true"
566 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.

```
567 local function get_data_page_number(data)
568   local pageno = data[KEY_PAGENO]
569   if pageno ~= nil then
570     return 'p' .. pageno .. ' (' .. data[KEY_ABSPAGENO] .. ')'
571   else
572     return data[KEY_ABSPAGENO]
573   end
574 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`.

```

575 local function parse_data(keyvalue_string,conversion_table,defaults_table)
576
577     local key
578     local value
579     local result = {}
580
581     for s in string.gmatch(keyvalue_string,'([^\,]+)') do
582
583         key,value = string.match(s,'^(.+)=(.+)$')
584         local conv = conversion_table[key]
585         if conv ~= nil then
586             result[key] = conv(value)
587         end
588
589     end
590
591     for key,value in pairs(defaults_table) do
592         if not(result[key] ~= nil) then
593             result[key] = value
594         end
595     end
596
597     return result
598
599 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`.

```

600 local function check_data(keyvalue_string,conversion_table,defaults_table,
601                           data_table,data_table_key_field,report_table)
602
603     local new_data = parse_data(keyvalue_string,
604                                 conversion_table,defaults_table)
605
606     local data_table_key = new_data[data_table_key_field]
607
608     local stored_data = data_table[data_table_key]
609     if stored_data == nil then
610         table.insert(
611             report_table,
612             get_data_page_number(new_data) .. ' New'
613         )
614     else
615         local change_report = ''

```



```

616     for k,_ in pairs(conversion_table) do
617         if stored_data[k] ~= new_data[k] then
618             change_report = change_report
619             .. ' ' .. k .. ':' ..
620             tostring(stored_data[k]) .. '->' .. tostring(new_data[k])
621         end
622     end
623     if change_report ~= '' then
624         table.insert(
625             report_table,
626             get_data_page_number(new_data) .. ' ' .. change_report
627         )
628     end
629     stored_data[KEY_CHECKED] = true
630 end
631
632 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`.

```

633 local function check_removed_data(data_table,report_table)
634     for _,data in pairs(data_table) do
635         if not data[KEY_CHECKED] then
636             table.insert(
637                 report_table,
638                 ' Removed'
639             )
640             break
641         end
642     end
643 end

```

(End of definition for check_removed_data.)

12.6 Processing of page data from .aux file

Conversion and default tables.

```

644 local PAGE_DATA_CONVERSION_TABLE = {
645     [KEY_PAGEDATANO] = tonumber,
646     [KEY_A BSPAGENO] = tonumber,
647     [KEY_HOFFSETORIGIN] = tonumber,
648     [KEY_VOFFSETORIGIN] = tonumber,
649     [KEY_HOFFSET] = tonumber,
650     [KEY_VOFFSET] = tonumber,
651     [KEY_PAPERHEIGHT] = tonumber,
652     [KEY_ODDSIDEMARGIN] = tonumber,
653     [KEY_EVENSIDEMARGIN] = tonumber,
654     [KEY_COLUMNCOUNT] = tonumber,
655     [KEY_COLUMNWIDTH] = tonumber,
656     [KEY_COLUMNSEP] = tonumber,
657     [KEY_TEXTWIDTH] = tonumber,

```

```

658 [KEY_TWOSIDE] = toboolean,
659 }
660 local PAGE_DATA_DEFAULT_TABLE = {
661   [KEY_PAGEDATANO] = 0,
662   [KEY_ABSPAGENO] = 0,
663   [KEY_HOFFSETORIGIN] = tex.sp('1in'),
664   [KEY_VOFFSETORIGIN] = tex.sp('1in'),
665   [KEY_HOFFSET] = tex.dimen['hoffset'],
666   [KEY_VOFFSET] = tex.dimen['voffset'],
667   [KEY_PAPERHEIGHT] = tex.dimen['paperheight'],
668   [KEY_ODDSIDEMARGIN] = tex.dimen['oddsidemargin'],
669   [KEY_EVENSIDEMARGIN] = tex.dimen['evensidemargin'],
670   [KEY_TEXTWIDTH] = tex.dimen['textwidth'],
671   [KEY_COLUMNWIDTH] = tex.dimen['columnwidth'],
672   [KEY_COLUMNSEP] = tex.dimen['columnsep'],
673   [KEY_COLUMNCOUNT] = 1,
674   [KEY_TWOSIDE] = false,
675   [KEY_CHECKED] = false,
676 }

```

store_page_data Store page data supplied by keyvalue_string in PAGE_DATA_MAIN_TABLE.

```

677 local function store_page_data(keyvalue_string)
678
679   local page_data = parse_data(keyvalue_string,
680                                 PAGE_DATA_CONVERSION_TABLE,
681                                 PAGE_DATA_DEFAULT_TABLE)
682
683   PAGE_DATA_MAIN_TABLE[page_data[KEY_PAGEDATANO]] = page_data
684
685 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.

```

686 local function store_default_page_data()
687
688   default_page_data = parse_data('',
689                                   PAGE_DATA_CONVERSION_TABLE,
690                                   PAGE_DATA_DEFAULT_TABLE)
691
692   default_page_data[KEY_ABSPAGENO] = 1
693   default_page_data[KEY_CHECKED] = true
694
695   PAGE_DATA_MAIN_TABLE[0] = default_page_data
696
697 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.

```

698 local function check_page_data(keyvalue_string)

```

```

699
700     check_data(keyvalue_string,
701                 PAGE_DATA_CONVERSION_TABLE,PAGE_DATA_DEFAULT_TABLE,
702                 PAGE_DATA_MAIN_TABLE,KEY_PAGEDATANO,
703                 PAGE_CHANGE_REPORT_TABLE)
704
705 end

```

(End of definition for check_page_data.)

12.7 Processing of item data from .aux file

Conversion and default tables.

```

706 local ITEM_DATA_CONVERSIONS = {
707     [KEY_ITEMNO] = tonumber,
708     [KEY_ABSPAGENO] = tonumber,
709     [KEY_PAGENO] = tonumber,
710     [KEY_XPOS] = tonumber,
711     [KEY_YPOS] = tonumber,
712     [KEY_HEIGHT] = tonumber,
713     [KEY_DEPTH] = tonumber,
714     [KEY_TYPE] = tonumber,
715     [KEY_POS] = tonumber,
716     [KEY_COLUMN] = tonumber,
717     [KEY_YSHIFT] = tonumber,
718     [KEY_YSEP_ABOVE] = tonumber,
719     [KEY_YSEP_BELOW] = tonumber,
720     [KEY_YSEP_PAGE_TOP] = tonumber,
721     [KEY_YSEP_PAGE_BOTTOM] = tonumber,
722     [KEY_CHECKED] = toboolean,
723 }
724 local ITEM_DATA_DEFAULTS = {
725     [KEY_ITEMNO] = 0,
726     [KEY_ABSPAGENO] = 1,
727     [KEY_PAGENO] = 1,
728     [KEY_XPOS] = 0,
729     [KEY_YPOS] = 0,
730     [KEY_HEIGHT] = 0,
731     [KEY_DEPTH] = 0,
732     [KEY_TYPE] = 0,
733     [KEY_POS] = 0,
734     [KEY_COLUMN] = -1,
735     [KEY_YSHIFT] = 0,
736     [KEY_YSEP_ABOVE] = tex.dimen['marginparpush'],
737     [KEY_YSEP_BELOW] = tex.dimen['marginparpush'],
738     [KEY_YSEP_PAGE_TOP] = tex.dimen['marginparpush'],
739     [KEY_YSEP_PAGE_BOTTOM] = tex.dimen['marginparpush'],
740     [KEY_COLNO_COMPUTED] = 0,
741     [KEY_XSHIFT_COMPUTED] = 0,
742     [KEY_YSHIFT_COMPUTED] = 0,
743     [KEY_SIDE_COMPUTED] = 0,
744     [KEY_MARGINNO_COMPUTED] = 0,
745     [KEY_ENABLED_COMPUTED] = true,
746     [KEY_CHECKED] = false,

```

```
747 }
```

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.

```
748 local function store_item_data(keyvalue_string)
749
750   local item = parse_data(keyvalue_string,
751                           ITEM_DATA_CONVERSIONS,
752                           ITEM_DATA_DEFAULTS)
753
754   ITEM_DATA_MAIN_TABLE[item[KEY_ITEMNO]] = item
755
756 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.

```
757 local function check_item_data(keyvalue_string)
758
759   check_data(keyvalue_string,
760             ITEM_DATA_CONVERSIONS, ITEM_DATA_DEFAULTS,
761             ITEM_DATA_MAIN_TABLE, KEY_ITEMNO,
762             ITEM_CHANGE_REPORT_TABLE)
763
764 end
```

(End of definition for check_item_data.)

12.8 Writing reports

write_report Write the data contained in report_table to T_EX in a format suitable for a package warning. The written text will contain at most max_length items.

```
765 local function write_report(report_table,max_length)
766
767   if #report_table > 0 then
768     local report_text
769     local report_length
770
771     if #report_table <= max_length then
772       report_length = #report_table
773       report_text = ' Here they are:\n'
774     else
775       report_length = max_length
776       report_text = ' Here are the first ' .. report_length .. ':\n'
777     end
778
779     for i=1,report_length do
780       report_text = report_text .. report_table[i]
781       if i < report_length then
782         report_text = report_text .. '\n'
783       end
784     end
785   end
```

```

784     end
785
786     tex.print(report_text)
787 end
788
789 end

```

(End of definition for write_report.)

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

```

790 local function write_problem_report()
791
792     write_report(PROBLEM_REPORT_TABLE, PROBLEM_REPORT_MAX_LENGTH)
793
794 end

```

(End of definition for write_problem_report.)

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

```

795 local function write_item_change_report()
796
797     check_removed_data(ITEM_DATA_MAIN_TABLE, ITEM_CHANGE_REPORT_TABLE)
798     write_report(ITEM_CHANGE_REPORT_TABLE, ITEM_CHANGE_REPORT_MAX_LENGTH)
799
800 end

```

(End of definition for write_item_change_report.)

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

```

801 local function write_page_change_report()
802
803     check_removed_data(PAGE_DATA_MAIN_TABLE, PAGE_CHANGE_REPORT_TABLE)
804     write_report(PAGE_CHANGE_REPORT_TABLE, PAGE_CHANGE_REPORT_MAX_LENGTH)
805
806 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`.

```

807 local RIGHTSIDE_LOOKUP_TABLE = {
808     [true] = {
809         [0] = {
810             [POS_AUTO] = true,
811             [POS_REVERSE] = false,

```

```

812     },
813     [1] = {
814         [POS_AUTO] = false,
815         [POS_REVERSE] = true,
816     },
817     [2] = {
818         [POS_AUTO] = true,
819         [POS_REVERSE] = false,
820     },
821 },
822 [false] = {
823     [0] = {
824         [POS_AUTO] = false,
825         [POS_REVERSE] = true,
826     },
827     [1] = {
828         [POS_AUTO] = true,
829         [POS_REVERSE] = false,
830     },
831     [2] = {
832         [POS_AUTO] = false,
833         [POS_REVERSE] = true,
834     },
835 },
836 }

```

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).

```

837 local MARGINNO_LOOKUP_TABLE = {
838     [true] = {
839         [0] = {
840             [false] = 1,
841             [true] = 0,
842         },
843         [1] = {
844             [false] = 1,
845             [true] = 5,
846         },
847         [2] = {
848             [false] = 4,
849             [true] = 0,
850         },
851     },
852     [false] = {
853         [0] = {
854             [false] = 2,
855             [true] = 3,
856         },
857         [1] = {
858             [false] = 2,
859             [true] = 5,

```

```

860     },
861     [2] = {
862         [false] = 4,
863         [true] = 3,
864     },
865 },
866 }

```

`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.

```

867 local function compute_items_horizontal(item_data_list,page_data)

```

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

```

868     if #item_data_list == 0 then
869         return
870     end

```

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

```

871     local pageno = item_data_list[1][KEY_PAGENO]
872     local twoside = page_data[KEY_TWOSIDE]
873     local recto = ((pageno % 2) == 1) or (not twoside)
874     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).

```

875     local x_textleft = {}
876     local x_textright = {}
877     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.

```

878     if recto then
879         x_textleft[0] = (
880             page_data[KEY_HOFFSETORIGIN]
881             + page_data[KEY_HOFFSET]
882             + page_data[KEY_ODDSIDEMARGIN]
883         )
884         x_textright[0] = (
885             x_textleft[0]
886             + page_data[KEY_TEXTWIDTH]
887         )
888     else
889         x_textleft[0] = (
890             page_data[KEY_HOFFSETORIGIN]
891             + page_data[KEY_HOFFSET]
892             + page_data[KEY_EVENSIDEMARGIN]
893         )
894         x_textright[0] = (
895             x_textleft[0]
896             + page_data[KEY_TEXTWIDTH]

```

```

897     )
898   end
899   x_textmiddle[0] = (x_textleft[0] + x_textright[0])/2
900
901
902   if columncount == 1 then

```

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

```

903     for i=1,#item_data_list do
904       item_data_list[i][KEY_COLNO_COMPUTED] = 0
905     end
906   else

```

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

```

907     x_textleft[1] = x_textleft[0]
908     x_textright[1] = (
909       x_textleft[1]
910       + page_data[KEY_COLUMNWIDTH]
911     )
912     x_textmiddle[1] = (x_textleft[1] + x_textright[1])/2
913
914     x_textleft[2] = (
915       x_textright[1]
916       + page_data[KEY_COLUMNSEP]
917     )
918     x_textright[2] = (
919       x_textleft[2]
920       + page_data[KEY_COLUMNWIDTH]
921     )
922     x_textmiddle[2] = (x_textleft[2] + x_textright[2])/2
923

```

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

```

924     local left_column_x_limit = (
925       x_textright[1]
926       + .5*page_data[KEY_COLUMNSEP]
927     )

```

Now set the field KEY_COLNO_COMPUTED for each item.

```

928     for i=1,#item_data_list do
929       local item_data = item_data_list[i]
930
931       if item_data[KEY_COLUMN] >= 0 then
932         item_data[KEY_COLNO_COMPUTED] = item_data[KEY_COLUMN]
933       else
934         if item_data[KEY_XPOS] <= left_column_x_limit then
935           item_data[KEY_COLNO_COMPUTED] = 1
936         else
937           item_data[KEY_COLNO_COMPUTED] = 2
938         end
939       end
940     end

```



```

941
942 end

```

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

```

943 for i=1,#item_data_list do
944     local item = item_data_list[i]
945
946     local pos = item[KEY_POS]
947     local colnocomputed = item[KEY_COLNO_COMPUTED]
948
949     if pos == POS_LEFT then
950         rightside = false
951     elseif pos == POS_RIGHT then
952         rightside = true
953     elseif pos == POS_NEAREST then
954         rightside = (item[KEY_XPOS] >= x_textmiddle[colnocomputed])
955     else

```

pos must be POS_AUTO or POS_REVERSE

```

956         rightside = RIGHTSIDE_LOOKUP_TABLE[recto][colnocomputed][pos]
957     end
958
959     local marginno = MARGINNO_LOOKUP_TABLE[recto][colnocomputed][rightside]
960
961     if rightside then
962         item[KEY_SIDE_COMPUTED] = 0
963         item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
964                                     + x_textright[colnocomputed]
965     else
966         item[KEY_SIDE_COMPUTED] = 1
967         item[KEY_XSHIFT_COMPUTED] = -item[KEY_XPOS]
968                                     + x_textleft[colnocomputed]
969     end
970     item[KEY_MARGINNO_COMPUTED] = marginno
971
972 end
973
974 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.

```

975 local function get_y_item_top(item_data)
976     return item_data[KEY_YPOS]
977         + item_data[KEY_YSHIFT_COMPUTED]
978         + item_data[KEY_HEIGHT]
979 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.

```

980 local function get_y_item_bottom(item_data)
981     return item_data[KEY_YPOS]
982         - item_data[KEY_DEPTH]

```

```

983         + item_data[KEY_YSHIFT_COMPUTED]
984     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).

```

985 local function get_ysep_list(item_data_list)
986
987     local ysep_list = {}
988
989     ysep_list[0] = 0
990     for i=1,#item_data_list-1 do
991         ysep_list[i] = math.max(
992             item_data_list[i][KEY_YSEP_BELOW],
993             item_data_list[i+1][KEY_YSEP_ABOVE]
994         )
995     end
996     ysep_list[#item_data_list] = 0
997
998     return ysep_list
999
1000 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.

```

1001 local function compute_items_vertical_optfixed_enabled(item_data_list)
1002
1003     local optfixed_item_data_list = {}
1004     local fixed_item_data_list = {}
1005
1006     for _,item_data in pairs(item_data_list) do
1007         if item_data[KEY_TYPE] == TYPE_OPTFIXED then
1008             optfixed_item_data_list[#optfixed_item_data_list+1] = item_data
1009         elseif item_data[KEY_TYPE] == TYPE_FIXED then

```

```

1010     fixed_item_data_list[#fixed_item_data_list+1] = item_data
1011   end
1012 end
1013
1014 for _,optfixed_item_data in pairs(optfixed_item_data_list) do
1015   local optfixed_y_item_top = get_y_item_top(optfixed_item_data)
1016   local optfixed_y_item_bottom = get_y_item_bottom(optfixed_item_data)
1017
1018   for _,fixed_item_data in pairs(fixed_item_data_list) do
1019     local fixed_y_item_top = get_y_item_top(fixed_item_data)
1020     local fixed_y_item_bottom = get_y_item_bottom(fixed_item_data)
1021
1022     if (
1023       (
1024         (fixed_y_item_bottom - optfixed_y_item_top)
1025         <
1026         math.max(
1027           fixed_item_data[KEY_YSEP_BELOW],
1028           optfixed_item_data[KEY_YSEP_ABOVE]
1029         )
1030       )
1031       and
1032       (
1033         (optfixed_y_item_bottom - fixed_y_item_top)
1034         <
1035         math.max(
1036           optfixed_item_data[KEY_YSEP_BELOW],
1037           fixed_item_data[KEY_YSEP_ABOVE]
1038         )
1039       )
1040     ) then
1041       optfixed_item_data[KEY_ENABLED_COMPUTED] = false
1042       break
1043     end
1044   end
1045 end
1046
1047 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.

```

1048 local function compute_items_vertical_adjustment(item_data_list,page_data)

```

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

```

1049   if #item_data_list == 0 then
1050     return
1051   end

```

```

1052
1053   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.

```

1054   local y_limit_above = (
1055     page_data[KEY_VOFFSET]
1056     + page_data[KEY_PAPERHEIGHT]
1057     - item_data_list[1][KEY_YSEP_PAGE_TOP]
1058   )
1059
1060   for i=1,#item_data_list do
1061     local item_data = item_data_list[i]
1062
1063     local y_item_top = get_y_item_top(item_data)
1064
1065     if y_item_top > y_limit_above then
1066       if item_data[KEY_TYPE] == TYPE_NORMAL then
1067         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1068                                           + (y_limit_above - y_item_top)
1069       end
1070     end
1071
1072     y_limit_above = get_y_item_bottom(item_data) - ysep_list[i]
1073   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.

```

1074   local y_limit_below = (
1075     page_data[KEY_VOFFSET]
1076     + item_data_list[#item_data_list][KEY_YSEP_PAGE_BOTTOM]
1077   )
1078
1079   for i=#item_data_list,1,-1 do
1080     local item_data = item_data_list[i]
1081
1082     local y_item_bottom = get_y_item_bottom(item_data)
1083
1084     if y_item_bottom < y_limit_below then
1085       if item_data[KEY_TYPE] == TYPE_NORMAL then
1086         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT_COMPUTED]
1087                                           + (y_limit_below - y_item_bottom)
1088       end
1089     end
1090
1091     y_limit_below = get_y_item_top(item_data) + ysep_list[i-1]
1092   end
1093
1094   end

```

(End of definition for compute_items_vertical_adjustment.)

12.10.3 Checking vertical adjustment

Messages to use when checking results of vertical adjustment.

```

1095 local ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES = {
1096   [TYPE_NORMAL] = 'Moveable item > ysep page top',
1097   [TYPE_FIXED] = 'Topmost fixed item > ysep page top',
1098   [TYPE_OPTFIXED] = 'Topmost optfixed item > ysep page top',
1099 }
1100 local ITEM_CLASH_MESSAGES = {
1101   [TYPE_NORMAL] = {
1102     [TYPE_NORMAL] = 'moveable items'
1103     .. ' (this shouldn\'t happen)',
1104     [TYPE_FIXED] = 'moveable item above fixed item',
1105     [TYPE_OPTFIXED] = 'moveable item above optfixed item',
1106   },
1107   [TYPE_FIXED] = {
1108     [TYPE_NORMAL] = 'moveable item below fixed item',
1109     [TYPE_FIXED] = 'fixed items',
1110     [TYPE_OPTFIXED] = 'fixed item above optfixed item '
1111     .. ' (this shouldn\'t happen)',
1112   },
1113   [TYPE_OPTFIXED] = {
1114     [TYPE_NORMAL] = 'moveable items below optfixed item',
1115     [TYPE_FIXED] = 'fixed item below optfixed item '
1116     .. ' (this shouldn\'t happen)',
1117     [TYPE_OPTFIXED] = 'optfixed items '
1118     .. ' (this shouldn\'t happen)',
1119   },
1120 }
1121 local ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE = {
1122   [TYPE_NORMAL] = 'Moveable item < ysep page bottom',
1123   [TYPE_FIXED] = 'Bottommost fixed item < ysep page bottom',
1124   [TYPE_OPTFIXED] = 'Bottommost optfixed item < ysep page bottom',
1125 }

```

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.

```

1126 local function check_items_vertical(item_data_list,page_data)

```

Immediately return if item_data_list is empty, to avoid edge cases

```

1127   if (#item_data_list) == 0 then
1128     return
1129   end

```

```

1131   local ysep_list = get_ysep_list(item_data_list)

```

```

1132

```

```

1133   local item_data

```

```

1134

```

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

```

1135   item_data = item_data_list[1]
1136   if (
1137     get_y_item_top(item_data) > page_data[KEY_VOFFSET]
1138     + page_data[KEY_PAPERHEIGHT]
1139     - item_data[KEY_YSEP_PAGE_TOP]
1140   ) then
1141     table.insert(

```

```

1142     PROBLEM_REPORT_TABLE,
1143     get_data_page_number(item_data)
1144     .. ' ' .. ITEM_PASSED_YSEP_PAGE_TOP_MESSAGES[item_data[KEY_TYPE]]
1145 )
1146 end
1147
1148 for i=2,#item_data_list do
1149     local item_data = item_data_list[i]
1150     local prev_item_data = item_data_list[i-1]
1151     if (
1152         get_y_item_top(item_data) > get_y_item_bottom(prev_item_data)
1153                                     - ysep_list[i-1]
1154     ) then
1155         table.insert(
1156             PROBLEM_REPORT_TABLE,
1157             get_data_page_number(item_data)
1158             .. ' Clash: ' ..
1159             ITEM_CLASH_MESSAGES[prev_item_data[KEY_TYPE]][item_data[KEY_TYPE]]
1160         )
1161     end
1162 end

```

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

```

1163 item_data = item_data_list[#item_data_list]
1164 if (
1165     get_y_item_bottom(item_data) < page_data[KEY_VOFFSET]
1166                                     + item_data[KEY_YSEP_PAGE_BOTTOM]
1167 ) then
1168     table.insert(
1169         PROBLEM_REPORT_TABLE,
1170         get_data_page_number(item_data)
1171         .. ' ' .. ITEM_PASSED_YSEP_PAGE_BOTTOM_MESSAGE[item_data[KEY_TYPE]]
1172     )
1173 end
1174
1175 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`.

```

1176 local function compute_items_vertical(item_data_list,page_data)

```

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

```

1177     for i=1,#item_data_list do
1178         local item_data = item_data_list[i]
1179
1180         item_data[KEY_YSHIFT_COMPUTED] = item_data[KEY_YSHIFT]
1181     end

```

Decide which items of type ITEM_DATA_OPTFIXED are to be disabled.

```
1182 compute_items_vertical_optfixed_enabled(item_data_list)
```

Strip any item_data with KEY_ENABLED_COMPUTED set to false from item_data_list.

```
1183 list_filter(item_data_list,function(item_data)
1184     return item_data[KEY_ENABLED_COMPUTED]
1185 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.

```
1186 table.sort(
1187     item_data_list,
1188     function(left,right)
1189         local y_diff = left[KEY_YPOS] - right[KEY_YPOS]
1190
1191         if y_diff > 0 then
1192             return true
1193         elseif y_diff < 0 then
1194             return false
1195         end
1196
1197         local x_diff = left[KEY_XPOS] - right[KEY_XPOS]
1198
1199         if x_diff < 0 then
1200             return true
1201         elseif x_diff > 0 then
1202             return false
1203         end
1204
1205         return (left[KEY_ITEMNO] < right[KEY_ITEMNO])
1206     end
1207 )
1208
1209 compute_items_vertical_adjustment(item_data_list,page_data)
1210
1211 check_items_vertical(item_data_list,page_data)
1212
1213 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.

```
1214 local function compute_items()
```

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

```
1215 local max_abspageno = 0
1216
1217 for k,v in pairs(ITEM_DATA_MAIN_TABLE) do
1218     max_abspageno = math.max(v[KEY_A BSPAGENO],max_abspageno)
1219 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.

```
1220 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`.

```
1221 for i=1,max_abspageno do
1222     per_abspage_item_data_list[i] = {}
1223 end
1224 for _,item_data in pairs(ITEM_DATA_MAIN_TABLE) do
1225     local temp_table = per_abspage_item_data_list[item_data[KEY_ABSPAGENO]]
1226     temp_table[#temp_table+1] = item_data
1227 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`.

```
1228 local per_abspage_page_data_list = {}
1229 local pagedatano = 0
1230 for abspageno = 1,max_abspageno do
1231     while (
1232         PAGE_DATA_MAIN_TABLE[pagedatano+1] ~= nil
1233         and
1234         PAGE_DATA_MAIN_TABLE[pagedatano+1][KEY_ABSPAGENO] == abspageno
1235     ) do
1236         pagedatano = pagedatano+1
1237     end
1238     per_abspage_page_data_list[abspageno] = PAGE_DATA_MAIN_TABLE[pagedatano]
1239 end
```

Iterate through all pages and perform the necessary computations.

```
1240 for abspageno=1,#per_abspage_item_data_list do
1241     local current_page_data = per_abspage_page_data_list[abspageno]
1242     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.

```
1243     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.

```
1244     local current_page_item_data_sublists = {}
1245
1246     for i=0,5 do
1247         current_page_item_data_sublists[i] = {}
1248     end
1249
1250     for _,item_data in pairs(current_page_item_data_list) do
1251         table.insert(
```



```

1252         current_page_item_data_sublists[item_data[KEY_MARGINNO_COMPUTED]],
1253         item_data
1254     )
1255 end

```

Compute vertical positons for each sublist.

```

1256     for i=0,5 do
1257         compute_items_vertical(
1258             current_page_item_data_sublists[i],
1259             current_page_data
1260         )
1261     end
1262 end
1263 end

```

(End of definition for compute_items.)

12.11 Passing item_data back to L^AT_EX

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

```

1264 local function load_item_data(itemno)
1265
1266     item = ITEM_DATA_MAIN_TABLE[tonumber(itemno)]
1267     if item == nil then
1268         item = ITEM_DATA_DEFAULTS
1269     end
1270
1271     tex.count['l__marginalia_page_int'] = item[KEY_PAGENO]
1272     tex.count['l__marginalia_column_computed_int'] = item[KEY_COLNO_COMPUTED]
1273     tex.dimen['l__marginalia_xshift_computed_dim'] = item[KEY_XSHIFT_COMPUTED]
1274     tex.dimen['l__marginalia_yshift_computed_dim'] = item[KEY_YSHIFT_COMPUTED]
1275     tex.count['l__marginalia_side_computed_int'] = item[KEY_SIDE_COMPUTED]
1276     tex.count['l__marginalia_marginno_computed_int']
1277         = item[KEY_MARGINNO_COMPUTED]
1278     if item[KEY_ENABLED_COMPUTED] then
1279         tex.count['l__marginalia_enabled_computed_int'] = 1
1280     else
1281         tex.count['l__marginalia_enabled_computed_int'] = 0
1282     end
1283
1284 end

```

(End of definition for load_item_data.)

12.12 Export public functions

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

```

1285 return {
1286     store_default_page_data = store_default_page_data,
1287     store_page_data = store_page_data,
1288     check_page_data = check_page_data,
1289 }

```

```
1290 store_item_data = store_item_data,  
1291 check_item_data = check_item_data,  
1292  
1293 compute_items = compute_items,  
1294  
1295 load_item_data = load_item_data,  
1296  
1297 write_problem_report = write_problem_report,  
1298  
1299 write_page_change_report = write_page_change_report,  
1300 write_item_change_report = write_item_change_report,  
1301 }  
1302 </lua>
```

Index

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

Symbols	
\'	1103, 1111, 1116, 1118
A	
\AddToHook	216, 221, 255, 280
B	
\baselineskip	27, 480
bool commands:	
\bool_to_str:n	276
box commands:	
\box_dp:N	354
\box_ht:N	352
\box_move_up:nn	461
\box_new:N	288
\box_use:N	463
C	
check commands:	
check_data	600
check_item_data	757
check_items_vertical	1126
check_page_data	698
check_removed_data	633
column (option)	6
\columnsep	275
\columnwidth	274
compute commands:	
compute_items	1214
compute_items_horizontal	867
compute_items_vertical	1176
compute_items_vertical_adjustment	1048
compute_items_vertical_optfixed-enabled	1001
cs commands:	
\cs_new:Npn	164, 168, 172, 176, 180, 184, 188, 192, 196, 200, 235, 260, 298, 397, 457, 467, 471, 477
\cs_set_eq:NN	207, 213, 219, 222, 225, 259, 282, 307, 316, 321, 328, 331, 342, 343, 403, 405, 407, 412, 414, 416, 421, 423, 425, 430, 432, 434, 439, 441, 443, 448, 450, 452
D	
\DeclareDocumentCommand	210
dim commands:	
\dim_new:N	289, 290, 293, 294
\dim_set:Nn	351, 353
\dim_set_eq:NN	339, 340
E	
\evensidemargin	39, 271
exp commands:	
\exp_not:N	360, 361, 363, 364
G	
get commands:	
get_data_page_number	567
get_y_item_bottom	980
get_y_item_top	975
get_ysep_list	985
group commands:	
\group_begin:	237, 301, 345
\group_end:	253, 349, 395
H	
\hoffset	39, 267
\hsize	9, 24, 339, 340
I	
\ignorespaces	346
int commands:	
\int_case:nn	399
\int_compare:nNnTF	326
\int_eval:n	266, 360
\int_gincr:N	262, 300
\int_if_zero:nTF	376, 378
\int_new:N	12, 20, 28, 74, 258, 287, 291, 292, 295, 296, 297
\int_set:Nn	16, 24, 32
\int_set_eq:NN	78
\int_value:w	265, 267, 268, 269, 270, 271, 272, 273, 274, 275, 304, 359, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373
iow commands:	
\iow_now:Nn	263
\iow_shipout_e:Nn	357
K	
\kern	381, 382, 389, 390
keys commands:	
\l_keys_choice_int	16, 24, 32, 78
\keys_define:nn	13, 21, 29, 36, 75, 82, 87, 103, 141
\keys_set:nn	45, 55, 61, 67, 94, 112, 122, 128, 134, 150, 302, 488

L

`\lastxpos` 363
`\lastypos` 364
`\leavevmode` 313
 legacy commands:
 `\legacy_if:nTF` 312
 `\legacy_if_p:n` 276
`\linewidth` 24, 340
 list commands:
 `list_filter` 547, 564
`\llap` 3, 25, 387
 load commands:
 `load_item_data` 1264
 lua commands:
 `\lua_now:n` 18, 19, 49, 161, 166, 170,
 174, 178, 182, 186, 190, 194, 198, 202

M

`\marginalia` 3–7, 10–13, 22, 30, 482
 marginalia internal commands:
 `\l_marginalia_aux_iow` . 219, 263, 357
 `\l_marginalia_column_computed_-int` 23, 292, 343
 `\l_marginalia_column_int` .. 28, 368
 `\l_marginalia_default_yshift_-dim` 82, 369
 `\l_marginalia_enabled_computed_-int` 23, 297, 376
 `\l_marginalia_item_box`
 24, 27, 288, 334, 352, 354, 463
 `_marginalia_item_box_set:Nn` ..
 328, 331, 334
 `\l_marginalia_item_depth_dim` ..
 289, 353, 366
 `\l_marginalia_item_height_dim` .
 289, 351, 365
 `\g_marginalia_itemno_int`
 287, 300, 304, 359
 `_marginalia_lua_check_item_-data:n` 20, 164, 184, 227
 `_marginalia_lua_check_page_-data:n` 20, 164, 172, 224
 `_marginalia_lua_compute_items:`
 164, 188, 218
 `_marginalia_lua_load_item_-data:n` 22, 200, 200, 303
 `_marginalia_lua_store_default_-page_data:` 164, 164, 217
 `_marginalia_lua_store_item_-data:n` 19, 164, 180, 215
 `_marginalia_lua_store_page_-data:n` 19, 164, 168, 209
 `_marginalia_lua_write_item_-change_report:` 164, 196, 243

`_marginalia_lua_write_page_-change_report:` 176, 248
`_marginalia_lua_write_problem_-report:` 164, 192, 238
`\l_marginalia_marginno_computed_-int` 26, 296, 399
`\l_marginalia_page_int` . 23, 291, 342
`\g_marginalia_pagedatano_int` ..
 258, 262, 265
`_marginalia_place_item_box` 25
`_marginalia_place_item_box:` ..
 383, 388, 457, 457
`\l_marginalia_pos_int` 20, 367
`_marginalia_process_item:nn` ..
 25, 28, 298, 298, 484
`_marginalia_process_item_-data:n` 19, 20, 211, 214, 226
`_marginalia_process_page_-data:n` 19, 20, 205, 208, 223
`_marginalia_set_xsep_width_-style` 397
`_marginalia_set_xsep_width_-style:` 333, 397
`\l_marginalia_side_computed_int`
 23, 25, 295, 378
`\l_marginalia_style_left_-between_tl` 141, 453
`\l_marginalia_style_recto_-inner_tl` 141, 417
`\l_marginalia_style_recto_-outer_tl` 141, 408
`\l_marginalia_style_right_-between_tl` 141, 444
`\l_marginalia_style_tl`
 26, 338, 407, 416, 425, 434, 443, 452
`\l_marginalia_style_verso_-inner_tl` 141, 435
`\l_marginalia_style_verso_-outer_tl` 141, 426
`\l_marginalia_type_int` 12, 362
`_marginalia_typeset:n`
 308, 317, 322, 355
`_marginalia_typeset_hmmode:n` . 467
`_marginalia_typeset_hmode:n` ..
 318, 471
`_marginalia_typeset_mmode:n` ..
 309, 467, 467
`_marginalia_typeset_vmode:n` ..
 323, 467, 477
`\l_marginalia_valign_int` 24, 74, 326
`\l_marginalia_width_dim` 24,
 26, 339, 405, 414, 423, 432, 441, 450
`\l_marginalia_width_left_-between_dim` 103, 451

<code>\l_marginalia_width_recto_-</code>		<code>\msg_new:nnn</code>	8, 229, 231, 233
<code>inner_dim</code>	103, 415	<code>\msg_warning:nnn</code>	241, 246, 251
<code>\l_marginalia_width_recto_-</code>			
<code>outer_dim</code>	103, 406		
<code>\l_marginalia_width_right_-</code>			
<code>between_dim</code>	103, 442		
<code>\l_marginalia_width_verso_-</code>			
<code>inner_dim</code>	103, 433		
<code>\l_marginalia_width_verso_-</code>			
<code>outer_dim</code>	103, 424		
<code>_marginalia_write_page_data</code> . .	259		
<code>_marginalia_write_page_data:</code> .			
.	21, 259, 283, 285, 492		
<code>_marginalia_write_page_data_-</code>			
<code>real:</code>	260, 284		
<code>_marginalia_write_reports:</code> . . .			
.	229, 235, 256		
<code>\l_marginalia_xsep_dim</code> . .	25, 26,		
	382, 389, 403, 412, 421, 430, 439, 448		
<code>\l_marginalia_xsep_left_-</code>			
<code>between_dim</code>	36, 449		
<code>\l_marginalia_xsep_recto_inner_-</code>			
<code>dim</code>	36, 413		
<code>\l_marginalia_xsep_recto_outer_-</code>			
<code>dim</code>	36, 404		
<code>\l_marginalia_xsep_right_-</code>			
<code>between_dim</code>	36, 440		
<code>\l_marginalia_xsep_verso_inner_-</code>			
<code>dim</code>	36, 431		
<code>\l_marginalia_xsep_verso_outer_-</code>			
<code>dim</code>	36, 422		
<code>\l_marginalia_xshift_computed_-</code>			
<code>dim</code>	23, 25, 293, 381, 390		
<code>\l_marginalia_ysep_above_dim</code> <u>87</u> , 370			
<code>\l_marginalia_ysep_below_dim</code> <u>87</u> , 371			
<code>\l_marginalia_ysep_page_bottom_-</code>			
<code>dim</code>	<u>87</u> , 373		
<code>\l_marginalia_ysep_page_top_dim</code>			
.	<u>87</u> , 372		
<code>\l_marginalia_yshift_computed_-</code>			
<code>dim</code>	23, 27, 293, 461		
<code>\marginaliacolumn</code>	5, 22, 24, 343		
<code>\marginalianewgeometry</code>	5, 21, 48, 490		
<code>\marginaliapage</code>	5, 22, 24, 342		
<code>\marginaliasetup</code>	5, 7, 486		
<code>\marginpar</code>	1, 3, 13		
<code>\marginparpush</code>	7, 8, 101		
<code>\marginparsep</code>	8, 72		
<code>\marginparwidth</code>	7, 10, 139		
mode commands:			
<code>\mode_if_horizontal:TF</code>	314		
<code>\mode_if_math:TF</code>	305		
msg commands:			
<code>\msg_critical:nn</code>	10		

P	
<code>\paperheight</code>	269
<code>\par</code>	348, 479
parse commands:	
<code>parse_data</code>	575
<code>pos</code> (option)	6
prg commands:	
<code>\prg_do_nothing:</code>	259
<code>\ProvidesExplPackage</code>	4
R	
<code>\rlap</code>	3, 6, 25, 380
S	
<code>\savepos</code>	24, 356
shipout commands:	
<code>\g_shipout_readonly_int</code> ...	266, 360
skip commands:	
<code>\skip_vertical:n</code>	480
<code>\smash</code>	27, 459
store commands:	
<code>store_default_page_data</code>	686
<code>store_item_data</code>	748
<code>store_page_data</code>	677
str commands:	
<code>\str_use:N</code>	362
<code>\strut</code>	27, 479
<code>style</code> (option)	10
<code>style left between</code> (option)	10
<code>style recto inner</code> (option)	10
<code>style recto outer</code> (option)	10
<code>style right between</code> (option)	10
<code>style verso inner</code> (option)	10
<code>style verso outer</code> (option)	10
sys commands:	
<code>\sys_if_engine luatex:TF</code>	6
T	
T _E X and L ^A T _E X 2 _ε commands:	
<code>\@bsphack</code>	27, 473
<code>\@esphack</code>	475
<code>\@mainaux</code>	219
<code>\@parboxrestore</code>	24, 335
<code>\c@page</code>	361
<code>\col@number</code>	273
<code>\marginalia@itemdata</code> .	19, 20, 210, 358
<code>\marginalia@notedata</code>	31
<code>\marginalia@pagedata</code>	19, 20, 31, 204, 264
<code>\textwidth</code>	272
tl commands:	
<code>\tl_if_blank:nTF</code>	239, 244, 249
<code>\tl_set:Nn</code>	238, 243, 248
<code>\tl_use:N</code>	241, 246, 251, 338
<code>\l_tmpa_tl</code>	238,
239, 241, 243, 244, 246, 248, 249, 251	
token commands:	
<code>\token_to_str:N</code>	264, 358
<code>\twocolumn</code>	5
<code>type</code> (option)	6
V	
<code>valign</code> (option)	8
<code>\vbox</code>	8
vbox commands:	
<code>\vbox_set:Nn</code>	328
<code>\vbox_set_top:Nn</code>	331
<code>\voffset</code>	268
<code>\vtop</code>	8
W	
<code>width</code> (option)	9
<code>width between</code> (option)	9
<code>width inner</code> (option)	9
<code>width left between</code> (option)	9
<code>width outer</code> (option)	9
<code>width recto inner</code> (option)	9
<code>width recto outer</code> (option)	9
<code>width right between</code> (option)	9
<code>width verso inner</code> (option)	9
<code>width verso outer</code> (option)	9
write commands:	
<code>write_item_change_report</code>	795
<code>write_page_change_report</code>	801
<code>write_problem_report</code>	790
<code>write_report</code>	765
X	
<code>xsep</code> (option)	8
<code>xsep between</code> (option)	8
<code>xsep inner</code> (option)	8
<code>xsep left between</code> (option)	8
<code>xsep outer</code> (option)	8
<code>xsep recto inner</code> (option)	8
<code>xsep recto outer</code> (option)	8
<code>xsep right between</code> (option)	8
<code>xsep verso inner</code> (option)	8
<code>xsep verso outer</code> (option)	8
Y	
<code>ysep</code> (option)	8
<code>ysep above</code> (option)	8
<code>ysep below</code> (option)	8
<code>ysep page bottom</code> (option)	8
<code>ysep page top</code> (option)	8
<code>yshift</code> (option)	8