PANDOC(1) Pandoc User Manuals

John MacFarlane

January 8, 2008

Contents

1 NAME

pandoc - general markup converter

2 SYNOPSIS

pandoc [*options*] [*input-file*]…

pandoc -s Pandoc_1.md –pdf-engine=xelatex -o Pandoc_1.pdf

pandoc -s Pandoc_1.md –pdf-engine=lualatex -o Pandoc_1.pdf

pandoc -s Manual_Pandoc.md -o Manual_Pandoc_A5.pdf –from markdown –template="template-pandoc.tex" -V papersize="A5"

pandoc -s Pandoc_1.md –variable="vmargin:1cm"

–variable="hmargin:1cm" -V papersize="A5" –pdf-engine=lualatex -o Pandoc_0.pdf

3 PDF with numbered sections and a custom LaTeX header:

pandoc -N –template=template-pandoc.tex -V papersize="A4" -V mainfont="Linux Libertine G" -V sansfont="DejaVu Sans" -V monofont="DejaVu Sans Mono" -V fontsize=12pt -V version=2.0 Pandoc_1.md –pdf-engine=lualatex –toc -o Pandoc_1_lualatex_A4.pdf

pandoc -N –template=template-pandoc.tex -V papersize="A5" -V mainfont="Linux Libertine G" -V sansfont="DejaVu Sans" -V monofont="DejaVu Sans Mono" -V fontsize=12pt -V version=2.0 Pandoc_1.md –pdf-engine=lualatex –toc -o Pandoc_1_lualatex_A5.pdf

pandoc -N –template=template-pandoc.tex –variable papersize="A4" –variable mainfont="Linux Libertine G" –variable sansfont="DejaVu Sans" –variable monofont="DejaVu Sans Mono" –variable fontsize=12pt –variable version=2.0 Pandoc_1.md –pdf-engine=xelatex

–toc -o Pandoc_1_xelatex_A4.pdf

pandoc -N –template=template-pandoc.tex –variable papersize="A5" –variable mainfont="Linux Libertine G" –variable sansfont="DejaVu Sans" –variable monofont="DejaVu Sans Mono" –variable fontsize=12pt –variable version=2.0 Pandoc_1.md –pdf-engine=xelatex –toc -o Pandoc_1_xelatex_A5.pdf

4 DESCRIPTION

Pandoc converts files from one markup format to another. It can read markdown and (subsets of) reStructuredText, HTML, and LaTeX, and it can write plain text, markdown, reStructuredText, HTML, LaTeX, ConTeXt, Texinfo, groff man, MediaWiki markup, RTF, OpenDocument XML, ODT, DocBook XML, EPUB, and Slidy or S5 HTML slide shows.

If no *input-file* is specified, input is read from *stdin*. Otherwise, the *input-files* are concatenated (with a blank line between each) and used as input. Output goes to *stdout* by default (though output to *stdout* is disabled for the `odt` and `epub` output formats). For output to a file, use the `-o` option:

```
pandoc -o output.html input.txt
```

Instead of a file, an absolute URI may be given. In this case pandoc will fetch the content using HTTP:

```
pandoc -f html -t markdown http://www.fsf.org
```

The input and output formats may be specified using command-line options (see **OPTIONS**, below, for details). If these formats are not specified explicitly, Pandoc will attempt to determine them from the extensions of the input and output filenames. If input comes from *stdin* or from a file with an unknown extension, the input is assumed to be markdown. If no output filename is specified using the `-o` option, or if a filename is specified but its extension is unknown, the output will default to HTML. Thus, for example,

```
pandoc -o chap1.tex chap1.txt
```

converts *chap1.txt* from markdown to LaTeX. And

```
pandoc README
```

converts *README* from markdown to HTML.

Pandoc's version of markdown is an extended variant of standard markdown: the differences are described in the *README* file in the user documentation. If standard markdown syntax is desired, the `--strict` option may be used.

Pandoc uses the UTF-8 character encoding for both input and output. If your local character encoding is not UTF-8, you should pipe input and output through `iconv`:

```
iconv -t utf-8 input.txt | pandoc | iconv -
f utf-8
```

5 OPTIONS

-f *FORMAT*, -r *FORMAT*, --from=*FORMAT*, --read=*FORMAT*

Specify input format. *FORMAT* can be `native` (native Haskell), `markdown` (markdown or plain text), `rst` (reStructuredText), `html` (HTML), or `latex` (LaTeX). If `+lhs` is appended to `markdown`, `rst`, or `latex`, the input will be treated as literate Haskell source.

-t *FORMAT*, -w *FORMAT*, --to=*FORMAT*, --write=*FORMAT*

Specify output format. *FORMAT* can be `native` (native Haskell), `plain` (plain text), `markdown` (markdown), `rst` (reStructuredText), `html` (HTML), `latex` (LaTeX), `context` (ConTeXt), `man` (groff man), `mediawiki` (MediaWiki markup), `texinfo` (GNU Texinfo), `docbook` (DocBook XML), `opendocument` (OpenDocument XML), `odt` (OpenOffice text document), `epub` (EPUB book), `slidy` (Slidy HTML and javascript slide show), `s5` (S5 HTML and javascript slide show), or `rtf` (rich text format). Note that `odt` and `epub` output will not be directed to *stdout*; an output filename must be specified using the `-o/--output` option. If `+lhs` is appended to `markdown`, `rst`, `latex`, or `html`, the output will be rendered as literate Haskell source.

-s, --standalone Produce output with an appropriate header and footer (e.g. a standalone HTML, LaTeX, or RTF file, not a fragment).

-o *FILE*, --output=*FILE* Write output to *FILE* instead of *stdout*. If *FILE* is '`-`', output will go to *stdout*.

-p, --preserve-tabs Preserve tabs instead of converting them to spaces.

--tab-stop=*TABSTOP* Specify tab stop (default is 4).

--strict Use strict markdown syntax, with no extensions or variants.

--reference-links Use reference-style links, rather than inline links, in writing markdown or reStructuredText.

-R, --parse-raw Parse untranslatable HTML codes and LaTeX environments as raw HTML or LaTeX, instead of ignoring them.

-S, --smart Use smart quotes, dashes, and ellipses. (This option is significant only when the input format is `markdown`. It is selected automatically when the output format is `latex` or `context`.)

-m*URL*, --latexmathml=*URL* Use LaTeXMathML to display embedded TeX math in HTML output. To insert a link to a local copy of the `LaTeXMathML.js` script, provide a *URL*. If no *URL* is provided, the contents of the script will be inserted directly into the HTML header.

--mathml Convert TeX math to MathML. In standalone mode, a small javascript will be inserted that allows the MathML to be viewed on some browsers.

--jsmath=*URL* Use jsMath to display embedded TeX math in HTML output. The *URL* should point to the jsMath load script; if provided, it will be linked to in the header of standalone HTML documents.

--gladtex Enclose TeX math in `<eq>` tags in HTML output. These can then be processed by gladTeX to produce links to images of the typeset formulas.

--mimetex=*URL* Render TeX math using the mimeTeX CGI script. If *URL* is not specified, it is assumed that the script is at `/cgi-bin/mimetex.cgi`.

--webtex=*URL* Render TeX math using an external script. The formula will be concatenated with the URL provided. If *URL* is not specified, the Google Chart API will be used.

-i, --incremental Make list items in Slidy or S5 display incrementally (one by one).

--offline Include all the CSS and javascript needed for a Slidy or S5 slide show in the output, so that the slide show will work even when no internet connection is available.

--xetex Create LaTeX outut suitable for processing by XeTeX.

-N, --number-sections Number section headings in LaTeX, ConTeXt, or HTML output. (Default is not to number them.)

--section-divs Wrap sections in `<div>` tags, and attach identifiers to the enclosing `<div>` rather than the header itself.

--no-wrap Disable text wrapping in output. (Default is to wrap text.)

--sanitize-html Sanitizes HTML (in markdown or HTML input) using a whitelist. Unsafe tags are replaced by HTML comments; unsafe attributes are omitted. URIs in links and images are also checked against a whitelist of URI schemes.

--email-obfuscation=*none|javascript|references*
Specify a method for obfuscating `mailto:` links in HTML documents. *none* leaves `mailto:` links as they are. *javascript* obfuscates them using javascript. *references* obfuscates them by printing their letters as decimal or hexadecimal character

references. If `--strict` is specified, *references* is used regardless of the presence of this option.

--id-prefix=*string* Specify a prefix to be added to all automatically generated identifiers in HTML output. This is useful for preventing duplicate identifiers when generating fragments to be included in other pages.

--indented-code-classes=*classes* Specify classes to use for indented code blocks–for example, `perl,numberLines` or `haskell`. Multiple classes may be separated by spaces or commas.

--toc, --table-of-contents Include an automatically generated table of contents (HTML, markdown, RTF) or an instruction to create one (LaTeX, reStructuredText). This option has no effect on man, DocBook, Slidy, or S5 output.

--base-header-level=*LEVEL* Specify the base level for headers (defaults to 1).

--template=*FILE* Use *FILE* as a custom template for the generated document. Implies `-s`. See TEMPLATES below for a description of template syntax. If this option is not used, a default template appropriate for the output format will be used. See also `-D/--print-default-template`.

-V KEY=VAL, --variable=*KEY:VAL* Set the template variable KEY to the value VAL when rendering the document in standalone mode. This is only useful when the `--template` option is used to specify a custom template, since pandoc automatically sets the variables used in the default templates.

-c *CSS*, --css=*CSS* Link to a CSS style sheet. *CSS* is the pathname of the style sheet.

-H *FILE*, --include-in-header=*FILE* Include contents of *FILE* at the end of the header. Implies `-s`.

-B *FILE*, --include-before-body=*FILE* Include contents of *FILE* at the beginning of the document body. Implies `-s`.

-A *FILE*, --include-after-body=*FILE* Include contents of *FILE* at the end of the document body. Implies `-s`.

-C *FILE*, --custom-header=*FILE* Use contents of *FILE* as the document header. *Note: This option is depre-*

cated. Users should transition to using `--template` instead.

--reference-odt=*filename* Use the specified file as a style reference in producing an ODT. For best results, the reference ODT should be a modified version of an ODT produced using pandoc. The contents of the reference ODT are ignored, but its stylesheets are used in the new ODT. If no reference ODT is specified on the command line, pandoc will look for a file `reference.odt` in the user data directory (see `--data-dir`). If this is not found either, sensible defaults will be used.

--epub-stylesheet=*filename* Use the specified CSS file to style the EPUB. If no stylesheet is specified, pandoc will look for a file `epub.css` in the user data directory (see `--data-dir`, below). If it is not found there, sensible defaults will be used.

--epub-metadata=*filename* Look in the specified XML file for metadata for the EPUB. The file should contain a series of Dublin Core elements (http://dublincore.org/documents/dces/), for example:

```
<dc:rights>Creative Commons</dc:rights>
```

```
<dc:language>es-AR</dc:language>
```

By default, pandoc will include the following metadata elements: `<dc:title>` (from the document title), `<dc:creator>` (from the document authors), `<dc:language>` (from the locale), and `<dc:identifier id="BookId">` (a randomly generated UUID). Any of these may be overridden by elements in the metadata file.

-D *FORMAT*, --print-default-template=*FORMAT*
Print the default template for an output *FORMAT*. (See `-t` for a list of possible *FORMAT*s.)

-T *STRING*, --title-prefix=*STRING* Specify *STRING* as a prefix to the HTML window title.

--data-dir=*DIRECTORY* Specify the user data directory to search for pandoc data files. If this option is not specified, the default user data directory will be used:

```
$HOME/.pandoc
```

in unix and

```
C:\Documents And Settings\USERNAME\Application Data\pand
```

in Windows. A `reference.odt`, `epub.css`, `templates` directory, or `s5` directory placed in this directory will override pandoc's normal defaults.

--dump-args Print information about command-line arguments to *stdout*, then exit. The first line of output contains the name of the output file specified with the `-o` option, or '`-`' (for *stdout*) if no output file was specified. The remaining lines contain the command-line arguments, one per line, in the order they appear. These do not include regular Pandoc options and their arguments, but do include any options appearing after a '`--`' separator at the end of the line. This option is intended primarily for use in wrapper scripts.

--ignore-args Ignore command-line arguments (for use in wrapper scripts). Regular Pandoc options are not ignored. Thus, for example,

```
pandoc --ignore-args -o foo.html -s foo.txt -- -
e latin1
```

is equivalent to

```
pandoc -o foo.html -s
```

-v, --version Print version.

-h, --help Show usage message.

6 TEMPLATES

When the `-s/--standalone` option is used, pandoc uses a template to add header and footer material that is needed for a self-standing document. To see the default template that is used, just type

```
pandoc --print-default-template=FORMAT
```

where `FORMAT` is the name of the output format. A custom template can be specified using the `--template` option. You can also override the system default templates for a given output format `FORMAT` by putting a file `templates/FORMAT.template` in the user data directory (see `--data-dir`, below).

Templates may contain *variables*. Variable names are sequences of alphanumerics, `-`, and `_`, starting with a letter. A variable name surrounded by `$` signs will be replaced by its value. For example, the string `$title$` in

```
<title>$title$</title>
```

will be replaced by the document title.

To write a literal `$` in a template, use `$$`.

Some variables are set automatically by pandoc. These vary somewhat depending on the output format, but include:

`legacy-header` contents specified by `-C/--custom-header` `header-includes`
contents specified by `-H/--include-in-header` (may have multiple values) `toc`
non-null value if `--toc/--table-of-contents` was specified `include-before`
contents specified by `-B/--include-before-body` (may have multiple values) `include-after`
contents specified by `-A/--include-after-body` (may have multiple values) `body`
body of document `title`
title of document, as specified in title block `author`
author of document, as specified in title block (may have multiple values) `date`
date of document, as specified in title block

Variables may be set at the command line using the `-V/--variable` option. This allows users to include custom variables in their templates.

Templates may contain conditionals. The syntax is as follows:

```
$if(variable)$
X
$else$
Y
$endif$
```

This will include `X` in the template if `variable` has a non-null value; otherwise it will include `Y`. `X` and `Y` are placeholders for any valid template text, and may include interpolated variables or other conditionals. The `$else$` section may be omitted.

When variables can have multiple values (for example, `author` in a multi-author document), you can use the `$for$` keyword:

```
$for(author)$
<meta name="author" content="$author$" />
$endfor$
```

You can optionally specify a separator to be used between consecutive items:

```
$for(author)$$author$$sep$, $endfor$
```

7 SEE ALSO

`markdown2pdf` (1). The *README* file distributed with Pandoc contains full documentation.

The Pandoc source code and all documentation may be downloaded from http://johnmacfarlane.net/pandoc/.