book-mk Documentation

Phil Ruffwind

Contents

Chapter 1

Introduction

`book-mk` is a set of scripts for building books from Markdown source files. Currently, HTML and PDF are supported.

`book-mk` is not meant to be installed as a package. Instead, it is intended to be embedded inside a subdirectory of your book's source tree. You can use Git submodules for this but it is not required and won't be explained here.

1.1 Requirements

First of all, you need a Unix-like system, because the build system relies heavily on makefiles. Additionally, the following tools are needed:

- GNU Make 4.2.*
- Pandoc 1.18.*.*
- pandoc-citeproc 0.10.*.* (if you don't use citations, you can just replace this with `sh -c cat`)
- pandoc-crossref 0.2.*.* (if you don't use cross-references, you can just replace this with `sh -c cat`)
- Rust 1.17.*
- For PDF output:
  - LaTeX 2e (including latexmk 4.52c)
  - If you have any SVG images:

        * Inkscape 0.92.*

The version numbers are advisory: things will probably still work on something a bit older or newer.

The book generator also relies on a *patched* version of mdBook (see `mdbook.patch` in the source tree), but this will be automatically downloaded and installed to `./.local/bin/mdbook` by the build system. The user shouldn't have to worry about this detail except on airgapped systems.

Chapter 2

Usage

2.1 Minimal example

Start by creating an empty directory. This will serve as the source tree of your book.

Download the `book-mk` source code and extract its contents into a subdirectory `./book-mk` so that "`./book-mk/book.mk`" points to the makefile. Other than things like `./book-mk/README.md` or `./book-mk/doc.md`, most of the files under `./book-mk` are essential and should not be removed.

Create `./Makefile` with the following contents:

```
metadata=-M title="My first book" \
         -M author="Your Name"

tool_dir=book-mk
include $(tool_dir)/book.mk
```

This is your top-level makefile and is the entry point of the build system. (The `metadata` and `tool_dir` variables are special, so don't rename them!)

Next, create `./src/SUMMARY.md` with your table of contents:

```
[Preface](preface.md)
- [Chapter 1](chapter-1.md)
- [Chapter 2](chapter-2.md)
  - [Section 2.1](section-2-1.md)
[Epilogue](epilogue.md)
```

Finally, create a file `./src/preface.md` with the following contents:

```
Hello world!
```

Don't worry about the other chapters – the build system will automatically create them if they're missing.

2.2 Building the book

Simply run:

```
make target/html target/pdf
```

- The HTML version is at `target/html/index.html`.
- The PDF version is at `target/pdf/book.pdf`. There'll be some auxiliary files created in `target/pdf` as well, but none of them are essential. They can be useful for debugging problems though.

The makefile also provides the following commands (phony targets):

- `clean`: Remove the `target` directory. This does *not* remove the `.local` directory, however, because building those can often take some time.
- `deploy-gh-pages`: Upload the HTML and PDF books to GitHub pages using the `origin` remote of the current Git repository.

Chapter 3

Customization

3.1 Makefile

The main point of customization is your `Makefile`. The following Make variables are supported:

- `tool_dir` (required): The directory that contains the tools.
- `metadata`: This can contain either:
  - `-M title="<title>"`
  - `-M author="<author>"` (you can repeat this flag multiple times for multiple authors)
- `pandoc_args`: Arguments given to all Pandoc invocations.
- `latex_documentclass`: LaTeX document class (defaults to `book`).
- `latex_pandoc_args`: Arguments given to the Pandoc invocation that generates LaTeX.
- `html_pandoc_args`: Arguments given to all Pandoc invocations that generate HTML.

Pandoc offers a lot of ways to customize your output, many of which can have dramatic effects on the end result. If for some reason one of the flags interacts poorly with the `book-mk` system, please report a bug so we can either document it as a limitation or (better yet) find a workaround!

For more advanced customizations, you'd want to read Sec. 4.

3.2 Environment variables

When building, the following variables are used to determine where the respective executables are found. This can be useful if you have, e.g. `pandoc` installed at an unconventional location that is not on `PATH`.

- `CARGO`
- `INKSCAPE`
- `LATEXMK`
- `PANDOC`
- `PANDOC_CITEPROC`
- `PANDOC_CROSSREF`

By default, they simply default to their unqualified executable names (i.e. `PANDOC` defaults to just `pandoc`, etc).

Chapter 4

Internals

```
src -------> target/stage ------> target/<format>
    frontend                backend
```

- Frontend (Sec. 4.1): format-independent processing through Pandoc
- Backend (Secs. 4.2.1, 4.2.2): format-specific rendering through Pandoc and mdBook/LaTeX

4.1 Frontend

```
src/*.md

    |
    | prepend-heading
    v

[md]

    |
    | pandoc (frontend)
    v

[json]
```

```
    |
    | preprocessor
    v

target/stage/src/*.json
```

Each Markdown file is transformed individually into an output-independent JSON file using Pandoc. The use of JSON ensures that all the details are preserved correctly, including citations (which would get lost if HTML was used for intermediate files). The preprocessor is not yet implemented.

The transformation of each file is independent of each other; there is no state maintained.

Still, even though the transformations are totally independent, the makefile has to know which files need to be transformed. This is what `target/stage/src/SUMMARY.mk` stores: a list of the book items in the correct ordering as a makefile variable `$(item_names)`. It is obtained via `.local/bin/get-book-items`.

4.2 Backend

Using the files in the staging directory (`target/stage`), we use mdBook to parse the structure of the book items (chapters, sections, etc) in `SUMMARY.md` and render the output files. This is handled by the following custom Rust tools:

- `html-mdbook-toml`, which translates the Makefile variable `$(metadata)` into `target/stage/html/book.toml` for mdBook.
- `html-merge`: combines the input files into an amalgamation so that `pandoc-crossref` works correctly (also appends the bibliography if present)
- `html-fix-links`: make the links in an amalgamation work correctly after splitting
- `html-split`: splits the amalgamation back into individual pages for `mdbook`

- `latex-merge`, reads `SUMMARY.md` and combines the various `target/stage/src/*.json` files into a single JSON file `target/pdf/book.json` to be consumed by Pandoc for PDF generation.

4.2.1 HTML Output

```
target/stage/src/*.json

        |
        | html-merge
        v

[json]

        |
        | pandoc-crossref
        | +
        | pandoc-citeproc
        | +
        | html-fix-links
        | +
        | pandoc (HTML)
        v

[html]

        |
        | html-split
        v

target/stage/html/src/*.htm

        |
        | mdbook
        v
```

`target/html`

The JSON files are translated into `.htm` using Pandoc. The choice of `.htm` instead of `.html` is mainly to work around an mdBook quirk.

In principle, this one should be straightforward: just call `mdbook build <dir>`. In practice, we have to use a patched version of mdBook because we want to render Markdown using Pandoc rather than the default pulldown-cmark.

4.2.1.1 The mdBook patch

We use a patched version of mdBook that renders directly through HTML rather than through Markdown. This is because we use want to use Pandoc to preprocess the input files and "standard" Markdown is not very expressive.

Originally, the plan was to preprocess with Pandoc and save them as CommonMark so that mdBook could parse them. However, the problem is that (a) Pandoc does not support a lossless conversion to CommonMark (b) pulldown-cmark does not adhere to CommonMark very well (e.g. it does not preserve whitespace within `<pre>` elements as the spec demands).

Therefore, we simply disable the Markdown processing altogether and use HTML. To avoid duplicating a large chunk of code from mdBook, so we opted to just patch the original code instead.

4.2.1.2 Using `.htm` instead of `.html`

After rendering, mdBook copies all files except `.md` files into the output directory. If we named the input files `.html` they would immediately overwrite the files we just generated! Moreover, using `.htm` makes it easy to delete the input files from the output directory afterward.

4.2.1.3 Keeping `target/html` tidy

Part of the reason to have `target/stage` is so that we can select precisely what files we want. mdBook is not very selective so it will copy everything in `src` into `dest` and then purge all the `.md` files.

4.2.2 PDF Output

```
target/stage/src/*.json

        |
        | latex-merge
        v

target/pdf/book{_head.tex,{_before,,_after}.json}

        |
        | pandoc-multiref
        | +
        | pandoc (LaTeX)
        v

target/pdf/book.tex

        |
        | latexmk
        v

target/pdf/book.pdf
```

`latex-merge` is responsible for combining the chapters together into a single JSON file, shifting the heading levels as necessary. Then `Pandoc` is invoked to generate the `.tex` file. Finally we run `latexmk` to produce the PDF.

The main reason for invoking `Pandoc` within `latexbook` is because there are auxiliary files (e.g. frontmatter file) that would otherwise get entangled with the Makefile.

4.2.2.1 Location of the `.bib` file

It's much easier to have the `.bib` file at the top-level. The reason is that we want it to work with both `pandoc-citeproc`, which treats it as a path relative to the top-level directory, and `natbib`, which treats it as a path relative to `target/pdf`. If we put it inside `src`, then `pandoc-citeproc` would complain, and mdbook would unwittingly copy the `.bib` file into `target/html`.

Example of a bibliographic citation: [Doe, 2000]

Bibliography

Jane Doe. *This is a test.* University Press, 2000.