Crowbook

Crowbook

Élisabeth Henry

Chapter 1

Crowbook

Render a markdown book in HTML, Epub or PDF.

Crowbook's purpose is to allow you to automatically generate multiple outputs formats from a book written in Markdown. Its main focus is novels, and the default settings should (hopefully) generate readables book with correct typography.

It is also possible to use `crowbook` for e.g. technical documentation, but it might not be the best tool for that.

Building and installing

Youl'll need to have the Rust compiler on your machine first; you can download and install it here. Once it is down:

```
$ cargo install crowbook
```

will download `crowbook` and install it.

Usage

The simplest command is:

```
$ crowbook <BOOK>
```

Where `BOOK` is a configuration file. Crowbook will then parse the config file and generate book in HTML, Epub, LaTeX, and/or PDF, according

to the setting in the configuration file. So if you clone this repository and run

```
$ crowbook book_example/config.book
```

(or `cargo run - book_example/config.book` if you don't want to install it), you'll generate the example book in various format. The HTML version should look like that.

Now, let's say you want to make your own book. Assuming you have a list of Markdown files, you can generate a template configuration file with the `-create` argument:

```
$ crowbook --create my.book chapter_*.md
```

This will generate a default `my.book` file, which you'll need to complete.

This configuration file contains some metadata, options, and lists the Markdown files. Here is a basic example:

```
author: Joan Doe
title: Some book
lang: en

output_html: some_book.html

+ chapter_1.md
+ chapter_2.md
+ chapter_3.md
+ ...
```

For more information see the configuration file page, or the whole book_example directory. (A (not necessarily up-to-date) rendered version is available in HTML here).

It is also possible to give additional parameters to `crowbook`; we have already seen `-create`, but if you want the full list, see the arguments page.

Current features

Output formats

Crowbook (to my knowledge) correctly supports HTML and EPUB (either version 2 or 3) as output formats: rendered files should pass

respectively the W3C validator and the IDPF EPUB validator for a wide range of (correctly Markdown formatted) input files. See the example book rendered in HTML and EPUB on github.io.

LaTeX output is more tricky: it should work reasonably well for novels (the primary target of Crowbook), but `pdflatex` might occasionally choke on some « weird » unicode character. Moreover, the rendering of code blocks is not satisfactory and images are not yet implemented. See the example book rendered in PDF on github.io to get an idea of the problems you might encounter.

ODT output is experimental at best. It might work if your inputs files only include very basic formatting (basically, headers, emphasis and bold), it will probably look ugly in the rest of the cases, and it might miserably fail in some. See the example book rendered in ODT on github.io if you want to hurt your eyes.

Input format

Crowbook uses pulldown-cmark and thus should support most of commonmark Markdown. There are, however, a few features that that are not supported:

- Tables and Footnotes are not yet implemented. Tables might be implemented soon, but footnotes are a bit more tricky.
- HTML in markdown is not implemented, and probably won't be, as the goal is to have books that can also be generated in PDF (and maybe ODT).

Maybe the most specific "feature" of Crowbook is that (by default, it can be deactivated) tries to "clean" the input files. By default this doesn't do much (except removing superfluous spaces), but if the book's language is set to french it tries to respect french typography, replacing spaces with non-breaking ones when it is appropriate (e.g. in french you are supposed to put a non-breaking space before '?', '!', ';' or ':'). This feature is relatively limited at the moment, but I might try to add more options and support for more languages.

Table of Contents

Crowbook currently only supports a table of contents for first-level headers (e.g. chapters). There is currently no option to integrate it in the document, so it is only generated in the EPUB format.

See also Bugs.

Acknowledgements

Besides the Rust compiler and standard library, Crowbook uses the following libraries:

- pulldown-cmark (for parsing markdown)
- mustache (for templating)
- clap (for parsing command line arguments)
- chrono (date and time library)
- uuid (to generate uuid)

While Crowbook directly doesn't use them, there was also inspiration from Pandoc and mdBook.

ChangeLog

See ChangeLog.

Library

While the main purpose of Crowbook is to be runned as a command line, the code is written as a library so if you want to build on it you can use it as such. The code is currently badly documented (and badly in a general manner), but you can look at the generated documentation here.

License

Chapter 2

The configuration file

If you want to use Crowbook for your book, this configuration file is all you'll have to add (assuming you'll already have the book in Markdown files; if you don't, you'll also have to write a book first, but that's besides the scope of this document).

The format is not very complicated. This is an example of it:

```
# metadata
author: Joan Doe
title: Some book
lang: en

output.html: some_book.html

# list of chapters
- preface.md
+ chapter_1.md
+ chapter_2.md
+ chapter_3.md
+ chapter_4.md
- epilogue.md
```

Basically, it is divided in two parts:

- a list of options, under the form `key: value`;
- a list of Markdown files.

Files starting with the # characters are comments and are discarded by Crowbook when parsing the files. Note that # must be at the *beginning* of the line, so e.g.:

```
author: John Smith # aka John Doe
```

will set the `author` key to `John Smith # aka John Doe`.

The list of files

There are various options to include a markdown file.

- `+ file_name.md` includes a numbered chapter.
- `- file_name.md` includes an unnumbered chapter.
- `! file_name.md` includes a chapter whose title won't be displayed (except in the toc for epub); this is useful for e.g. including a copyright at the beginning or the book, or for short stories where there is only one chapter.
- `42. file_name.md` specifies the number for a chapter.

So a typical usage might look like this:

```
! copyright.md
- preface.md
# We want first chapter to be Chapter 0 because we are programmers!
0. chapter_0.md
# Next chapters can be numbered automatically
+ chapter_1.md
+ chapter_3.md
...
```

There are two important things to note:

1. you must *not* use quotes around the file names
2. the path of these files are relative to the directory where your config file is, *not* to the directory where you are when running `crowbook`. E.g. you can run `crowbook books/my_trilogy/first_book/config.` without being in the book's directory.

Also note that you don't have to specify a title. This is because the title of the chapter is inferred from the Markdown document. To go back to our previous example:

```
+ chapter_1.md
```

does not specify a chapter title, because it will read it directly in `chapter_1.md`, e.g.:

```
The day I was born
==================

...
```

You should have one and only one level-one header (i.e. chapter title) in each markdown file.

If you have more than one, Crowbook won't get too angry at you and will just print a warning and treat it as another chapter (numbered according to the scheme specified for including the file). It will however mess the table of contents if Crowbook tries to generate one (e.g. for Epub).

It's also a problem if you do *not* have a level-1 header in a markdown file. If it is a numbered chapter Crowbook will still be able to infer a chapter name, but if it is not numbered Crowbook will fail to generate an Epub file.

*****

So, to sum it up. *please*: one file = one chapter, a chapter starts with a title, and this way this will work nice.

Crowbook options

The first part of the configuration file is dedicated to pass options to Crowbook. Each one is of the form `option: value`. Note that you don't have to put string in quotes, e.g.:

```
title: My title
```

If you *do* use quotes, Crowbook will actually put those quotes in the string, so basically don't do that.

Metadata

author

Quite obviously, the author of the book. Note that it's currently just a single string, so if you want to have multiple authors, you'll have to do something like:

```
author: Jane Doe, John Smith
```

default: `Anonymous`

title

The title of the book.
default: `Untitled`

lang

The language of the book, in a standard format. "en", "fr", and so on.
default: `en`

cover

The file name of a cover image for the book. Note that, here again, you must not use quotes:

```
cover: cover.png
```

default: `None`

subject

What your book is about: e.g. Programming, Science-Fiction...
default: `None`

description

A description of your book. Note that Crowbook does *not* support multi-line strings in configuration field, and it is a field where it might be a problem if you don't like very long lines.
default: `None`

Output options

These options specify which files to generate. You must at least set one of this option, or Crowbook won't do anything.

Recall that all file paths are relative to the directory where the config file is, not to the one where you run `crowbook`. So if you set

```
output.epub = foo.epub
```

and runs

```
$ crowbook some/dir/config.book
```

`foo.epub` will be generated in `some/dir`, not in your current directory.

Crowbook will try to generate each of the `output.xxx` files that are specified. That means that you'll have to set at least one of those if you want a call to

```
$ crowbook my.book
```

to generate anything. (It's still possible to generate a specific format, and only this one, by using the `-to` argument on the command line).

output.epub

The name of the epub file you want to generate.

default: `None`

Crowbook use the `zip` command to generate the epub, so it won't work if this command is not installed on your system.

output.html

The name of the HTML file you want to generate. Note that this HTML file is self-contained, it doesn't require e.g. CSS from other files.

default: `None`

output.tex

The name of the LaTeX file you want to generate.

default: `None`

output.pdf

The name of the PDF file you want to generate. Crowbook uses LaTeX to generate it, so it won't work if it isn't installed on your computer. (See `tex_command` to specify which flavour of LaTeX you want to use.)

default: `None`

output.odt

The name of the ODT (OpenOffice) file you want to generate. Beware: ODT support is still experimental.

default: `None`

As for Epub generation, Crowbook depends on the presence of the `zip` command to generate this file.

Generic options for rendering

numbering

A boolean that sets whether or not you want numbering. Setting it to `false` is equivalent to including all your chapters with `- my_chapter.md`. Note that even if it is set to `true`, numbering will be desactivated for chapters that are included with `- my_chapter.md`.

default:: `true`

numbering_template

A string will be used as chapter title. You can use `{{number}}` and `{{title}}` in this string, e.g.:

```
numbering_template: Chapter {{number}} {{title}}
```

Note that:

- this string isn't used for unnumbered chapters;
- this string isn't used for LaTeX, either.

default: `{{number}}. {{title}}`

autoclean

This option cleans a bit the input markdown. With the default implementation, it only removes consecutive spaces, which has not real impact (they are ignored anyway both by HTML viewers and by LaTeX).

However, if `lang` is set to `fr`, it also tries to add non-breaking spaces in front (or after) characters like '?', '!', ';' to respect french typography.

default: `true`

nb_char

This option allows you to specify the non breaking character used by the french cleaning method (see above). Probably not really something you need to modify.

default: ' ' (i.e. narrrow non-breaking space)

Options for HTML rendering

html.template

A file containing a (mustache) HTML template.

default: `None` (built-in template)

html.css

A file containing a stylesheet for the HTML file.

default: `None` (built-in)

Options for Epub rendering

epub.template

A file containing a (mustache) xhtml template for the files generated for each chapter.

default: `None` (built-in template)

epub.css

A file containing a stylesheet for the Epub file.

default: `None` (built-in)

epub_version

Sets the version for generated Epub, either 2 or 3.
default: `2`

Options for LaTeX / PDF rendering

tex.template

Sets the LaTeX template.
default: `None` (builtin)

tex.command

The command used to generate a PDF file.
default: `pdflatex`

tex.links_as_footnotes

Display the link in the document, using the `\footnote{}` command. This is useful if you want your readers to be able to see the URLs in a printed document.
default: `true`

Additional options

temp_dir

When it is generating epub or pdf files, Crowbook creates a temporary directory (which is then removed), named from a random uuid (so we can be pretty certain it's not gonna exist). This option specify where to create this directory. E.g., if you set:

```
temp_dir: /tmp
```

crowbook might create a temporary directory `/tmp/7fcbe41e-1676-46ba-b1a7-40c2fa`
By default, this temporary directory is created where the config file is.
default: `.`

Chapter 3

Arguments

Crowbook can takes a list of arguments:

```
Render a markdown book in Epub, PDF or HTML.

USAGE:
        crowbook [OPTIONS] <BOOK> [--] [ARGS]

OPTIONS:
        --autoclean <BOOL>    Try to clean input markdown [values:
        --create              Creates a new book file with existing
    -h, --help                Prints help information
        --numbering <BOOL>    Number chapters or not [values: true,
    -o, --output <FILE>       Specify output file
    -t, --to <FORMAT>         Generate specific format [values: epu
    -V, --version             Prints version information
    -v, --verbose             Activate verbose mode

ARGS:
    <BOOK>     A file containing the book configuration
    <FILES>...    Files to put in book when using --create

Command line options allow to override options defined in <BOOK> co
E.g., even if this file specifies 'verbose: false', calling 'crowbo
will activate verbose mode.

Note that Crowbook generates output files relatively to the directo
$ crowbook foo/bar.book --to pdf --output baz.pdf
```

`will thus generate baz.pdf in directory foo and not in current directory.`

The most important ones is obviously , i.e. the file configuration book. It is mandatory: if you don't pass it, `crowbook` will simply display this help message. In a normal use case this is the only argument you'll need to pass, and `crowbook` will generate the book in all formats specified in the configuration file.

It is, however, possible to pass more arguments to `crowbook`.

`-create`

Usage: `crowbook -create <BOOK> file_1.md file_2.md ...`

Creates a new book from a list of Markdown files. It will generate the file `BOOK` (or abort if it already exists) with all file names specified added as chapters.

Most other `crowbook` options have no impact when using `-create`, but if `-numbering` is set to `false`, the files will be included with `-` instead of `+` in the Markdown file.

Example

```
crowbook --create foo.book README.md ChangeLog.md LICENSE.md
```

will generate a file `foo.book` containing:

```
author: Your name
title: Your title
lang: en

# Uncomment and fill to generate files
# output.html: some_file.html
# output.epub: some_file.epub
# output.pdf: some_file.pdf

# Uncomment and fill to set cover image (for Epub)
# cover: some_cover.png

# List of chapters
+ README.md
+ ChangeLog.md
+ LICENSE.md
```

-verbose

usage: `crowbook -verbose <BOOK>`

If this flag is set, Crowbook will print some more messages.

-to

usage: `crowbook -to <FORMAT> <BOOK>`

(or `crowbook -t <FORMAT> <BOOK>`)

Generate only the specified format. `FORMAT` must be either `epub`, `pdf`, `html`, `odt` or `tex`.

If an output file for the format is not specified in the book configuration file, `crowbook` will fail to render PDF, ODT and Epub (whereas it will print HTML and Tex files on stdout). It is however possible to specify a file with the `-output` option.

Examples

```
crowbook --to html foo.book
```

will generate some HTML, and prints it either to the file specified by `output.html` in `foo.book`, or to stdout.

```
crowbook --to pdf --output foo.pdf foo.book
```

will (try to) generate a `foo.pdf` file,.

-output

usage: `crowbook -to <FORMAT> -output <FILE> <BOOK>`

(or `crowbook -t <FORMAT> -o <FILE> <BOOK>`)

Specifies an output file. Only valid when `-to` is used.

Note that Crowbook generates output files relatively to the directory where `BOOK` is:

```
$ crowbook foo/bar.book --to pdf --output baz.pdf
```

will thus generate `baz.pdf` in directory `foo` and not in current directory.

-numbering

usage: `crowbook -numbering <BOOL> <BOOK>`

Turns numbering on or off. Overrides the option set in `BOOK` configuration file.

Example

```
crowbook --numbering false foo.book
```

-autoclean

usage: `crowbook -autoclean <BOOL> <BOOK>`

Turns autoclean on or off. Overrides the option set in `BOOK` configuration file.