Crowbook

Crowbook

Élisabeth Henry

Contents

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.

1.1 Installing

There are two ways to get `crowbook`: either use a precompiled binary or build it yourself.

1.1.1 Binaries

See to download a precompiled binary for your architecture (currently: Linux, Windows and MacOSX). Just extract the archive and run `crowbook` (you might also want to copy the binary somewhere in your `PATH` for later usage).

1.1.2 Building

You'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 automatically download the latest `crowbook` release on and install it.

1.2 Usage

The simplest command is:

```
$ crowbook <BOOK>
```

Where `BOOK` is a configuration file. Crowbook will then parse this file and generate a book in HTML, Epub, LaTeX, and/or PDF, according to the settings in the configuration file. So if you clone this repository and run

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

you'll generate the example book in various formats. The HTML version should look like that.

To create a new 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.

1.3 Current features

1.3.1 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 a bit 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.

1.3.2 Input format

Crowbook uses pulldown-cmark and thus should support most of commonmark Markdown. Inline HTML, however, is not implemented, and probably won't be, as the goal is to have books that can also be generated in PDF (and maybe eventually ODT).

Maybe the most specific "feature" of Crowbook is that (by default, it can be deactivated) it 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.

See also Bugs.

1.4 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)

It also uses configuration files from [rust-everywhere] to use [Travis] and [Appveyor] to generate binaries for various platforms on each release.

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

1.5 ChangeLog

See ChangeLog.

1.6 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.

1.7 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, and it looks a bit like YAML. 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`.

2.1 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.

2.2 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.

It is possible to use multiline strings with either `>` or `|`, and then indenting the lines that are part of the string:

```
title: >
 A
 long
 title
author: Joan Doe
```

will set `title` to `"A long title"`, whereas

```
title: >
 A
 long
 title
author: Joan Doe
```

will set `title` to `"A\nlong\ntitle\n"` (replicating line returns).

This feature is useful for options like `description` who may take a long string.

2.2.1 Options summary

2.2.2 Metadata

- `author`
  - **type**: string
  - **default value**: `Anonymous`
  - The author of the book
- `title`
  - **type**: string
  - **default value**: `Untitled`
  - The title of the book
- `lang`
  - **type**: string
  - **default value**: `en`

  - The language of the book

- `subject`
  - **type**: string
  - **default value**: `not set`
  - Subject of the book (used for EPUB metadata)
- `description`
  - **type**: string
  - **default value**: `not set`
  - Description of the book (used for EPUB metadata)
- `cover`
  - **type**: string
  - **default value**: `not set`
  - File name of the cover of the book

2.2.3 Output options

- `output.epub`
  - **type**: string
  - **default value**: `not set`
  - Output file name for EPUB rendering
- `output.html`
  - **type**: string
  - **default value**: `not set`
  - Output file name for HTML rendering
- `output.tex`
  - **type**: string
  - **default value**: `not set`
  - Output file name for LaTeX rendering
- `output.pdf`

  - **type**: string
  - **default value**: `not set`
  - Output file name for PDF rendering

- `output.odt`
  - **type**: string
  - **default value**: `not set`
  - Output file name for ODT rendering

2.2.4 Misc options

- `numbering`
  - **type**: integer
  - **default value**: `1`
  - The maximum heading levels to number (0: no numbering, 1: only chapters, ..., 6: all)

- `autoclean`
  - **type**: boolean
  - **default value**: `true`
  - Toggles cleaning of input markdown (not used for LaTeX)

- `verbose`
  - **type**: boolean
  - **default value**: `false`
  - Toggle verbose mode

- `side_notes`
  - **type**: boolean
  - **default value**: `false`
  - Display footnotes as side notes in HTML/Epub

- `nb_char`
  - **type**: char
  - **default value**: `'  '`

  - The non-breaking character to use for autoclean when lang is set to fr

- `temp_dir`
  - **type**: string
  - **default value**: `.`
  - Path where to create a temporary directory

- `numbering_template`
  - **type**: string
  - **default value**: `{{number}}. {{title}}`
  - Format of numbered titles

2.2.5 HTML options

- `html.template`
  - **type**: string
  - **default value**: `not set`
  - Path of an HTML template

- `html.css`
  - **type**: string
  - **default value**: `not set`
  - Path of a stylesheet to use with HTML rendering

2.2.6 EPUB options

- `epub.version`
  - **type**: integer
  - **default value**: `2`
  - The EPUB version to generate

- `epub.css`
  - **type**: string
  - **default value**: `not set`

  - Path of a stylesheet to use with EPUB rendering

- `epub.template`
  - **type**: string
  - **default value**: `not set`
  - Path of an epub template for chapter

2.2.7 LaTeX options

- `tex.links_as_footnotes`
  - **type**: boolean
  - **default value**: `true`
  - If set to true, will add foontotes to URL of links in LaTeX/PDF output

- `tex.command`
  - **type**: string
  - **default value**: `pdflatex`
  - LaTeX flavour to use for generating PDF

- `tex.template`
  - **type**: string
  - **default value**: `not set`
  - Path of a LaTeX template file

2.2.8 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).

Note that some formats depend on some commands being installed on your system. Most notably, Crowbook depends on LaTeX (`pdflatex` by default, though you can specify the command to use with `tex.command`) to generate a PDF file, so PDF rendering won't work if it is not installed on your system. Crowbook also uses the `zip` command to generate the EPUB and ODT, files. want to use.)

2.2.9 Generic options for rendering

2.2.9.1 numbering

An integer that represents the maximum level of numbering for your book. E.g., `1` will only number chapters, while `2` will number chapters, sections, but not anything below that. `6` is the maximum level and turns numbering on for all headers.

default:: `1`

2.2.9.2 numbering_template

A string that will be used for chapter titles. 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 by LaTeX, either.

2.2.9.3 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.

2.2.9.4 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.

2.2.10 Additional options

2.2.10.1 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] [--] [ARGS]

OPTIONS:
        --create              Creates a new book with existing mark
    -h, --help                Prints help information
    -l, --list-options        Lists all possible option
        --list-options-md     List all options, formatted in Markdo
    -o, --output <FILE>       Specifies output file.
    -s, --set <KEY_VALUES>    Sets a list of book options
    -t, --to <FORMAT>         Generate specific format [values: epu
    -V, --version             Prints version information
    -v, --verbose             Activate verbose mode

ARGS:
    <BOOK>        File containing the book configuration.
    <FILES>...    Files to list in book when using --create
```

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

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.

The most important option obviously <BOOK>, i.e. the file configuration book. It is mandatory for most options: 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`.

3.1 `-create`

Usage: `crowbook [BOOK] -create file_1.md file_2.md ...`

Creates a new book from a list of Markdown files. It will generate a book configuration file with alle file names specified as chapter. It either prints the result to stdout (if `BOOK` is not specified) or generate the file `BOOK` (or abort if it already exists).

3.1.1 Examples

```
crowbook foo.book --create  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
```

while

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

will prints the same result, but to stdout (without creating a file).

When `crowbook` is runned with `-create`, it can also uses the keys/-values set by `-set` (see below):

```
$ crowbook foo.book --create file1.md file2.md --set author "Pierre
Dupont" title "Mon œeuvre" lang fr
```

will generate a `foo.book` file containing

```
author: Pierre Dupont
title: Mon œeuvre
lang: fr

# List of chapters
+ file1.md
+ file2.md
```

3.2 -set

usage: 'crowbook <BOOK> –set [KEY] [VALUE]...

(or `crowbook <BOOK> -s [KEY] [VALUE]...`

This options takes a list `KEY VALUE` pairs and allows to set or override a book configuration option. All valid options in the configuration files are valid as keys. For more information, see the configuration file page.

3.2.1 Examples

```
$ crowbook foo.book --set html.css style.css
```

will override the CSS for HTML generation (the `html.css` key) to `style.css`.

```
$ crowbook foo.book --set author Foo --title Bar
```

will override the book title to `Bar` and its author to `Foo`.

3.3 -list-options

usage: `crowbook -list-options`
(or `crowbook -l`)

Displays all the valid options to use either in a book configuration file or with `-set`, with a short description. There is also `-list-options-md`, which outputs markdown.

3.4 -print-template

usage: `crowbook -print-template template`

Prints to stdout the built-in template. Useful if you want to customize the appearance of your document. E.g., if you want to modify the CSS used for HTML rendering:

```
$ crowbook --print-template html.css > my_style.css
# edit my_style.css in your favourite editor
$ crowbook my.book --set html.css my_style.css
# or add "html.css: my_style.css" in my.book
```

Note that it is possible to use this option in conjonction with `-set`, though it is currently only useful for EPUB template:

```
$ crowbook --print-template epub.template --set epub.version 2
# Returns the template for Epub 2 (default one)
$ crowbook --print-template epub.template --set epub.version 3
# Returns the template for Epub 3
```

3.5 -verbose

usage: `crowbook <BOOK> -verbose`

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

3.6 -to

usage: `crowbook <BOOK>-to [FORMAT]`
(or `crowbook <BOOK> -t [FORMAT]`)

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.

3.6.1 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,.

3.7 `-output`

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

ChangeLog

0.2.0 (unreleased)

- Command line arguments:
  - New argument `-print-template` now allows to print a built-in template to stdout.
  - New argument `-list-options` prints out all valid options in a config file (or in `set`), their type and default value.
  - New argument `-set` allows to define or override whatever option set in a book configuration.
  - `-create` can now be used without specifying a `BOOK`, printing its result on `stdout`.
- Configuration file:
  - Added support for multiline strings in `.book` files, with either '|' (preserving line returns) or '>' (transforming line returns in spaces)
  - New option `display_toc` allows to display the table of contents (whose name, at least for HTML, is specified by `toc_name`) in HTML and PDF documents.
  - Option `numbering` now takes an int instead of a boolean, allowing to specify the maximum level to number (e.g. `1`: chapters only, `2`: chapters and sectino, ..., `6`: everything).
- Rendering:
  - Added support for numbering all headers, not just level-1 (e.g., having a subsection numbered `2.3.1`).

  - Tables and Footnotes are now implemented for HTML/Epub and LaTeX output.

- Refactored `Book` to use an HashMap of `BookOptions` instead of having like 42 fields.

0.1.0 (2016-02-21)

- initial release