Announcing Crowbook 0.11.0!

Lise Henry

I'm pleased to announce the 0.11.0[1] release of Crowbook[2], a converter from Markdown books to PDF, EPUB and HTML.

Since I never made a proper blog post to talk about Crowbook (in english at least) (the first release was announced on the Rust reddit[3] but didn't go into much detail), and it's a good time in the year to pause a little and look at what has been done in the last 12 months, I thought it would be a good occasion to do so.

What is crowbook?

So, what is Crowbook? It's yet another tool that takes Markdown files and convert them to HTML, PDF and EPUB. The format is quite simple, e.g. looking not further than this very document[4], the code looks like this:

```
---
author: Lise Henry
title: Announcing Crowbook 0.11.0!

output.html: 0.11.0.html
output.pdf: 0.11.0.pdf
---

Annoucing Crowbook 0.11.0!
```

[1] https://github.com/lise-henry/crowbook/releases/tag/v0.11.0

[2] https://github.com/lise-henry/crowbook

[3] https://www.reddit.com/r/rust/comments/46xla6/crowbook_yet_another_converter_from_markdown_to/

[4] http://lise-henry.github.io/articles/0.11.0.md

```
==============================

I'm pleased to announce the
[0.11.0](https://github.com/lise-henry/crowbook/releases/tag/v0.11.0)
release of [Crowbook](https://github.com/lise-henry/crowbook),
a
converter from Markdown books to PDF, EPUB and HTML.
[...]
```

So, yeah, it's Markdown, plus some options to set title, author, and the ouput files that must be generated (here, PDF and HTML). With this, you can then run

```
$ crowbook -s 0.11.0.md
```

and it will generate an HTML[5] and a PDF[6] file. The `-s` argument is short for `--single`, and it simply tells `crowbook` that it is processing a Markdown file embedding some additional options. The other way of calling Crowbook is without with argument and with a full book configuration file, listing all Markdown chapters that are part of the book:

```
author: Jane Doe
title: A great novel
lang: en
cover: some_cover.png

output.html: great_novel.html
output.pdf: great_novel.pdf
output.epub: great_novel.epub

- intro.md
+ chapter_01.md
+ chapter_02.md
```

So, that's basically a list of key/value pairs in a YAML syntax (you can run `crowbook --list-options` to get the full list of valid options), followed by the lisf of chapter names, prefixed by `-` (if they are not numbered), `+` (if they are numbered) or `!` (if the title should be hidden).

[5] http://lise-henry.github.io/articles/0.11.0.html
[6] http://lise-henry.github.io/articles/0.11.0.pdf

I won't go too much into the details here, you can look at the Github page[7] and the user guide[8] for more information. There is also an online demo here[9] if you want to quickly test how it looks without insalling anything.

Current features

The current features of Crowbook are the following:

- It should be able to generate valid HTML, EPUB and LaTeX files, and can transform the later to PDF if you have a working installation of LaTeX. There was the beginning of a work on rendering to ODT but it is nowhere yet.
- It tries to make your document "typographically correct": in english (or other languages) this mostly means this so-called "smart quote" feature, in french it also involves quite a lot of work on non-breaking spaces (there is currently no other supported languages for this feature, though I'm open to it if there are some needs).
- There's this weird feature where it can generate proofread copies, connecting to a local LanguageTool[10] server for grammar checking, and it can also detect repetition (using another library I developed, Caribon[11]).
- It's localized in english (the default) and french.

Why did I start Crowbook?

Well, first, honestly, because I could and it was fun. But there were some other reasons, too. The thing is, I'm french, and I'm a somewhat semi-professional (whatever that means) fantasy writer, and I found the support of correct french typography in other tools (except LaTeX, but it's no good at exporting to EPUB) quite lacking.

[7] https://github.com/lise-henry/crowbook
[8] http://lise-henry.github.io/crowbook/book/book.html
[9] http://vps.crowdagger.fr/crowbook/
[10] https://languagetool.org/
[11] https://github.com/lise-henry/caribon

> (Really, french typography is quite a pain. In english, you just stick e.g. a question mark to the last word, don't you? In french, we add a space. But it should be a non-breaking space, so the `?` doesn't end at the beginning of the newline. Except it shouldn't just be (though these days it is more and more used because really few tools do it right) the "standard" `U+0020` non-breaking space, but a narrow one, so the questionmark doesn't end too far away from the actual question.)

Of course, I could only have written some kind of "pre-processor" and continued using e.g. pandoc[12] (which was what I used before), but well, Rust was so cool to program in that I felt I should rather somewhat reinvent the wheel.

> (I did, somewhat later, move these "formatting" functions to a separate crate, crowbook-text-processing[13], which while being name-prefixed by "crowbook-" can be used elsewhere if you need some automatic french typographic formatter (or smart quotes, actually)).

Comparison, and why didn't I rather contribute to mdBook?

So, compared to other similar tools, I think Crowbook[14] is more focused on:

- novels and fiction than technical books (code blocks or tables should work but they will certainly not look as nice as with other converters, and there is no support for e.g. mathematics or inline HTML);
- correct typographical settings (without requiring any work from the writer), than not ever breaking some quote that should have stayed straight because it can no longer be copy/pasted on your terminal (though it's possible to disable it);
- EPUB and PDF than HTML (HTML should still be valid and hopefully not too ugly, but comparatively there is probably less relative work on it than other tools);

[12] http://pandoc.org/
[13] https://crates.io/crates/crowbook-text-processing
[14] https://github.com/lise-henry/crowbook

- french (the github pages, code, and even currently the docs are still (only) in english, but it tries to respect french typography when `lang` is set to french, and it is localized in french).

I won't really talk about why I didn't contribute to other tools instead, and rather link to this wonderful picture posted on the Rust Reddit[15], but will focus on "Why not contribute to mdBook[16] instead?", since it is also written in Rust and is a quite similar tool.

There are some not-really-good reasons, and I don't want to evacuate them:

- creating a new project is fun, contributing to an existing project is hard;
- having to work with other people can be time-consuming but also a bit frightening: when I push a commit to my own Github project, I don't have to worry about people rejecting it and saying it's horrible (I'm not saying this kind of reaction happens often in the Rust community, but I mean, I think it requires a bit more self-confidence).

But I also think the two projects really have different focus, and trying to merge too different approaches might lead to more friction than do any good (e.g., as a writer, I enjoy the proofreading features, but they might seem ludicrous to insert into a converter). Maybe trying to make features usable as independent libraries can be a better way to ensure a bit of code reuse? I already separated the formatting part to crowbook-text-processing[17]; I'm thinking about doing the same for the EPUB building part (though doing it cleanly might require a bit more work and thinking).

What I learned

When I started developing Crowbook, it was more as a toy project than a serious thing, and I didn't expect to invest that much time in it. But then I started using it for my books, which gave me more motivation to add features or fix bugs. So I guess dogfooding is good (though I don't like the term, I'm more of a cat person).

[15] https://www.reddit.com/r/rust/comments/5l2uc6/me_irl/
[16] https://github.com/azerupi/mdBook/
[17] https://crates.io/crates/crowbook-text-processing

When I chose the Rust language for this project, it was only because I liked this language and wanted to get a bit deeper into it; not because I felt it was the more appropriate. Clearly, there is nothing low-level or that is "systems programming" stuff or even that requires that much safety or performances (anyway in most of my usage the call to LaTeX will take an order of magnitude longer to complete). I still found it really enjoyable, and I really think that Rust is relevant to even non-really-that-low-level projects.

There are various aspects of Rust that I thought I would never get into, but still did: e.g. I started being like "I can clone anything I really don't care about performances", yet I still wrote an article about Optimizing string performances[18] (I guess the attention to performances is quite contagious in the Rust community). I also got deeper into macros than I thought I would because I wrote a somewhat hacky internationalization library[19].

I really enjoyed the type system of Rust and the impression it gave me that the compiler would do a lot of work to make sure that my code was correct. I don't think I have ever been that confident writing code: it's still astonishing to me when I make some substantial modifications to my code, and the compiler point me to some problems, but when it does compile, it works correctly on the first try. I was not used to that. Obviously, now I know that there are other more high-level languages that can have similar guarantees (e.g. OCaml), but while there *was* some friction induced by the low-levelness of Rust (namely the borrow checker), once I had learned it it was on a whole rather minimal.

I also found the Rust community quite nice, and I think that it's also because of its atmosphere that I ended up digging into things that I didn't know that well to begin with.

This isn't to say that everything is perfect in the Rust world: most notably, I think the ecosystem is still immature on certain aspects. E.g., if I ended up writing another internationalization libary, it's because there isn't a current standard on how to do it; actually, looking at the reverse dependencies of internationalization libraries on crates.io, I think Crowbook might very well be the first Rust program that is localized in another language than english. There are libraries that work just fine but are terribly undocumented (though there are also wonderful documentations, such as clap[20] and regex[21]). And, of course,

[18] http://lise-henry.github.io/articles/optimising_strings.html
[19] http://lise-henry.github.io/articles/localization.html
[20] https://crates.io/crates/clap
[21] https://crates.io/crates/regex

there are libraries that I would love to exist in Rust but still are not here (yet?) (e.g. I'd love a grammar checking library in Rust instead of Java or Python, or a Rust equivalent to PyPdf). Still, on a whole, it's pretty nice, and Cargo and Rustup are wonderful tools.

To sum it up, I'm quite pleased that I was able to develop a program that at least fits my needs (if it fits other people's ones too that would obviously be great). It's probably not great code but thanks to Rust I'm quite confident that all hell won't break loose if I try to add a new feature or fix a bug. If I had to choose a word to describe Rust (which is a current thread), I'd definitely would feel "empowering".