Make books // Nick Berendsen

Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc id cursus metus aliquam eleifend mi in. Etiam dignissim diam quis enim lobortis scelerisque fermentum dui. Adipiscing bibendum est ultricies integer quis. Quis vel eros donec ac odio tempor orci. Dolor magna eget est lorem ipsum dolor sit amet. Sit amet consectetur adipiscing elit ut aliquam purus. Faucibus a pellentesque sit amet porttitor eget dolor morbi. Maecenas sed nim ut sem viverra. Non quar cus suspendisse faucib erdum. Fermentum posu a nec tincidunt praesent bi tincidunt augue interd

Make books

Make books

Nick Berendsen

Revision 3.0 July 2020

Desbeers http://www.desbeers.nl

Many thanks!

To all the developers from the Open Source communities
who gave us so much fun stuff to play with.

The LaTeX community

The Pandoc community

And all you others out there!

CONTENTS

INTRODUCTION

Write a beautiful book

I like to write books and I like to make books. Both literarily. They have a lot in common and even more *not* in common. Writing is a creative process. Words, words and words. On the other hand, *making* books is mostly a technical process.

It's a nice combination.

For years and years, I'm searching for the "perfect" book environment. Perfect for writing and perfect for self publishing. It's been a long journey, I made my first books around 2004. Half the time was writing; the other half programming. At that time, I was much into web programming so my "publishing system" was home-made in PHP and MySQL. Fun to do, impossible to maintain.

So, time to make a new system! Instead of reinventing the wheel I looked around, learned LaTeX, found Pandoc, learned a bit of shell and Lua scripting and glued that whole stuff together to my brand new writing environment.

I'm not a writer, nor a programmer, just a hobbyist who does this for fun. It was a lot (lot, lot, lot) try-and-error and I'm sure many parts can be done better but that does not matter for me. That's part of the fun, just keep learning!

It's all on Github, a repo with all the bits and pieces to make beautiful PDF's and ePub books. An sample book, which is actually this book, is included and it explains it all...

It's my small contribution to the Open Source World who gave me all this great stuff to play with!

Enjoy,

Nick

1

INSTALLATION

A basic technical knowledge is required to use this system; it's working from the `terminal`. However, I will give it a fair try to help you with it. As always, Google is your best friend if you read this chapter and it makes no sense to you, haha!

This system is only tested on macOS Catalina and Ubuntu 20.04. There is no Windows support and that is also not a my todo list; I have no clue about that system.

Software

The following software must be installed on your computer:

- `Pandoc`, the swiss-army knife for file converting.
- A `LaTeX` distribution like MacTEX for the Mac. For Ubuntu, I did `sudo apt install texlive-full` to get all the goodies in one time and no complains about missing packages.
- The `zsh` shell; standard on macOS Catalina; install it on Ubuntu.
- `rsync`; should be standard...
- The `Make` program. Part of Xcode or grab it from Homebrew when you are on a Mac. Else; do your Linux magic.
- For a "Kobo" ePub, "kepubify" and "ImageMagick" are needed. Get it here and here.

Font

The "drop caps" font `GreatVibes` has to be be installed on your system. It's in the `fonts` folder. On Mac, use `fontbook`. On Linux, as always, find your own way. If the font is not properly installed you get the following friendly warning:

```
Package fontspec Error: The font "GreatVibes" cannot be found.
```

And the whole fun stops... I'll tried to "just use the font from the folder" and, although I could get that done, it gave more troubles... LuaLaTeX remembers where fonts are and will never forget it; unless

you do magic... Didn't want to do that kind of magic; not worth the risk of breaking even more. So... just install the font, it's gorgeous!

For PDF, the font `otf-cm-unicode` is used. You have to have it installed as part of your LaTeX installation. For Arch Linux, it is in the "aur".

Place the folder

The content of the `Make-books` repo can be copied to any location on your computer. Mine is in `~/Documents/Leesvoer/scripts` for example. Only one thing, don't move any files inside this folder, keep them in place.

Add the scripts to your `$PATH`

The folder `/terminal` has the shell scripts to make the books. It's handy to have this folder in your `$PATH`; saving you a lot of typing.

On my Mac, I dropped a file containing the full path to the scripts: `/Users/Desbeers/Documents/Leesvoer/scripts/terminal` into the folder `/etc/paths.d`. That did the job for me.

Linux? Up to you!

That's it! Now a bit of configuration and you're good to go!

2

CONFIGURATION

GOOD news! There is really not much to configure for this build-system. There is only one file that needs a bit of your attention...

`/config.zsh`

It needs to know where to find your stuff...

```
#!/bin/zsh

### Configuration of this build-system
### ----------------------------------

# Base directory of the books; full path.
# Used when making collections and all books.

books="/Users/Desbeers/Documents/Leesvoer"

# Export directory for the books; full path.
# Automatically created if it does not exist.

export="/Users/Desbeers/Documents/Mijn boeken"

# PDF options
# -----------
# The options are from the LaTeX Memoir class.
# See: https://www.ctan.org/pkg/memoir

# The default papersize:

papersize=ebook

# The default fontsize:

fontsize=11pt
```

And yeh; you can have spaces in your path name... As long as you have them between those nice quotes...

Paper and font options

Paper size

Those options are defined in the LaTeX Memoir class... Oops, that sounds pretty nerdy... Yes it is! See their manual if you want to know all the available options.

Just a few common ones: `ebook` (9 by 6 inches, my default), `a4paper`, `letterpaper`, and `legalpaper`. If you like more options, you know where to find the manual now!

Font size

Again, defined in the LaTeX Memoir class. There are not so many options fortunately. They are `9pt`, `10pt`, `11pt`, `12pt` and `14pt`. And no, there is no `13pt`. I don't know why, it's simply not an option. Because... just because I guess...

Both options can be overruled in the `make-book.md` file for the individual books. How this is done I will explain later.

All the rest *is as it is*, as we always say at my job. That means, any other kind of customisation that you might want to do requires dirty hacking... Lucky for you, the code is full with comments to make your wishes at least *a bit* easier...

Write a book

Well, I'm not going to tell you how to write, I'm sure you know, otherwise you wouldn't read this. I'm only gonna tell you how the write *for this build-system* because there are some strict rules to follow. Read and don't break the rules!

1

THE ANATOMY OF A BOOK

Ok, let's have a look at the structure of a book. Not being an expert, I might use the wrong wording once in a while, but that's ok I hope. It is not rocket-science, so I'm pretty sure my explanation will be good enough. I'll trow in a bit of LaTeX terminology because, well, that's the only terminology I know, haha!

Please not that is *not* the really the technical structure of the book created by this build-system; it's just to give it a general idea.

The book:

```
cover page (optional)
half title
title
copyright
dedication (optional)
table of contents
\frontmatter
foreword (for example)
\mainmatter
\part (optional)
chapter 1
chapter 2
\part (optional)
chapter 3
chapter 4
\backmatter (optional)
afterword (for example)
```

That's it! That's the anatomy for a book. At least, for my books.

A bit of additional explanation:

Half title & title

This is just something I see in my "real books" at home. Real books just have that. On the first page the "half title" which is actually *not* half the title, but only the title and not the author. Don't know why they

call that kind of page *half title* but that's the official name. The second page is the title of the book, followed by the authors name. It's called the *title* page; even thought it has the title *and* the author. Anyway, I like that look, so here it is!

Copyright

This page will automatically created based on the provided metadata. See next chapter for that.

Dedication

Maybe not the correct name, but it is a page that comes before the table of contents. I use it as dedication page but you can use it for whatever you like.

\frontmatter

These are the pages that will go before the start of your *real* content. A foreword for example, or an introduction.

\part

A book can be divided into *parts*. Maybe for *episodes* for example. Further-on I will explain how to make those parts.

\backmatter

These are the pages that will go after your *real* content has ended. You can use it for an afterword for example.

2

MARKUP YOUR WORDS

The books have to be written in *markdown*. Cut and paste from Wikipedia: "Markdown is a lightweight markup language with plain-text-formatting syntax. Its design allows it to be converted into many output formats."

There are many flavours of Markdown and since I use *Pandoc* for the book creation it's obvious I use Pandocs flavour. See their manual for the details. However, there are some "rules" to follow, specifically for this book building system:

Headers

- Chapters are marked with `#`.
- Sections within chapters are marked with `##`.
- Subsections within sections are marked with `###`.

And that's it! There are no more headers available. This for a good reason. When creating an ePub; the headers are shifted according their place in the book. A book can be divided into `books` (when making collections) and `parts`. So, if a chapter is inside a *part*; the header will be shifted. If the *part* is part (haha) of a *book* it is shifted one more time. So, that is why `####` and `#####` are not available. The `######` is special, see later on...

Paragraphs

In good *Markdown style*, "enters" in the source files are more or less ignored. One enter does nothing; two enters are just starting the text on a new line. By design, there is no space between paragraphs. You can enter as much as you like, but you will never even get *some* space between paragraphs this way. They are simply ignored.

To get some space between the paragraphs, as you can see here, this system is "misusing" the `horizontal rule` tag. In Markdown, that's "`---`" or variations on that. See the Pandoc documentation.

So, the `horizontal rule` is not available for its normal intended use. Too bad, but I think this is the "cleanest" hack I could think of.

Chapter's Précis

Markup `######` has a special function. A line with this markup will be converted to a *Chapter's Précis*. Thats a line below the chapter name. Call it a *sub-chapter-name* or something, it's a "subparagraph" in LaTeX language. I use it to write the date and place underneath a story for example:


1

WAT EEN GAAF LEVEN!

Zondag, 28 November 2004 | Golf van Aden

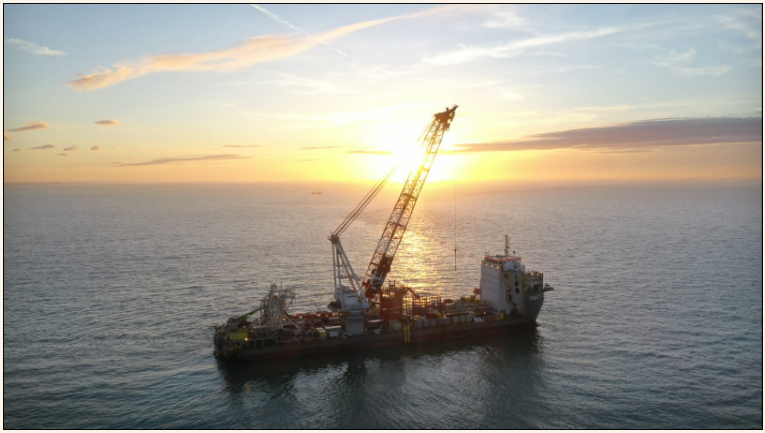
Op het gemakje varen we door de golf van Aden. Korte broek en slippers aan want het is hier ruim dertig graden. Geweldig om

Chapter's Précis

Images

Images are done in "normal" Markdown; except you can set the width for an image as well. This is not something I invented; it's part of Pandocs flavour of Markdown.

```
![Caption for image](images/chapter-header.png){width=85%}
```



Caption for image

The `caption` is not required; nor the `width`. If no width is specified; it will default to 92% because I think that looks nice with the indentation of the paragraphs. Assuming the image is larger than the page itself...

All the rest

Well, just "normal" Markdown. Plain (haha) and simple.

Inline styling:

```
Lets write in *italic* or in **bold**...
```

Lets write in *italic* or in **bold**...

```
I made a ~~mistake~~ stiketrough...
```

I made a ~~mistake~~ striketrough...

Quotations:

```
> "I'm a quote!"
```

> "I'm a quote!"

Lists:

```
- I'm a list item
- And so am I!
```

- I'm a list item
- And so am I!

They can have numbers as well...

```
1. I'm a numbered list item
2. And so am I!
```

1. I'm a numbered list item
2. And so am I!

Footnotes:

Footnotes can be in the paragraph itself `^[This is a footnote.]` However, the Pandoc way of putting footnotes in a book also allows for naming the footnote inline like so:

```
Here is a footnote reference,[^1]
```

and the footnote itself can be at the end of the document like so:

```
[^1]: Here is the footnote.
```

Last but not least: have an empty line between all the block elements; eg, headers, lists, images, quotes etcetera or else strange things can happen. Also, at all times, end your documents with an empty line. You've been warned!

3

PARTS, PREFACE, AND OTHER SPECIAL PAGES

A chapter is a chapter, plain and simple. But what about the other bits and pieces that makes a book *a book*?

Special pages are also written in Pandoc's Markdown. To let the build-system knows if a page is special, you have to add a `class` to your heading. That's all. All these files, except the `dedication` page are living in the same folder as your *normal* content. So, order is important! See *the folder structure for your book* page in this book.

Frontmatters

The `frontmatter` pages should go before you start any chapter or optional `part` page. Please note this should not be a *dedication* page. It can be a foreword for example.

To make a `\frontmatter` page, write the following header:

```
# My frontmatter title {.frontmatter}
```

And ten just write whatever you want to write. You can have as many `frontmatters` as you like; as long as they have all the above class added.

Parts

To make a `\part` page, write the following header:

```
# My part title {.part}
```

Parts can have normal text as well; just like any other chapter, frontmatter or backmatter page. It's just styled in a different way to make it look like a real part division. A PDF part-page will always start on the right; followed by an empty page.

Backmatters

To make a `\backmatter` page, write the following header:

```
# My backmatter title {.backmatter}
```

And then, write along again! You can have as many `backmatters` as you like; as long as they have all the above class added.

Preface

A `preface` page is a separate Markdown file that lives in the `/assets` folder. See the chapter about the folder structure of your book later on. It does not need any `class`; you just write it like a chapter page.

4

METADATA FOR YOUR BOOK

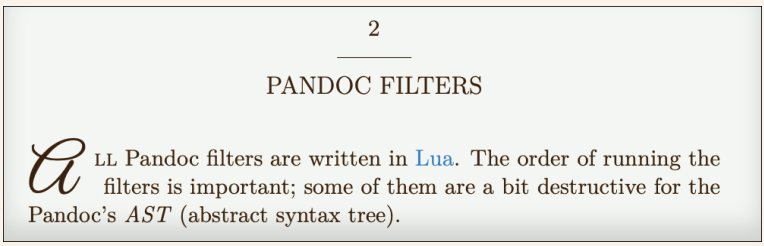
Every book has a `make-book.md` file in the `/assets` subfolder of your book; providing the metadata for the book. It's just plain text and without this file, the book will not be created. For the nerdy people: it's actually a `yaml` format file.

```
---
title: The name of the book
author: Nick Berendsen
date: 2020
revision: Revision 1.0 March 2020 (optional)
lang: nl-NL
subject: Manual (optional)
publisher: Desbeers (optional)
rights: Creative Commons (optional)
publisher: Desbeers (optional)
chapter-style: plain (optional)
revision: Revision 1.0 March 2020 (optional)
papersize: ebook (optional, defaults to your global setting)
fontsize: 11pt (optional, defaults to your global setting)
belongs-to-collection: My collection (optional)
group-position: 1 (collection number, optional)
---
```

The order of the tags does not matter.

Chapter styles

Normally, chapters look like this:



For books that have numbered chapters by itself, like *Chapter 1*, it looks a bit ugly. If you set the `chapter-style: plain`, the number and line above the chapter name will be removed.

Any other chapter style provided by the LaTeX *Memoir* class can be used for PDF output. For ePub output, there is no style provided in the CSS file, besides for the standard style and `plain`.

Paper and font options

As I described in the *configuration* page; all options are coming from the LaTeX Memoir class. Have a look over there if you skipped that chapter...

Revision

This does not really *do* anything. Whatever you set here will literarily be dropped on the copyright page.

Collections

The `collections` related tags have no function for the output of the book. They are not yet supported by Pandoc. I hope that will come in the future; it is requested on Github (Update: my PR is just merged; wowoo!!). The tags are "official ePub3" standards and programs like Calibre supports it. Apple's Books unfortunately not; as always has Apple its own mind about "standards"...

For now, the tags are used by the `make-collection` and `make-all-collections` build-scripts to create one *collection book* out of the other books who have the same `belongs-to-collection` tag. I will explain that later.

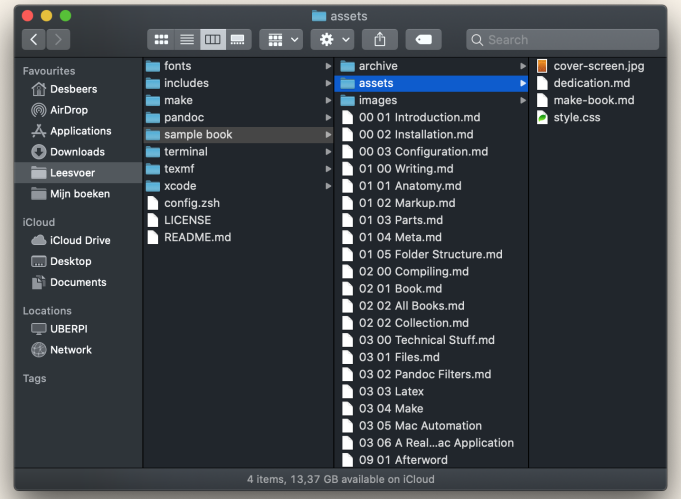
All the rest

The `make-book.md` file is just a part of the Pandoc possibilities, so, anything that Pandoc supports can be in this file. In theory... Lot's of Pandoc options are set by the build-scripts so they might conflict with each other. According the manual, the command line, so, these build-scripts, will "win" from settings in the `make-book.md` file...

5

THE FOLDER STRUCTURE OF YOUR BOOK

THERE are some rules to follow to get everything nice and dandy. Your book must be organised in a specific way:



The folder structure

All your *normal* text files should be in the root folder. That includes the `\frontmatter`, `\part` and `\backmatter` pages. The pages are processed in alphabetical order. As you can see above, I just numbered the files in `part-chapter-name` order. So, reordering pages is just a matter of renaming the files.

THE `assets` FOLDER

There are a couple of files in this folder:

dedication.md

> The *dedication* file for your book, as described before. It is not required; however, feel free to thank me!

cover-screen.jpg

: If this file exists, it will be used as cover page when creating PDF's and ePubs. It is not required.

make-book.md

: The `make-book.md` file as described before.

style.css

: If this file exists, it will be included in the ePub export. Your *one and only* chance to override the default styling!

The `images` folder

Of course, this is where the images for your book are stored. Reason to make this a "rule" is because if you want to make *collections* of your books; the build-script should know where to find those images. Also, it is a smart behaviour to give all the images a unique name. When *collecting* a book; all your images will be trow into one big pile. If there are duplicates, well, they will be overwritten...

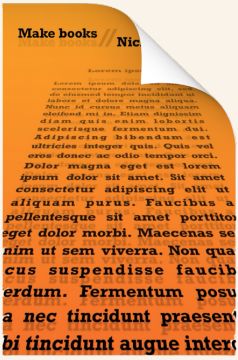
The `archive` folder as seen in the screenshot

Not required. I just store all non-essential files that I still want to keep in the "book folder" into this `archive`.

Compiling

Yeh, a fancy word, better so say *nerdy* word for making the actual book... Compiling... Let's get our hands dirty!

It's time go get that plain text into a beautiful book!

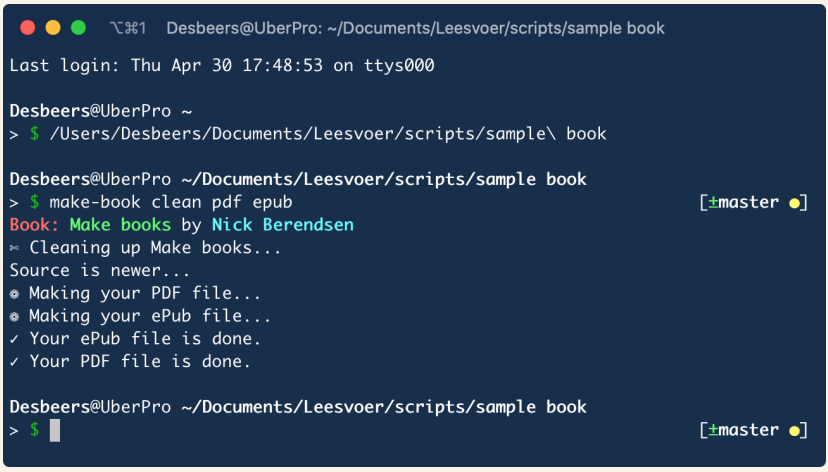


1

COMPILING YOUR BOOK

Compiling the book will be done on the command line with the `make-book` command, followed by some optional arguments.

You have to be in the folder containing the book. A picture says more than thousand words, so please have a look:



Compiling this book

If you run `make-book` without any argument it will give you a PDF and an ePub in the export folder. The optional arguments for the `make-book` command are the following:

Clean the export folder: **clean**

This is a special command. It's doesn't create anything. Even the opposite; it will *clean* whatever book might have been compiled in the past and is still in the export folder. Don't worry; it will not get even close to your writing stuff! Please note; the script, or more precisely, the Make program, is smart. It will know when you have changed something in your book and will always grap *the latest and greatest* files. So, there is no need to *clean* for that reason.

Output formats: **pdf**, **print**, **epub**, **kobo**

Just put any type you want to compile as argument. You can add as many as you like at the same time and order does not matter.

A few notes about the export formats:

pdf & **print**

They are basically the same. The only difference is that `pdf` will have the (optional) cover included and has a slightly coloured background while `print` will never have a cover and is just plain white. That makes the `print` file more suitable for uploading to a POD service for example. Also, `print` does not have coloured hyperlinks; even thought the links are still working.

epub & **kobo**

An `epub` is, you might have guessed it, an ePub! An `kobo` is also an ePub, however, optimised for the Kobo reader.

tex (LaTeX format)

It will be the complete `tex` file but I'm pretty sure you will not be able to compile it with any other software. The file will be stuffed with dependancies on this build-system and your images will be completely forgotten...

2

COMPILING ALL BOOKS IN ONE TIME

Inside the `/terminal` folder, there is another handy script. It's called `make-all-books` and guess what it does? Yes, it will compile all the books the script can find.

It does this by sniffing your whole disk starting from the folder defined in `config.zsh`.

It just search for every `make-book.md` file; double-checks if it is actually a book and then tells the `make-book` script to compile that book for you. All automatically, wow!

All arguments are the same as with the single `make-book` script.

So, to compile all and every book to a PDF, just run the following command in the terminal:

```
make-all-books pdf
```

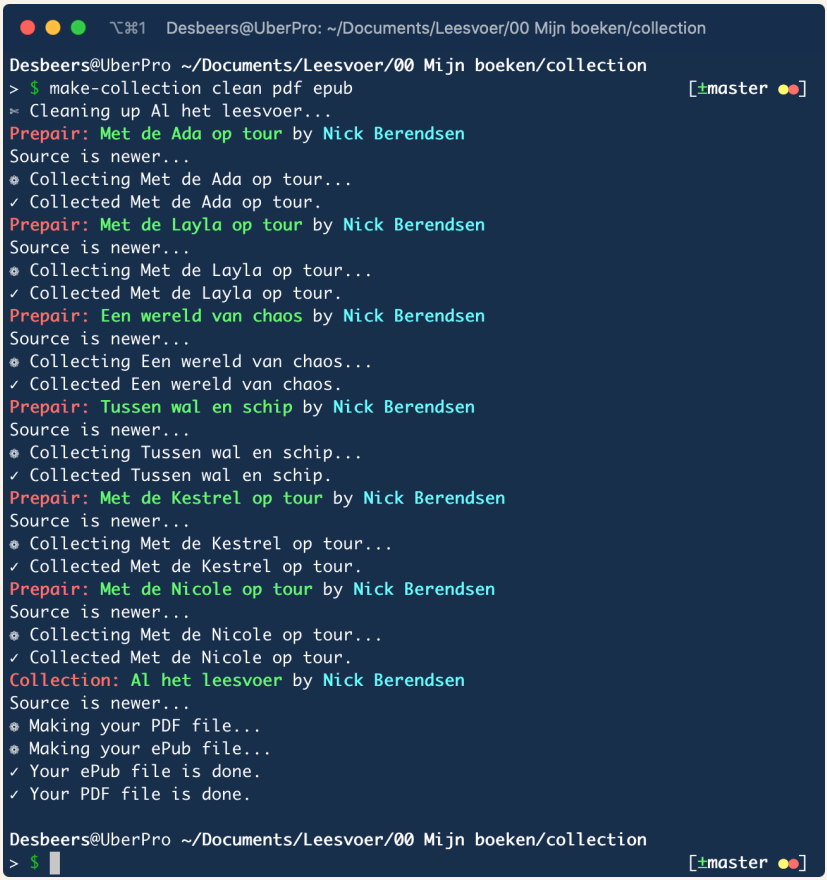
Assuming you put the scripts into your `$path` as described on the installation page of this book, you can run this script from anywhere.

3

COMPILING BOOK COLLECTIONS

ANOTHER script in the `/terminal` folder is `make-collection`. It's ment to make a book, containing any of your other books. A big, fat, huge masterpiece of writing!

It might be a bit of a struggle for your computer because collecting more than thousand pages, as I have tried, is not an easy job... and for sure not a quick job... But doable.



Compiling a compilation of my own books

Unlike the *normal* books, this kind of book needs a slightly different `make-book.md`. So different actually, that it is called `make-collection.md`. All the other files in the folder are just the same.

This file is where the collections related tags, as described in the *metadata chapter*, are essential. It still needs the other tags as well, of course. Name, title, etcetera.

Give your collection a name

```
---
belongs-to-collection: My Huge Collection
---
```

Add books to this collection

To get a *normal* book into this collection; give it *exactly* the same `belongs-to-collection` tag, together with a number in its `make-book.md` file

```
---
belongs-to-collection: My Huge Collection
group-position: 1
---
```

Thats all. Make sure you are in the folder of your collection and enter the following command in the terminal:

```
make-collection pdf epub
```

The arguments are again just like when you are making a normal book. You can make a collection in any kind of format that you can make your normal books.

Additional files

If you like, you can add additional front- and backmatter pages to your collection. They are just like they are in a normal book, also in the same place. Keep the naming of the files in mind! Your `compilation` book will be collected in a temporarily folder, together with the selected books. The name of the book in the temporarily folder will start with

the number from the `group-position` tag. . So, most probably starting with 1. As long as you start the name of your `front matters` with a 0; you are safe. And name your backmatter "99-name", unless you have more than 99 books of course, haha!

Backmatter

You can add a `backmatter`; however, the header must be named a bit different than normal. This is because the build-system has no clue about the difference between a "collected backmatter" and a "collection backmatter. So, define the header as follows:

```
# My collection backmatter page {.backcollection}
```

Make all collections

Just as with the `make-all-books` we also have a `make-all-collections` in the `/terminal` folder. It behaves exactly the same as well.

All my books stuffed in one ePub!

4

COMPILING TAGS COLLECTIONS

Another script in the `/terminal` folder is `make-tag-book`. It's ment to make a book, containing any tag found in all your source files.

Same as with *collection* books, this kind of book needs a slightly different `make-book.md`. For "tag books", it is called `make-tag-book`. All the other files in the folder are just the same.

This is where you define your `tag`. It still needs the other tags as well, of course. Name, title, etcetera.

```
---
tag: #favorite
---
```

It actually just search each and every file for whatever you define as the tag. It doesn't have to be a specific `#` word.

Thats all. Make sure you are in the folder of your `make-tag-book.md` and enter the following command in the terminal:

```
make-tag-book pdf epub
```

The arguments are again just like when you are making a normal book. You can make a collection in any kind of format that you can make your normal books.

Make all tags

Just as with the `make-all-books` we also have a `make-all-tags` in the `/terminal` folder. It behaves exactly the same as well.

Technical stuff

Now we are going to get technical! No essential reading, however, if you want to have some more background information about how all these magic tricks are happening you are at the right place!

1

WHAT ARE ALL THOSE FILES?

Let's get trough the piles of folders and files inside the `make-book` folder and give it some explanation....

I will not go trough them one by one; that's too boring and also not so interesting. No, instead I take you on a tour. Let say we are in the folder containing this book and type `make-book clean pdf epub`.

File: /terminal/make-book

`make-book` is a shell script in the `/terminal` folder. It will check if you are in a *correct book folder*. If not, it will punish you and stop.

If it is all ok; it will grab your `config.zsh` in the root folder so it knows where to drop your fresh book in the end. That's why you need to set an `export folder`.

Then it will set-up a whole bunch of necessary *variables* by including `/includes/variables.zsh` . This file is shared by all the other scripts as well to save me from typing over and over again.

Then, the time has come... We are ready for Make. Make is a verb, oh yeh, because it is an application you have installed if you followed the install instructions nicely. Make is a program to make stuff; in our case, a book. Let's make it by running make!

File: /make/Makefile

The `Makefile` is the heart of the build-system; all the smart stuff is happening over there. Make needs that file because it will set a lot of settings, grabs your writing stuff and gives that as homework to Pandoc to do what Pandoc can do best. Convert your Markdown to PDF and an ePub in this case...

To give Pandoc proper homework, the Makefile includes a whole bunch of other files...

Folder: /pandoc/templates

This is where the templates are stored for Pandoc. It should know how we like the layout to be, of course! Pandoc has a lot of *default* templates

but I like my own better. There are templates for PDF and ePub in here, as well sub-templates to get the `dedication` page for example.

Folder: /pandoc/filters

To customise Pandoc's output you can run the content trough filters. And believe me, the content will run really hard! A lot of smart-ass stuff is happening in there, so it even deserves its own page in this book...

Folder: /pandoc/css

Here you can find the stylesheets for the ePub and html output. I did not put much effort in the html styling yet...

Folder: /texmf/tex/latex

Pandoc is doing great, but lets not give it *all* the credit. It does get some help to make that PDF. PDF's are actually made by `LuaLaTeX`. Also a story by itself that will come later on. All I will say now that the LaTeX templates and some font info are stored here. Ok, and I will also tell you now that this script-stuff is temporarily disabling your local tex folder if you happen to have one... So you know...

Folder: /xcode

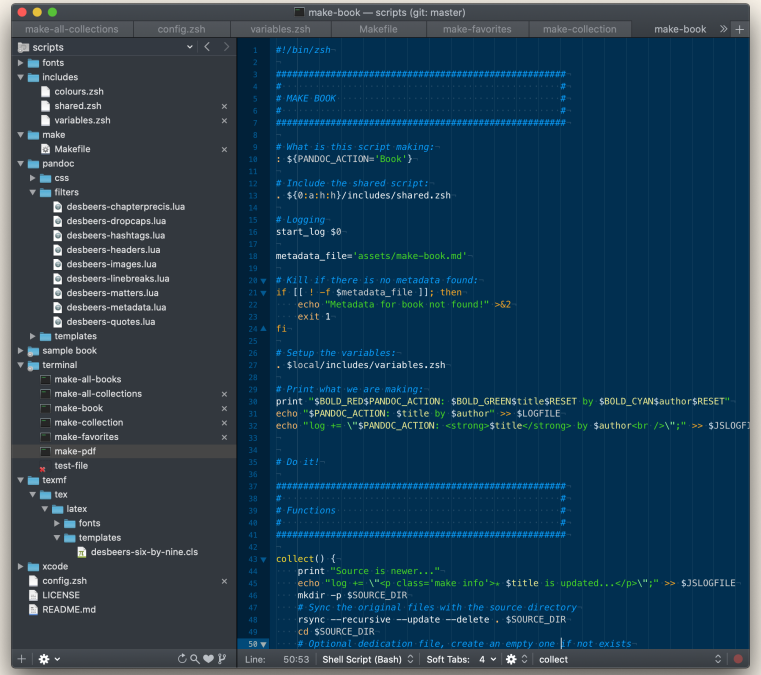
Last but not least... The Xcode folder... Feel free to have a look in it. It's the source code for a Mac application that can run a bunch of the scrips here. It does work for me, somehow, however, I have no clue what I did to make it work so *it is as it is*. Don't dare to ask me any questions about it; because the answer is: **I don't know**, haha!

Bonus file: /terminal/make-file

This script converts a Markdown document into a single PDF. Use it on the command line again:

```
make-file my-text.md
```

The PDF will be dropped on your Desktop. Be careful; if there is already a file over there with the same name it will be overwritten! Thats why this is only a bonus file; it's not too smart...



Textmate, often my best mate!

2

PANDOC FILTERS

All Pandoc filters are written in Lua. The order of running the filters is important; some of them are a bit destructive for the Pandoc's *AST* (abstract syntax tree).

Metadata filter

This filter sets default values for paper and font size if they are not in the book's metadata file. The values are taken from the `config.zsh` file.

Hashtags filter

I write my Markdown in iA Writer and that program writes hashtags (`#tag`) directly into the files. This filter removes them.

Drop caps filter

Makes the first letter of the first paragraph after a `#` header a drop cap. This is done with the Lettrine package for LaTeX and with CSS for ePub/html.

If there is a `blockquote` after a `#` header, it will not get a drop cap, however, the first paragraph after the quote wil get the drop cap.

If the paragraph starts with a quote (''), the quote-symbol will be before the dropped cap. The double quote is hard-coded; so sorry for those who use singles quotes.

This filter must run before the Header Filter or else the headers are most probably changed to something unusable for this filter.

Drop caps are not working in docx files because I don't care too much about that format.

Linebreaks filter

The `horizontal rule` is misused to get some space between two paragraphs. For LaTeX, it will be replaced by a `\bigskip`. For ePub/html by a `<div.bigskip>` and for docx by something I copied/pasted and seems to work.

Chapter précis filter

The `######` header has a special function for me. I use it to write the date/place of a story after the `#` header. In Latex, it is a `\subparagraph` styles as a "Chapter précis"and in ePub/html a div with class `chapterprecis`.

Headers filter

When there are `\parts` in a book, the headers within those parts will be shifted for ePub, html and docx output. This is so that the TOC is correct.

If the `\parts` are also in a `\book`, as with collections; they are shifted twice.

Matters filter

For Latex, it checks the Markdown for `.book`, `.part`, `.frontmatter`, `.mainmatter` and `.backmatter` classes and adds the corresponding LaTeX for that.

It also fiddles with the `\openany` and `\openright` options in the LaTeX output.

Images filter

Figures for LaTeX will be rewritten, so it has a caption without "Fig:" in front and it will have the alignment that I like for all images.

The pictures default to a width of 92% when not set.

Quotes filter

I like *double quotes* (`“”`) and this filter makes sure they are just like that; independent of what's in the Markdown. A quote inside a quote will be a single quote (`‘’`).

3

LATEX

Why make things easy if you can make it very difficult? Yes, give me one good reason *not* to use LaTeX for the layout of a very simple book... Just one good reason... Because it is very difficult? I like that... Because it's like driving to the supermarket in a Ferrari? I don't care... For you info; I play a *real* Fender Telecaster as *hobbyist* too, hahaha!

There was, and still is, a lot to learn for me when it comes to LaTeX. But, that's the reason why I use it. I like the challenge and even more I like the great results you can get if you just keep trying hard enough!

The template

I use the `Memoir class` as base for my templates. The `/tex/latex/` folder has the first part of my template; providing it's own class. The Pandoc template in `/pandoc/templates` is using this class.

The `$TEXMFHOME` path

The `/includes/variable.zsh` script sets the TEXMFHOME variable to this build-system so LuaLaTeX can actually find its stuff:

```
# For LaTeX, set TEXMFHOME to these scripts:
export TEXMFHOME=$local/texmf
```

Once the script is done and the terminal session is closed; everything is again back to normal. So, that means if you fiddle with the `build-scrips` and you expect LuaLaTeX to find your own stuff you are out of luck... Now you know why!

Fonts

I'm using LuaLaTeX for the actual compiling and that means system font are available for me. Yeh, great! For me... However, nice guy that I am, what about you?

You most probably don't have that charming font on your system that I use for `drop caps`. So, for you, yes, special for you, I found out after a long time how to get this right. Install this font!

Don't just change this font for a different one; the result will not be pleasant. And you will hurt my feelings as well, because, really, it took me a long time to get this right.

Whining and crying

LaTeX can be a bitch. One character in the wrong place and boom! A cryptical punishment will follow. It looks like LaTeX doesn't really like any other character than A to Z, hahaha!

If you edit your `make-book.md` for example; sometimes you can use the `":"`, sometimes you can't. I'm afraid it depends on the direction of the wind sometimes... LaTeX is mysterious...

4

MAKE

Well, `Make` is living in an old world where filenames have no spaces. I can live with a Case Sensitive File SySyeM, however, in a world without SpaceS, no...

Let's dump `Make`

No. `Make` is fancy and smart, it knows what to do and what it has done. So, I wanted it for this `build system`. It took me a long time, but I found-out how to deal with it. Your book will be copied to a `cache` directory and will be removed from All Evil Spaces. Then, `Make` can do its usual great work.

But the question is; when to copy? Always? That's lazy... When the Source is Newer than the destination? Oh, yeh! Just like `Make` is doing... So I reinvented `Make`. It checks your book and if anything is newer than we have in the `export folder`; it copies the stuff.

I `Make` it better

Fun thing is; while figuring this out, I'm even smarter than `Make`. If you alter an image in your book, it knows... It updates.. Just brilliant! It knows everything! Even your thoughts, be warned...

Now I still don't like that `Make` is living in its own "old fashion" Spaceless world; but it learned me new tricks that I shouldn't have learned otherwise...

Respect the old!

5

AUTOMATION ON THE MAC

Even thought the Terminal is damn cool in my opinion, it is not very *Mac* like. While I don't mind to spend my time in the Terminal, most people seems to be a bit scared of it. A Mac would not be a Mac if we can simplify this book stuff...

How wrong can you be?

Well, a Mac is not *the* Mac anymore nowadays, sad but true. Cupertino is turning our beloved Macs more and more into a Fort Knox. Gatekeeper they call it; a very appropriate name. And, in general, it's for the good. Our Mac's got a lot safer over the years.

Maybe a bit too safe for hobby-hackers like me. Catalina is pretty unhappy I'm running smart-ass scripts from my home directory. Catalina should be happy instead! I know very well how to run this stuff from *outside* my home directly but I try to be a good Mac citizen. So, up to now, I'm only a bit naughty. Just a bit and it is tolerated. For now...

Meet my Workflow

Fancy Pancy, nowadays its very easy to make Automation Workflows. Every Mac even has an Application to make them; `Automator`. Super cool; click, click, click and I had a `Workflow` to run my scripts I thought...

NOT!

Gatekeeper told me to ****-off; the door between my scrips and my workflow was hermetic closed.

> "YOU WILL NOT RUN SCRIPTS WITHIN YOUR OWN HOME DIRECTORY, YOU NAUGHTY BOY!"

Then meet my Application?

Ok; I didn't get it to work but I had some more tricks on my sleeve. What if I make a Automator `Application` instead of a `Workflow`? Will that work? After I open it and gave permission to sniff my *documents* folder? Will that open the Gate?

Oh, yeh! That works!

We are one step closer. We have a working Application that's not crying too much. So far, so good. But *not* good enough I want to have such fancy Workflow button in my Finder sidebar!

Maybe my Workflow can date my Application?

Hehehe, yes it can! My Workflow will receive files from the Finder and send them to my Application. My Application will run the scrips. Mission completed!

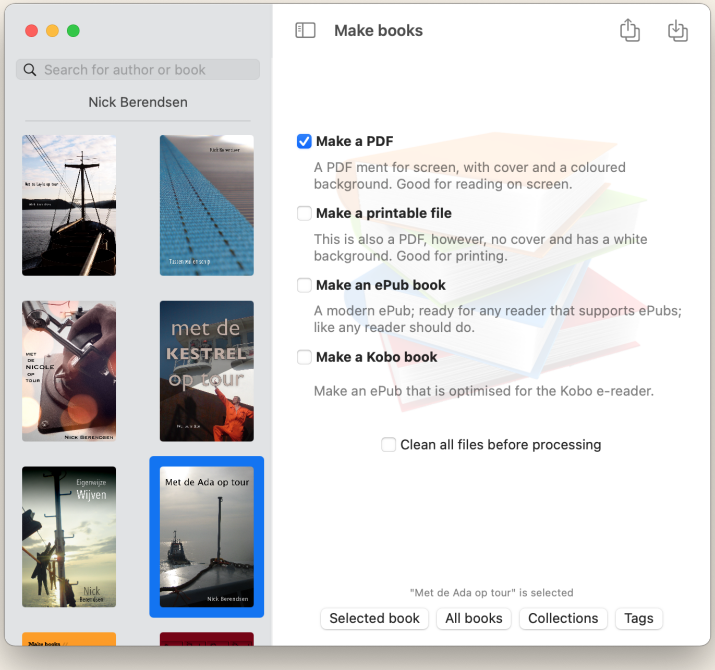
> Sometimes I feel myself so unbelievable cool!

6

A REAL MACOS APPLICATION

Remember that folder named `/xcode` that I don't really want to talk about? It is not the Automator Application and Workflow that I was talking about. No no, this is the Real Deal. A real macOS application written in `SwiftUI`. However, it is just a front-end for the scripts not a standalone application. The "Catalina" version is not updated anymore, I moved to "Big Sur" only.

This frontend works great if all goes well, haha! If the scripts fail for whatever reason you will not know why... Then there is no other option than to go back to the command line... That's where the real magic happens...



AFTERWORD

Well, that was all the fun! There is actually no real *afterword*. This page just exists because because it shows the possibility to say something at the end of your book.

The only thing I might have to say is that this is my first Github project; learning bit by bit. It is not my first contribution to Open Source projects, however, it is my first "project". Hope it is useful for anyone and I promise to try to be a good maintainer of this project. I have a lot of fun with it and I hope you have some fun with it too!

Well, that's it!

Nick