Newspaper Layout

Reference Manual

Robert Lee

22nd April 2024

1 Abstract

This manual discusses the reference implementation for a newspaper typesetting algorithm.

The editor produces a number of inserts, including articles, titles, pictures and so on, and they are typeset into an attractive newspaper layout. This frees the editor from most typesetting tasks, while still allowing them nearly full control over the process.

2 Installation Requirements

OpenJDK-11.0.2

To use the programme, a Java Runtime Environment (JRE) is required. A full Java Development Kit (JDK) can be used to compile the source code. The programme is developed against Java 11; newer versions should also work. At time of writing, there are no external Java dependencies required at runtime.

LaTeX 2ε

Either PdfLaTex or LuaLaTex is recommended, and LuaLaTeX is strongly recommended if you want to use markdown in your input articles. This is used both to measure the size of articles prior to layout, and to perform the final typesetting of the document. The following packages are required:

multicol	Used by headspan articles
fontsize	Used to set up the default font size commands and associated spacing.
inputenc	Used to read UTF-8 characters in inputs
newtx	Supplies the default *Times Roman* font
geometry	Used when measuring article sizes, and to set the final paper size.
babel	Used per default to set hyphenation points
indentfirst	Used per default to indent the first paragraph of each article.
markdown	Required only if using markdown input modes.

TODO: package version requirements.

Optional Requirements

JUnit-5.8.1 (for testing)

The Java programme comes with a reasonably complete set of unit tests written under the JUnit framework. The framework is not required to compile or run the programme, but is fairly essential to further develop the application.

The Programme

The programme itself can be downloaded from:-

`https://github.com/robertjlee/newspaper`

3 Article Requirements

The aim of this package is to collect discrete inputs and to present them in a professional-standard newspaper layout. Each input must have a single file that the programme understands, although it may input other files arbitrarily.

All inputs must be provided in `UTF-8` text encoding, one per file, with special *headers* provided at the start of the file, as described later in this manual.

The programme currently understands the following formats:-

LaTeX

The input will be typeset one or more times to measure its length, and then once in the final output. While any LaTeX package (other than class files) can be used, some require running the document multiple times to obtain the final output. These are unlikely to work well, unless the overall *length* of the article is unchanged.

Plain Text

Documents in plain text are supported. You should use a blank line to separate paragraphs; any special spacing in the input will be lost. Most other characters behave as you'd expect, although you can add extra dashes (-) in the input to produce longer dashes in the output (`1--3 dashes --- like this, or-else`; 1–3 dashes — like this, or-else). You can also use backticks (`) and apostrophies (´) to produce quotation marks (``` ``like this´´ ``` for "like this").

Markdown

There is some support for markdown, which requires the `lualatex` executable to be used, with the `--enable-shell` option. **Caution:** You should examine all input files carefully when using this configuration, other than plain text: it enables arbitrary code to be run on your computer, both when running the programme and when compiling the newspaper.

4 Glossery of Terms

Some of these terms are common in newsprint typesetting, others are specific to this algorithm. An understanding of how these terms will make the manual easier to follow.

Article

A collection of prose forming a cohesive story, which can be typeset in a number of ways.

Article Fragment

For a paste-up article, this is an element consisting of a cut-out section of the article placed on the page.

Alley

Whitespace - or anything else - placed between columns or elements on the page.

Element

The term is used here for an article fragment, title, or other insert set onto the page.

Footers

Any information provided at the end of an article, such as a *"continued on page…"* indicatior, or footnotes.

Headers

Any information provided at the start of an article or fragment, such as headlines, sub-headings, authors and so on. If an article breaks across pages, the *"…continued from page"* indicator is also considered to be a header.

Headline

A title of an article, set in a larger, bold font. The reader can scan the page for headlines to determine which articles they want to read.

Head-span Article

An article, set onto the page in a rectangle, with a headline spanning the entire article.

Input

A general term for an article, input or other file read by this programme to produce the output.

Input directory

A filesystem directory (or folder) containing one or more inputs. There may be one or more input directories.

Insert

An element added to the page with fixed dimensions. Typically, this is a graphic image or title, but the algorithm considers anything with a fixed-size to be an insert, including a head-span article where the number of columns is known.

Newsprint

The ink used to print the newspaper. The term is used here for anything typeset in the style of a newspaper, especially a portion of a newspaper page.

Output directory

A filesystem directory (or folder) into which the typeset version of the newspaper is placed.

Overflow

When typesetting a page, any elements (especially article fragments) that do not fit on a page are known as the overflow. Overflow is always typeset at a later place than it would normally be expected to.

Paste-up

A method of creating newsprint by setting an entire article in a single, long column, then splitting that column across the page. Distinguished by having the headline inside the column. Originally, this was done by literally cutting and pasting to create a master proof.

5 Filesystem Layout

This is an implementation of an algorithm to take *inputs* from files supplied by the user, and to produce an output layout. The implementation outputs a LaTeX 2_ε source file that references back to the supplied inputs, and can then be typeset directly to produce a rendered document. By default, the reference programme will also perform this typesetting.

The user is responsible for supplying the files required as inputs. The user must create one or more *input directories* on the filesystem, which will contain the content in roughly the order of typesetting. The user must also create an *output directory*, which should have the same parent directory as the first input direcotry.

This arrangement allows a lot of flexibility. For example: different sub-editors could supply different directories of inputs for typesetting, which can then be chained together; or you could supply one directory for each page of the newspaper, satisfied that any overflow from one page will still move to the next page as required.

The output from the programme must also go somewhere. It's not convenient to write the output to the same directory as the input, because of the way that LaTeX uses multiple temporary files during its layout — at minimum, an `.aux` and a `.log` file — which may affect subsequent runs, and can clutter the input directory. So, to make sure we can always get back to a "clean slate" to recreate the newspaper again, we always place the output in a directory of its own. The user can choose the name of this directory, but if they don't, a new directory named simply `out`, will be created.

6 Settings File

The settings file is named `settings.properties` and is formatted as a Java language Properties file, described in full in the Java SDK[1].

Four our purposes, each file contains a number of lines. Blank lines, and lines starting with a hash/pound sign (`#`) are ignored. Settings are given one per line, with the setting name followed by an equals sign (=) or colon (:), followed by the setting value. The backslash (\) escapes the following character, so a single backslash in a property value will be encoded as two backslashes in the property file.

When the application is run, each source directory is checked for a settings file. It is strongly recommended that the filename be in all lowercase, for portability. Settings files must be plaed directly in the source directory; subdirectories are not checked. If the same setting key is found in more than one settings file, the value from the later file is used, based on the order in which the source directories are given to the programme.

[1] `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Properties.html#load(java.io.Reader)`

Each setting has a default value, which may sometimes be calculated based on other settings (in which case, changing the other setting will change the default value).

Where a setting is a length, it should be specified as a floating-point number, followed directly by a unit indicator. 0 lengths do not require a unit. Units are not case-sensitive. Supported units are:

`in`	inches (1in = $\frac{1}{36}$ International Yard)
`inch`	inches
`inches`	inches
`pt`	TEX points (1pt = 0.35145980mm)
`mm`	millimeter
`cm`	centimeter
`dm`	decimeter
`m`	meter

Note that the typesetting units `em` and `ex` are not currently supported, and calculations cannot be supplied.

A note on paper sizes

In essence, a newspaper is a method of providing a large amount of accessible text onto a single page. For this reason, newspapers have often been printed onto larger size paper sheets.

Choosing a paper stock depends on your use-case, the size of paper you can obtain, and what your printing equipment can support. You may configure any paper size within the limits of LATEX (up to 2267.05408in for both `pageWidth` and `pageHeight`), but if you intend to print it, you will need to ensure you have the available paper and suitable printing equipment. There must be sufficient space for at least one column and the page must be larger than your largest fixed-size insert — including margins.

In most of the world, home printing uses paper sizes of the finished "A" series defined by ISO 216 (and incorporated by ANSI into ASME Y14.1M). The "B" series of the same standard supplies larger paper commonly used in commercial printing; it is available in larger sizes and is mainly used for printing "A"-series paper with bleed margins.

In North America, the ANSI standards are formalised in ASME Y14.1; the ARCH system is not a standard, but provides useful paper sizes preferred by architects.

The following table provides various dimensions of paper from around the world:-

Common Paper Sizes				Generic Newspaper Sizes		
	Code	`pageWidth`	`pageHeight`	**Code**	`pageWidth`	`pageHeight`
ISO 216	A3	297mm	420mm	**Tabloid**		
ISO 216	A4	297mm	210mm	CanadaT	260mm	368mm
ISO 216	B2	500mm	707mm	NorwayT	280mm	400mm
ISO 216	B3	353mm	500mm	BritainT	280mm	430mm
ANSI	Letter	11in	8.5in	**Berliner**		
ANSI	Legal	14in	8.5in	Midi	315mm	470mm
ARCH	A	9in	12in	**Broadsheet**		
ARCH	B	12in	18in	USA	381mm	578mm
ARCH	C	18in	24in	Norwegian	400mm	570mm
ARCH	D	24in	36in	South Africa	410mm	578mm
ARCH	E	36in	48in	(A2) Australia/	420mm	594mm
ARCH	E1	30in	42in	New Zealand		
ARCH	E3	27in	39in	Britain	375mm	597mm

A note on Fonts and Font Selection

Choosing fonts is always difficult, particularly where a high level of readability is required. Newspapers should be quick to skim and easy to read in detail, in spite of often very small text.

Generally, most documents use only a single typeface, in several variants (like bold text for titles). LaTeX has notoriously fiddly font handling, because it's designed around this use-case: it's possible to change fonts arbitrarily throughout a document, but it's far more common to simply switch to a variant — like using *italic* for emphasis — temporarily.

For a newspaper, there are generally three distinct typefaces in use: the main title is typically a bold, gothic style; headlines use a large serif font that's easily picked out on the page; and the article text uses a small, narrow-spaced serif font that guides the eye along the page. (There have been few published studies on the effectiveness of serif fonts on printed newspapers; however, when viewed on a backlit electronic display, sans-serif fonts are more readable: consider your audience).

A font is a physical container of engraved metal letters used in a printing press; it contains the complete alphabet, including ligatures and punctuation, for a consistent size, scale and form of the text. Electronic printing has allowed many thousands of fonts to be represented, and the term "font" has come to mean the specific shapes of the symbols once held in a single physical font. Notably, *10pt Times* and *11pt Times* are different *fonts*, even though they are the same *typeface* (Times).

In LaTeX, each font has a number of properties:-

Encoding

This controls how TeX's internal codes map to each glyph in the font.

Family

The typeface, or group of related typefaces

Series

Weight and width of the font

Shape

Upright, italic, slanted (oblique), small caps, outline, …

Size

The point size of the font. As a rule of thumb, the letter "M" will be this tall when measured in typesetting points — but the exact size is set by the font author.

Spacing

LaTeX actually considers this part of the font size; it tells TeX how far apart to set the baselines between each line of text within a paragraph. By default, this program leaves *2pt* between each line, but this can be varied when selecting the font depending on visual cues like the length and frequency of descenders (parts of text extending below the baseline, like in the letters "p" and "q"). The aim should be to balance the whitespace around the text and a clear distinction between lines, against the constant pressure to increase the volume of text on the page.

To use a font with the *LaTeX 2ε Font Selection System* (`FSS`), it must first be installed on your system; this typically happens when LaTeX is installed, but there is a (somewhat non-trivial) process to install your own. This process produces a file with a ".fd" extension for each combination of encoding and family, and only by inspection of this file can you determine the available values of series and shape.

If using *XeLaTeX*, or *LuaLaTeX* you can simply use any font installed on your operating system, by setting the forllowing options in a settings.properties file. See *The Fontspec package*[2] for further details:-

```
preambleFont0=\usepackage{fontspec}
preambleFont1=\setmainfont{Full name of font}
```

If not using XeLaTeX or LuaLaTeX, the default encoding (`TU`) may not be available. For most english-language text, you are recommended to choose the `T1` encoding, which is the orginal "standard" TeX encoding, and widely supported. The LaTeX project has a full description of the various font encodings[3]: the trick is to find encodings that are supported both by the font, and also by the LaTeX platform you are using: otherwise TeX will fail to find the characters and replace them with, typically, *Computer Modern Roman*. To switch the default encoding to `T1`, add this to your `settings.properties` file:-

```
defaultFontEncoding=T1
```

Having changed the main font family to Times, a new problem emerges: there is no suitably narrow monospaced font in the Times family. Yet this document supports Markdown, and monospaced fonts are typically seen in many Markdown documents found online, especially when discussing computers and data files. The default monospaced font (*Computer Modern Teletype*) has too wide a spacing for use in narrow columns, while also being visually

[2] `https://ctan.org/pkg/fontspec`

[3] `https://www.latex-project.org/help/documentation/encguide.pdf`

out of kilter with *Times Roman*. It is beyond the scope of this project to develop a new *Times Teletype* font, so instead the new default is *Latin Modern Mono Light Condensed* (*Latin Modern Mono* is the family; *Light Condensed* is the weight). This is chosen because in it a universally available high-quality font, ideally suited for narrow columns, and because no font that is more visually suited to *Times Roman* currently exists. You can switch the default teletype font using the following (TODO).

Having carefully selected your fonts, you should create a test paper, perhaps using the demo input files, and then check the LaTeX output log file carefully for warnings: it's possible for fonts to be missing expected glyphs, or even entire shapes or series.

One further warning: The *Mode=Plain* input expects the font you are using to provide characters for backslash (\), and braces ({}) at the standard unicode character points. Not all fonts installed in TeX do this, and so some choices may result in unexpected characters in the output; if you change the default document font to one which does not support these characters — inclruding `cmr`, the default font for TeX— then these characters may be omitted in the output, or different characters displayed instead, when they appear in a *Mode=Plain* input.

Further reading on fonts is available at the LaTeX Font Guide[4], and documentation on many pre-installed fonts is found at the LaTeX Font Catalogue[5].

6.1 Setting: `version`

- Type: symantic version number[6]
- Default: 1.0.0

This setting will be used by future releases of this programme. Where there are changes that break the visual output of the newspaper, setting this version number in your `settings.properties` to the release number of the programme, will make a best-effert to produce the output of the previous version.

This does not apply where a bugfix corrects clearly-erroneous output in a previous release; you are recommended to keep your PDF for posterity rathar than regenerating the `.tex` file each time an old PDF is required.

6.2 Setting: `allowTexFileOverwrite`

- Type: "true" or "false", any case
- Default: false

Normally, the programme will abort if the output file — typically `newspaper.tex` — already exists, to ensure that you can't accidentally overwrite the last edition's newspaper. Set this to "true" if you are going to run the programme repeatedly.

Always take care to back up your final edition newspapers regardless of this setting.

6.3 Setting: `pageSize`

- Type: code in above table
- Default: 650mm × 750mm

The `pageSize` simply sets both `pageWidth` and `pageHeight` based on the above table. For Australia/New Zealand, use code "A2".

This setting changes the default width and height of the page, and is overridden by `pageWidth` and/or `pageHeight`. Note that this is the size of a *page*; the *paper* size should normally be twice this, to allow four pages per leaf.

[4] `https://www.latex-project.org/help/documentation/fntguide.pdf`
[5] `https://tug.org/FontCatalogue/allfonts.html`
[6] `https://semver.org/`

Note that *A4* pages are printed in landscape by default, as headers can appear cramped on paper much narrower than around 300mm.

6.4 Setting: `pageWidth`

- Type: length
- Default: 650mm, or from `pageSize`

The width of a page, including all margins.

The default page size is a for typical broadsheet newspaper; see `pageSize`.

6.5 Setting: `pageHeight`

- Type: length
- Default: 750mm, or from `pageSize`

The height of a page, including all margins.

6.6 Setting: `columnWidth`

- Type: length
- Default: 1.5in

The width of a newsprint column, not including any alleys set to its sides.

6.7 Setting: `columnHeightRatioOfPage`

- Type: number with decimal places, dimensionless units
- Default: 0.9

The ratio of the height of the page to the height of a column. Ignored if the `columnHeight` is set. Must be greater than 0 and no greater than 1. Sets the top and bottom margins of the page.

6.8 Setting: `columnHeight`

- Type: length
- Default: The height of a column. If not specified, it is calculated based on `pageHeight` and `columnHeightRatioOfPage`

6.9 Setting: `alleyWidth`

- Type: length
- Default: 0.125in

Space between columns, including any dividing lines.

6.10 Setting: `alleyHeight`

- Type: 0.125in
- Default: length

When elements are set above each other, this is the amount of space to leave between them, including any dividing lines.

6.11 Setting: `alleyThickWidth`

- Type: 0.0125in
- Default: length

The width of a vertical line to place to the sides of newspaper columns. Set to 0 to disable vertical dividing lines.

6.12 Setting: `alleyThickHeight`

- Type: length
- Default: 0.0125in

The height of a horizontal line to place between elements that are placed above each other. Set to 0 to disable horizontal dividing lines.

6.13 Setting: `minSideMargins`

- Type: length
- Default: 0.125in

The minimum amount of whitespace to set at the sides of each page. Even if this is set to 0, there will almost always some space at the sides, because the alrogithm will only fit entire columns onto the page, with fixed-size alleys between them, and may put fewer columns onto a page than will fit in order to reduce the chance of orphan columns on the last page.

This is the guaranteed page border, totalled over left and right margins.

6.14 Setting: `defaultFontSize`

- Type: TEX points, typically 10, 11 or 12
- Default: 10

This setting controls the point size of the default document, in points.

The TEX distribution provides *class options* (primarily spacing defaults) for 10, 11, and 12 points. Where these values are not used, the nearest file will be loaded, and some sizes adjusted by scaling. For these other sizes, you may need to provide a `defaultFontSizeClo` file to obtain visually perfect output.

6.15 Setting: `defaultFontSizeClo`

- Type: String
- Default: *no value*

This advanced setting allows you to change the *class options* for the current document, and it will be loaded by the `fontsize` package. You should supply a file named *nameSIZE*`.clo` — where *SIZE* is the value of the `defaultFontSize` property, and *name* is the value of this property. The *.clo* file will then be used to set up the various size options for the default font.

For more information, see the FontSize documentation[7]

6.16 Setting: `defaultFontEncoding`

- Type: LATEX font encoding name
- Default: TU

This is the font encoding name used when no specific encoding is listed. Common choices are `T1` — for good compatibility, especially in English, when using pdflatex — `TU` for Unicode fonts supported by LuaLATEXand XeLATEX— and `TS1`, the "Cork" encoding, for most fonts originating from the `Metafont` project. `CJK` covers Chinese, Japanese, and Korean characters, while `X2` covers Cryillic, and `LGR` covers Greek.

6.17 Setting: `defaultFontFamily`

- Type: LATEX 2_ε font family code
- Default: ptm

This is the code for an installed typeface on your LATEX installation. The default is `ptm`, a Times Roman clone. Change this with care, as Times is ubiquitous for newspapers.

Common values are `cmr` for *Computer Modern Roman*, `cmrs` for *Computer Modern Roman Sans*, or `phv` for a Helvetica clone. This can also be a LATEX macro defined in the preamble that expands to the correct name.

Ignored if `defaultFontFamilyFromHeaders` is changed to `true`.

[7] `https://mirror.ox.ac.uk/sites/ctan.org/macros/latex/contrib/fontsize/fontsize.pdf`

6.18 Setting: `defaultFontFamilyFromHeaders`

- Type: *true* or *false*, case-insensitive
- Default: false

When false, this has no effect.

When true, this disables the attempt to change the value of the default typeface after loading the preamble, disabling the `defaultFontFamily`, `defaultFontSeries`, `defaultTeletypeFamily` and `defaultTeletypeSeries` settings. This allows you to import a package in the preamble section that changes the default typefaces. Otherwise, the typeface will be set to the TEX default, *Computer Modern Roman*, which is not well-suited to narrow columns.

6.19 Setting: `defaultFontSeries`

- Type: LATEX font series code
- Default: m

This is the series code for the default font. It produces a line in the output document `\DeclareFontSeriesDefault`

rm

`{md}{m}` — where the final value is the font series to be used by the default, medium-density roman font used in the document.

6.20 Setting: `defaultTeletypeFamily`

- Type: LATEX font family name
- Default: lmtt

This is the default family code for the default *teletype* (monospaced) font. While teletype fonts are rarely used in newspapers except for special effects, no good matching font currently exists, so this uses the *Latin Modern Teletype* font, which supports many languages and the ultra-condensed spacing.

6.21 Setting: `defaultTeletypeSeries`

- Type: LATEX font series code
- Default: lc

This setting selects the series (variant) of the teletype (monospaced) font. the *Light condensed* variant — supported by *Latin Modern Teletype* — is used by default, as this font (designed for margin notes) works well in narrow columns. It is, however, a lightweight font and stands out against the Times Roman text; this may or may not be desirable depending on usage.

6.22 Setting: `inputFilter`

- Type: Comma-separated strings
- Default: `.tex, .md, .markdown, .txt, .text`

This setting simply defines which files in the input will be checked for a *Type* header. This has no bearing on the type of content in that file; for that, use the *Mode* header.

Any file that matches one of these values will be considered for inclusion; so by default, you can write inputs as: `.tex`; `.md` or `.markdown`; and `.txt` or .text. It is suggested these be used with TEX, markdown, or plain text, respectively.

It is not possible to include a comma in the filename extension.

There is no wildcard character, whitespace before or after a comma is ignored, and blank extensions are ignored.

6.23 Setting: `head…`

- Type: Font definition
- Default: *18pt bold*

These settings set the font used to typeset the article headlines. They can be overridden for each article by changing the corresponding `Font…` headers. Note that if `headCommand` is used, it can only be overridden by the

`FontCommand` header; otherwise, each header overrides the corresponding setting.

The font may be configured using the following options:-

headEncoding

The LaTeX encoding code for the text (up to 3 upper-case letters, followed by optional digits)

headFamily

The LaTeX internal name of the font family (typeface). This can be a command, such as `\rmdefault` for the document default font family, or an internal name like `cmr` for Computer Modern Roman.

headSeries

The font series code for the font (usually "m" for medium or "b" for bold.

headShape

The shape code for the font (usually "n" for upright.

headSize

The size of the font to use, in LaTeX points.

headSpacing

The spacing of lines in this font; by default, *headSize* + 2pt. This may be a LaTeX rubber length, but only makes a difference if multiple lines of text are included.

headCommand

Some packages provide convenient user commands to switch fonts. This header, if specified, will override the other *head...* headers, and use this command instead.

6.24 Setting: `markdown`

- Type: LaTeX header code
- Default: `\usepackage`

$$smartEllipses, fancyLists$$

`{markdown}`

The programme supports accepting input in markdown format using the header `%#Mode=markdown`. When such a header is found, this setting is automatically added as a line in the document preamble.

To successfully compile a markdown article, you will need to set either `latex=lualatex` or `latex=xelatex`. Otherwise, you may need to add `--shell-escape` to the `latexCmdLine` setting; without a version of TeX built on a *Lua* engine, the markdown package needs to call a supporting programme from LaTeX. **Caution: This could be a security risk if mixing `markdown` and `LaTeX` article types in the same document!** it's better to use `lualatex` if in doubt.

The complete list of options are listed in the Markdown package documentation[8]. A quick reference of the more useful options follows:-

blankBeforeBlockquote

Chevrons (>) not following a blank line will not start a blockquote

blankBeforeHeading

A heading must follow a blank line.

blankBeforeList

A list must follow a blank line.

breakableBlockquotes=false

Ignore blank lines in the middle of a blockquote

definitionLists

After declaring a term, start a new line with a colon (:) to give its definition.

fancyLists

Change the presentation of lists.

fencedCode

Enable fenced-code blocks. An open fence is a line containing three backticks, and an optional format name, and preceeds a block of code. A closing fence line contains three

[8] `https://mirror.apps.cam.ac.uk/pub/tex-archive/macros/generic/markdown/markdown.html#options`

backticks, and follows the block of code. The code will be displayed as is, perhaps highlighted depending on the format name.

hashEnumerators

Use a hash sign (in US english, a pound sign) followed by a fullstop (`#.`) to introduce each item of a numbered list.

html

Allow inline static HTML within the markdown article.

hybrid

Allow inline LaTeX within the markdown article. This can be useful for special effects, but changes the meaning of backslash (\) and braces ({}) and introduces the possibility of breaking the newspaper compilation.

linkAttributes

A syntax for including inline images, like this: `![image](foo.jpg){#id .class width=30 height=20px}`

lineblocks

Successive lines starting with a bar (|) will be considered as a block, and leading spaces preserved. Useful for addresses.

mark

Use two equals signs either side of text to highlight it by changing the background colour.

notes

Allows a syntax similar to footnotes, but presented inline in the text.

shiftHeadings=*number*

The given *number* is added to the heading level; positive numbers make headings smaller.

smartEllipses

With this option, any occurance of `...` in the input will be properly formatted as an ellipsis character (…); without it, you will see three dots with poor spacing (...).

strikeThrough

With this option, text surrounded with double tildes (~) will be overtyped with a horizontal line.

subscripts

Text surrounded with single tildes (~) will be shown subscript; e.g. `H~2~O` becomes H_2O.

subscripts

Text surrounded with single carets (^) will be shown subscript; e.g. `e=mc^2^` becomes e=mc^2 — see also `texMathDollars`.

texComments

A percent sign (%), and any text on a line after it, will be ignored. Note that lines starting with a % will always be ignored due to how the Markdown input is included.

texMathDollars

When text is surrounded by dollar signs, it will be typeset in LaTeX math-mode. e.g. `$e=mc^2$` is typeset as $e = mc^2$. Double dollars use block math mode.

tightLists=false

By default, lists lists written as one line per item are rendered with less spacing than lists containing paragraphs of text. This option disables this tighter-grouping and adds more whitespace.

underscores=false

By default, underscores declare emphasis like asterisks; this option disables this. Recommended with the `hybrid` option.

6.25 Setting: `continuedOnPageText`

- Type: LaTeX
- Default: \hfill\textit{\small Continued on page %s\dots}

This LaTeX text is displayed after each article fragment where the article does not fit completely on the page. If the string contains "%s", then it will be replaced with the page number of the continuation page.

6.26 Setting: `continuedFromPageText`

- Type: LaTeX
- Default: \textit{\small Continued from page %s\dots}\hfill

This LaTeX text is displayed before each article fragment where the previous article fragment was displayed on a previous page. If the string contains "%s", then it will be replaced with the page number of the continuation page.

6.27 Setting: `jobname`

- Type: filename without suffix
- Default: newspaper

Primarily, this sets the name of the output files written by the alrogithm. Usually, this also sets the name of the `PDF` and `.log` files produced by LaTeX, although that can be overwritten by setting the TeX variable `\jobname` or passing the `-jobname` argument through the `latexCmdLine` setting.

6.28 Setting: `logFile`

- Type: filename or path relative to output directory
- Default: layout.log

When creating the output newspaper, a log file is usually created to supply detailed information about processing. Depending on how you work, this may be more or less convenient that the information supplied by the console. This file can be anazylsed to determine why something was not processed as expected.

By default, this file is placed in the output directory and named `layout.log`. This setting simply changes the filename.

See `logFileLevel` for how to configure the contents of this file.

If this file is not created, the console output should be consulted.

6.29 Setting: `logFileLevel`

- Type: One of: `silent`, `quiet`, `elements`, `algorithm` or `dump_all`
- Default: algorithm

This setting determines what is logged to the log file. If `silent`, no log file is created. `Quiet` will ensure the file only contains error messages, *i.e.* the reason the output `.tex` file could not be produced. `Elements` will provide statistics for each inputs file (this could perhaps be used to determine how many column-inches of content have been supplied by each author). `Algorithm` will trace the algorithm as it considers how to lay out each page, which can be useful for editors looking to better understand and tweak the way pages are laid out. `Dump_all` can produce a large amount of content, and includes the text of all intermediate `.tex` files used to measure the length of articles.

Each level implies the output of previous levels as well.

NB: The `Algorithm` level includes an estimate of how much more content is required to completely fill the number of pages needed to set the provided input. This is an overestimate, as it does not allow for the column-inches used by "continued on" labels, "continued from" labels, or alleys between articles.

6.30 Setting: `stdOutLevel`

- Type: One of: `silent`, `quiet`, `elements`, `algorithm` or `dump_all`
- Default: elements

This setting is identical to `logFileLevel`, but controls the output produced on the console — technically, on the *standard output stream*.

Please note that errors parsing the settings file will always appear on the console via the *standard error* stream; see `stdErrLevel`.

6.31 Setting: `stdErrLevel`

- Type: One of: `silent`, `quiet`, `elements`, `algorithm` or `dump_all`
- Default: silent

This advanced setting is of interest to users wishing to automate the process. The setting is identical to `stdOutLevel`, except that it controls the *standard error stream*, which remains displayed on the console even if the standard output is redirected.

`Silent` here means to include only the most critical of errors, such as parsing the settings files: such messages will always be sent to *standard error*.

Any messages sent to *standard error* will not be sent to *standard output*, regardless of the `stdOutLevel` flag. For example, if `stdErrLevel` is set to `quiet`, and `stdOutLevel` to `elements`, then all errors will be sent to *standard error* and statistics will be sent to *standard output*.

6.32 Setting: `latex`

- Type: *OS-specific* path-name
- Default: lualatex

This is the executable used to size articles and lay out the final document. For security, it is best to set this to the full path to the LaTeX executable on your system. Using a `pdflatex` or `lualatex` executable is recommended, as the original `dvi` format output may cause issues with some fonts, including the default Times clone.

6.33 Setting: `latexCmdLine`

- Type: LaTeX command-line options
- Default: `--interaction=nonstopmode`

This advanced setting changes how the programme invokes the LaTeX command, both for the final output, and for the measuring of element sizes. See the LaTeX documentation for full details.

For example, you could add `-jobname=edition1` to change the output filenames to `edition1.tex` and `edition1.pdf` and the LaTeX log filename to `edition1.log`.

6.34 Setting: `inputWithoutCopy`

- Type: "true" or "false", any case
- Default: false

In most cases, the program will copy the text from the input files into the output `.tex` file. If the input files are each stand-alone (*i.e.* use to external graphics or text files themselves), this produces a stand-alone document.

Some users may prefer to set `inputWithoutCopy=true`, to link the input files to the output file instead. This may allow a faster turnaround for small edits, as the input files can be tweaked and the output file run thourgh LaTeX without invoking the programme again. (Note, however, that the column-inches for each article would not be re-calculated, possibly resulting in articles overflowing or underfilling columns).

This mode is currently ignored by `mode=plain` articles, which do not support the copying mode.

6.35 Setting: `tolerance`

- Type: dimensionless integer, between 0 and 10000 inclusive
- Default: 500

This is an advanced option, and sets the value of the TeX setting `\tolerance`. This must be no less than 0, and values over 10000 will have no effect. Increasing this number will allow extra spacing between words to produce a better layout of text within a column.

6.36 Setting: `out`

- Type: out
- Default: pathname

This is the path-name used to resolve the output directory, where the output and logs of the programme are placed. The path is resolved on the default filesystem and path: for most systems, this means it is relative to the working directory.

6.37 Setting: `pretolerance`

- Type: dimensionless integer, between -1 and 10000 inclusive
- Default: -1

This is an advanced option, and sets the value of the TEX setting `\pretolerance`. This should be less than `tolerance` in most cases. If not set to -1, it instructs TEX to first attempt to break paragraphs into lines without hyphenation; if this produces an adequate result, the full line-break alrogithm is bypassed. This has the effect that fewer hyphenated line-breaks will appear in the final document, without much loss in quality.

Generally, this setting is not useful in newspapers, because the text is too narrow to typeset effectively without hyphenation. However, if you have an especially narrow font, or many especially wide columns, then this setting may compile slightly faster with some fewer hyphenation points.

6.38 Setting: `emegencystretch`

- Type: LATEX function code
- Default: `\emergencystretch = 4pt \multiply #1`

This is an advanced option, and controls how the value of the TEX setting `\emergencystretch` should be set.

When setting text in narrow columns, there is much less chance of a natural line-break position (space or hyphenation point) occurring just before the end of a line. TEX responds to this by producing overfull lines, with warnings in the output file, and expects the user to resolve these warnings by changing the text appropriately so that the line-break positions are moved. In full-width text, this happens rarely, and the addition of an extra comma or filler word is usually all that is needed to resolve the issue, guaranteeing high-quality text output. However, this becomes very onerous for newspaper text, due to the much-increased number of lines, and the lower chance of a good line-break combination.

In 1989, to avoid this problem, TEX introduced the `\emergencystretch` register (in version 3.0). This register tells TEXhow much extra whitespace is permissible, per line, before it counts the line as being bad and tries another combination of breaks. Higher settings mean fewer overfull lines for the user to manually fix, at the cost of extra horizontal whitespace.

As a rule of thumb, `\emergencystretch` can be larger when there are more average spaces per line in the input. This setting holds the definition of a LATEX function, taking as parameters: the number of columns (#1), and the current width of the text on a column (*\hsize* or #2). It should produce no output when expanded, but set the value of the *\emergencystretch* register instead. This syntax is compatible with the `multicols` environment, and means that an appopriate setting can be used even when the column width changes (see the `innerCols` setting).

An comprehensive article on choosing an optimal value is given in the TEX Users' Group magazine, *TUGboat*, volume 38, from p.65[9]

[9] `https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf`

It should be noted that the *multicols* environment — principly used for 2–3 columns on A4 or letter paper, perhaps more in landscape mode — uses the value of *4pt per column*, and tends to produce very visually spaced output. The authors warn that this may not be optimal, and it seems to have been intruduced when the number of columns was limited to 10, so may not even have been tested with dozens of columns like on a newspaper.

The current default of `0.1 textbackslash hsize` — one tenth of the column with, excluding alleys — seems to work well with even large numbers of columns.

6.39 Setting: `preamble…`

- Type: LaTeX preamble code
- Default: *(see description)*

Each setting beginning with "preamble" is added to the document preamble.

Note that the settings are sorted lexicographically, so `preamble10` will run before `preamble9`, because `1` comes before `9` in the dictionary — regardless of the order of lines in the `settings.properties` files.

It is also possible to add preamble lines in the individual header files.

While writing a LaTeX preamble is relatively advanced, it provides almost complete control over your document. A number of defaults are provided, which should not normally be changed. There are:-

```
preamble!00head=\usepackage{indentfirst}
```

to indent the first line of each paragraph;

```
preamble!01head=\usepackage[british]{babel}
```

to allow hyphenation within words;

```
preamble!02head=\usepackage[utf8]{inputenc}
```

to switch the input encoding to `UTF-8`;

```
preamble!03head=\usepackage{newtxmath,nextxtext}
```

to switch the default font to use the new Times package (which must come after `babel`; and

```
preamble!04head=\usepackage{csquotes}
```

, which does nothing by default, but allows you to add "\MakeInnerQuote{}"to allow the user to type a single " character in most places where a quotation mark would appear (change "MakeInnerQuote" to "MakeOuterQuote" for American-style quotation marks, where single-quotes are used around quotations). At time of writing, there seem to be compatibility issues with head-span articles in markdown when using \MakeInnerQuote.

The use of underscores (`!`) in the propery keys indicates that these are default settings; any new defaults in future versions are likely to be named `preamble!04head` or similar, so while you may use these names to override the default settings, you are recommended to choose your own distinct convention for any additional lines, to avoid conflicts in future.

Inputs

The programme takes inputs corresponding to the logical inputs accepted by an editor: newspaper articles, illustrations, title lines and so on. Each of these inputs corresponds to a file in the input directory, using a defined extension (*i.e.* the end of the filename), with a special header to configure how the document is to be laid out.

Each input produces one or more *elements* — which may be *article fragments* or fixed-size *inserts* — that the algorithm will place on the page.

Each file contains the metadata to configure how it's to be treated by the layout algorithm in a number of *header lines*. Header lines change the behaviour of the layout programme in producing the typeset newspaper, but not the behaviour of how TEX typesets the final document. So each header line will begin with `%#` as TEX will ignore any input line starting with a `%` character, so any line starting with a `%` but not `%#` can still be used for comments and will be ignored.

Only the first 50 lines of the document may contain header lines; header lines after this will be ignored.

Headers consist of a number of key-value pairs. The form is to use the Java `.properties` standard, but with the addition of `%#` at the start of every line. Thus, headers form a list of key-value pairs, similar to the `settings.properties` input file(s).

For example, the following complete input file produces a *paste-up* article consisting of nonsense pseudolatin:-

```
%# Type: article
%# Head: Lorum Ipsum
%# Preamble: \usepackage{lipsum}
\lipsum
```

The allowable header values are:-

6.1 Header: `Type`

- Type: case-insensitive
- Default: *None*

One of the following:-

Article

A paste-up article

Fixed

A fixed-size insert. *Width* and *Height* are required.

HeadSpan

An article. *Cols* is required. The headline will span the number of columns given by *Cols*, and the length of the article will be as long as required to balance the content among this number of columns. The article will be set as a unit and not broken up.

This currently uses the `multicol` package, which is a part of the standard LATEX distribution.

Title

Intended for the headline at the start of the paper; this is a full-width insert displayed in headline font, that can be customized through various headers.

Other input types may be added in future.

A document that does not declare a `Type` header will not produce an input in the document. This allows support for complex inputs consisting of multiple files.

6.2 Header: `Asset…`

- Type: filename
- Default: *None*

This is an advanced option that may be useful in fixing the issue of articles overrunning their allocated space, if you include many files together to form your newspaper article.

Normally, each article is copied to a temporary directory and compiled, with `TEXMFPATH` and `\input@files` set to the input folder(s). In most cases, this is sufficient to accurately render the document. However, when using third-party packages to include files — particularly through Markdown, but sometimes with other packages as well — the file may be missing, and the article will not have the size it needs in the final document.

In this case, you can list your assets in the Header file, with keys starting "Asset", and values equal to the filenames relative to the input folder. The assets will be copied to the temporary directory and used to calculate a more accurate article size.

6.3 Header: `Mode`

- Type: case-insensitive
- Default: LaTeX

One of the following:-

LaTeX

This is the default. The text of the article will be used directly in the output TEX file. Characters such as `\, { and }` have special meaning, and failure to follow LATEX conventions will break the document — but any number of special effects are possible. See the `markdown` setting for further details.

Plain

Input is plain `utf-8` text, with paragraphs separated with blank lines. Hyphens (-), backticks (`) and apostrophies (´) will be treated the same as in TEX. The text will be formatted as a regular article, but special effects (like *italic text*) are not possible. Please see also the warning in the note on Fonts and Font Selection.

Markdown

Input is in a simple `markdown` encoding. This lets your authors use basic decorations, such as adding asterisks around text to make it ***bold***, or underscores for _*italic*_. Markdown is a widely adopted de-facto standard format.

6.4 Header: `Cols`

- Type: Natural number
- Default: *Required* for *Type=Headspan*

With *Type=HeadSpan*, this gives the width of the insert, in columns. The article will exactly fill the stated number of columns.

This uses the "multicols" environment, which has a built-in limit of 20 internal columns. You can split the document across more than 20 columns of the newspaper, but fewer columns than expected may appear within the box.

6.5 Header: `InnerCols`

- Type: Natural number, at least 2
- Default: Defaults to *Cols*; used only with *Type=Headspan*

With *Type=HeadSpan*, this gives the number of columns to divide the article into to typeset; unlike *Cols* which determines the size of the article on the newspaper grid. By default, if there are up to 20 columns, the article will match the rest of the newspaper grid.

This setting can be used to set this article in columns of a different width than the rest of the paper.

If this setting is more than 20 columns, then the `multicol` package will report a warning, and 20 columns will be used. This is a limitation of `multicol`.

6.6 Header: `ColumnHint`

- Type: Natural number
- Default: *None*

This option allows some influence over the horizontal postiion of *Type=Fixed* and *Type=HeadSpan*. It specifies the preferred starting column for the insert. Note that the layout of such fixed-size elements on the page does allow for some movement after the initial placement, so this is not guaranteed to produce the expected result in the output.

The left-most column is column "1", and columns count upwards to the right.

6.7 Header: `RuleWidth`

- Type: Length
- Default: *alleyThickWidth* setting

May be used with *Type=HeadSpan* to change the width of the rules between the columns within the head-span article. Set to 0 to remove lines within the article itself.

6.8 Header: `Width`

- Type: Length
- Default: *Required* for *Insert*

Used with *Type=Insert* to set the width of the fixed-size insert. Extra whitespace may be added to the left and right of the insert, to allow for partial columns.

6.9 Header: `Height`

- Type: Length
- Default: *Required for Insert*

Used with *Type=Insert* to set the height of the fixed-size insert.

6.10 Header: `Head`

- Type: Text
- Default: *No headline*

Not used with *Title* or *Insert* types. Sets a headline to be set on top of an *Article* or *HeadSpan*. *Article* headlines may be set inside the column, while *HeadSpan* will span the article. The headline font can be chaged for each article by `Head…`, which operates the same as the `head…` setting.

6.11 Header: `Preamble`

- Type: LaTeX header code
- Default: *None*

This applies to all input types, and it itstructs the programme to add an output line to the preamble of the final document. This can be used to load packages required by the article.

*Caution:*LaTeX provides few ways to stop one article from interfering with another, and by its nature, the preamble can affect the entire document. Therefore, it is recommended to limit using *Preamble* to importing macro packages, and to put anything else into the settings file.

6.12 Header: `Page`

- Type: Integer
- Default: *None*

Sometimes it's useful to put certain content on certain pages: headline news is reserved for the first page; sports and comics go to the back. This is an editorial decision affecting layout, and can be affected by this option.

When this is omitted, or set to zero, it has no effect: articles will be set in the order they appear in the file listings, one directory at a time. When set to a non-zero value, it specifies the minimum page number on which

the article will appear, either from the front or end page of the newspaper.

Where the value is a (positive) whole number, it gives the minimum "natural" page number; i.e. couning the first page as "1", the second as "2", and so on.

Where this is set to a negative integer, this also specifies the minimum page number, but using a different count: the "natural" last page is "-1", the penultimate page is "-2", and so on, counting down and backwards. NB: The number of pages here is based on the simple area of articles to be typeset. Extra pages may be created if this setting is used to create gaps: *e.g.* if all inputs have either "Page: 2" or "Page: -1", then the first page may be blank; also, in some rare cases, the final article may overflow the last page, and an additional page will be created *after* the articles are placed. This additional pages would be numbered as page "0", "1", and so on, as they cannot be counted until after the document is complete.

There is currently no support for double-truck/"centerfold" pages; however, it is expected that these would be counted as a single page.

Note that this option may leave gaps in the newspaper, and that it can only make an article appear later than it would otherwise.

6.13 Header: `Date`

- Type: LaTeX text
- Default: *localised date of Java programme run*

Used with *Type=Title* to add an extra line, left-aligned, just above the title inself. This might declare the date of the newspaper edition.

6.14 Header: `Edition`

- Type: LaTeX text
- Default: *empty*

Used with *Type=Title* to add an extra line, right-aligned, just above the title inself. This might declare the price of the newspaper.

6.15 Header: `Price`

- Type: LaTeX text
- Default: *empty*

Used with *Type=Title* to add an extra line, right-aligned, just below the title inself. This might declare the price of the newspaper.

6.16 Header: `TagLine`

- Type: LaTeX text
- Default: *empty*

Used with *Type=Title* to add an extra line, left-aligned, just below the title inself. This might declare the motto of the newspaper.

6.17 Header: `Font…`

- Type: Font definition
- Default: *54pt Almedera Bold*

Used with *Type=Title* to configure the font for the title text itself. See the `head…` setting (in the settings section) for a description of the available headers.

6.18 Header: `Head…`

- Type: Font definition
- Default: *none*

If any of the `Head…` font headers are specified, the headline for the article will be set in the supplied font. This allows different styles for each headline, in the manner of a tabloid. Each overrides the corresponding `head…` setting.

If `headSize` is overridden by `FontSize`, the default value of `FontSpacing` becomes *FontSize* + 2pt.

While modern broadsheets do not typically use different headline fonts per article, historical newspapers certainly did: using larger headlines to draw the eye to more important stories.

It is suggested to consider *Headseries=bx* on wider *Type=Headspan* articles, where this font definition is provided.

6.19 Header: `Date…`

- Type: Font definition
- Default: *8pt Times*

Used with *Type=Title* to configure the font for the date line itself. This works identically to `Font…`, but replacing the prefix "Font" with "Date".

6.20 Header: `Price…`

- Type: Font definition
- Default: *8pt Times*

Used with *Type=Title* to configure the font for the price line itself. This works identically to `Font…`, but replacing the prefix "Font" with "Price".

6.21 Header: `Edition…`

- Type: Font definition
- Default: *8pt Times*

Used with *Type=Title* to configure the font for the edition line itself. This works identically to `Font…`, but replacing the prefix "Font" with "Edition".

6.22 Header: `TagLine…`

- Type: Font definition
- Default: *8pt Times*

Used with *Type=Title* to configure the font for the tag line itself. This works identically to `Font…`, but replacing the prefix "Font" with "TagLine".

6.23 Header: `LeftBox…`

- Type: LaTeX text
- Default: *empty*

Used with *Type=Title* to add the given content centered in a box to the left of the title. Such boxes are typically used to display logos or advertise pages of potential interest to the reader.

Additionally, the following related headers can configure the box itself:-

LeftBoxWidth

Sets the width of the box; default *1in*.

LeftBoxHeight

Sets the height of the box; default is *LeftBoxWidth* for a square box.

LeftBoxRaise

Sets the vertical offset of the box's content. Default is given by $\frac{\mathit{LeftBoxWidth}}{2} - 0.05\text{in} = 0.45\text{in}$, assuming the default LeftBoxWidth.

LeftBox...

You can also set the font for the left box by specifying `LeftBoxCommand` or `LeftBoxFamily` etc; see `Font…`. Note that the default is the document main font.

6.24 Header: `RightBox…`

- Type: LaTeX text
- Default: *empty*

As *LeftBox…* except that this applies to a box shown to the right of the title. The lengths for each box must be configured separately.

6.25 Header: `RaiseLength`

- Type: Length
- Default: *10pt*

The top line of the title (date and edition) is not allocated any space, as to do so would make the newspaper title appear too low on the page.

This setting is the distance that the top line should be raised up, into the page borders. The value of `HeadSpacing` is typically a good starting point for tweaking this value.

Command-Line

The programme is distributed as a `.jar` file: an archive of executable Java code. To run the programme, you will need to execute the `java.exe` executable:

```
java.exe -jar layout.jar srcdir1/ srcdir2/
```

Here, `srcdir1` and `srcdir2` are paths to the *source directories* containing your newspaper inputs.

This will show a console window, showing some information about the newspaper layout, and any errors that are raised by LaTeX.

All of this information shown on the console is also available in a log file inside the output directory, provided that the output directory setting can be read, and points to a usable directory. You may prefer to disable the console window; in which case, use the windowless version `java.exe` instead:-

```
javaw.exe -jar layout.jar srcdir1/ srcdir2/
```

Note that, without a console window, errors relating to discovery of the output directory or creation of the log file, will not be shown.