How It Is Made: Master Research Thesis

Perro Tuerto

2020

How It Is Made: Master Research Thesis
Perro Tuerto

Last edition: February 16, 2020

This text is a blog entry from
Publishing is Coding: Change My Mind
www.perrotuerto.blog

All the content is under Licencia Editorial Abierta y Libre (LEAL). "Licencia Editorial Abierta y Libre" is translated to "Open and Free Publishing License." "LEAL" also means "loyal" in Spanish. With LEAL you are free to use, copy, reedit, modify, share or sell any of this content under the following conditions:

1. Content produced must be under some type of LEAL.
2. All files—editable or final formats—must be public access.
3. The content usage cannot imply defamation, exploitation or survillance.

Hecho en México / Made in Mexico

How It Is Made: Master Research Thesis was composed on February 16, 2020.
Document made with LaTeX.

a thesis as a file. In the end, the possibility became a fact, but in a limited way.

I think that the idea of a thesis as a automated repo is doable and could be a better way for research deployment rather than uploading a single file. But this implementation contained many leaks that made it unsuitable for escalation.

Further work is necessary to be able to ship this as a standard practice. This technique could also be applied for automation and uniformity among publications, like papers in a journal or a book collection. The required labor isn't too much, and *maybe* it's something I would engage with during a PhD. But for right now, this is all that I can offer!

Thanks to @hacklib for pushing me to write this post and, again, thanks to my S.O. for persuading me to study a Master's degree and for reviewing this post. Thanks to my tutor, Programando LIBREros and Gaby for their academic support. I can't forget to give thanks to Colima Hacklab, Rancho Electrónico and Miau Telegram Group for their technical support. And also thanks to all the people and organizations I mentioned in the acknowledgment section of the research!

How It Is Made: Master Research Thesis

Uff, after six months of writing, reviewing, deleting, yelling and almost giving up, I finally finished the Master's research thesis. You can check it out in `maestria.perrotuerto.blog`.

The thesis is about intellectual property, commons and cultural and philosophical production. I completed the Master's of Philosophy at the National Autonomous University of Mexico (UNAM). This research was written in Spanish and it consists of almost 27K words and ~100 pages.

Since the beginning, I decided not to write it with a text processor such as LibreOffice nor Microsoft Office. I made that decision because:

- Office software was designed for a particular kind of work, not for research purposes.
- Bibliography managing or reviewing the writing could be very very messy.
- I needed several outputs which would require heavy clean up if I wrote the research in ODT or DOCX formats.

- I wanted to see how far I could go by just using Markdown, a terminal and FOSS.

In general the thesis is actually an automated repository where you can see everything—including the entire bibliography, the site and the writing history. The research uses a rolling release model—"the concept of frequently delivering updates." The methodology is based on automated and multiformat standardized publishing, or as I like to call it: branched publishing.

This isn't the space to discuss the method, but these are some general ideas:

- We have some inputs which are our working files.
- We need several outputs which would be our ready-to-ship files.
- We want automation so we only focus on writing and editing, instead of losing our time in formatting or having nightmares with layout design.

In order to be successful, it's necessary to avoid any kind of WYSIWYG and Desktop Publishing approaches. Instead, branched publishing employs WYSIGYM and typesetting systems.

So let's start!

Inputs

I have two main input files: the content of the research and the bibliography. I used Markdown for the content. I decided to use Bib for the bibliography.

- All files—editable or final formats—must be publicly accessible.
- The content usage cannot imply defamation, exploitation or surveillance.

You could remove my name and put yours, it's permmited. You could even modify the content and write that I LOVE intellectual property: there isn't a technical solution to avoid such defamation. But MD5 checksum shows if the files were modify by others. Even if the files differs by one bit, the MD5 checksum is gonna be different.

Copyfarleft is the way—but not the solution—that suits our context and our possibilities of freedom. Don't come here with your liberal and individualistic notion of freedom—like the dudes from Weblate that kicked this blog out because its content license "is not free," even though they say the code, but not the content, should use a "free" license, like the fucking GPL this blog has for its code. This type of liberal freedom doesn't work in a place where no State or corporation can warrant us a minimum set of individual freedoms, as it happens in Asia, Africa and the other America—Latin America and the America that isn't portrayed in the "American Dream" adds.

Last thoughts

As a thesis works with a hypothesis, the technical and legal pathway of this research works with the possibility of having a thesis as an automated repository, instead of

only to then drop-out. In a country with many crises, scholarships are granted to improve your communities, not only you.

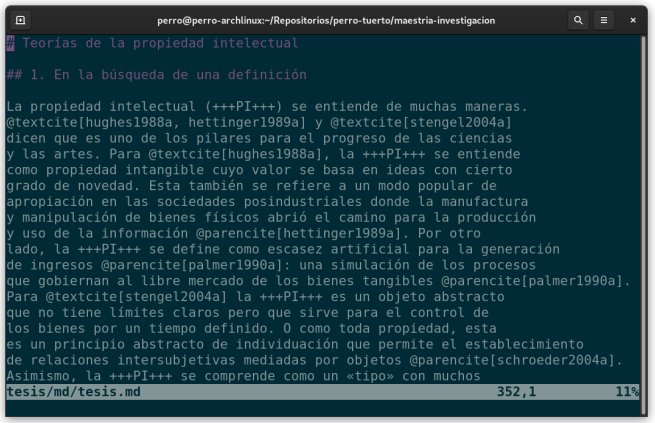
In general, I have the conviction that if you are a researcher or a graduate student and you already get paid—it doesn't matter if it's a salary or a scholarship, it doesn't matter if you are in a public or private university, it doesn't matter if you get the money from public or private administrations—you have a commitment with your community, with our species and with our planet. If you wanna talk about free labor and exploitation—which does happen—please look at the bottom. In this shitty world you are on the upper levels of this nonsense pyramid.

As a researcher, scientist, philosopher, theorist, artist and so on, you have an obligation to help other people. You can still feed your ego and believe you are the shit or the next AAA thinker, philosopher or artist. These two things doesn't overlap—but it's still annoying.

That is why this research has a copyfarleft license for its content and a copyleft license for its code. Actually, it's the same licensing scheme of this blog.

With Open and Free Publishing License (abbreviated LEAL, that also means "loyal" in Spanish) you are free to use, copy, reedit, modify, share or sell any of this content under the following conditions:

- Anything produced with this content must be under some type of LEAL.



The research in its original MD input.

Markdown

Why Markdown? Because it is:

- easy to read, write and edit
- easy to process
- a lightweight format
- a plain and open format

Markdown format was intended for blog writing. So "vanilla" Markdown isn't enough for research or scholarly writing. And I'm not a fan of Pandoc's Markdown.

Don't get me wrong, Pandoc *is* the Swiss knife for document conversion, its name suits it perfectly. But for the type of publishing I do, Pandoc is part of the

automation process and not for inputs or outputs. I use Pandoc as a middleman for some formats as it helps me save a lot of time.

For inputs and output formats I think Pandoc is a great general purpose tool, but not enough for a fussy publisher like this *perro*. Plus, I love scripting so I prefer to employ my time on that instead of configuring Pandoc's outputs—it helps me learn more. So in this publishing process, Pandoc is used when I haven't resolved something or I'm too lazy to do it, LOL.

Unlike text processing formats as ODT or DOCX, MD is very easy to customize. You don't need to install plugins, rather you just generate more syntax!

So Pecas' Markdown was the base format for the content. The additional syntax was for citing the bibliography by its id.

Bib

Formatting a bibliography is one of the main headaches for many researchers. It requires a lot of time and energy to learn how to quote and cite. And no matter how much experience one may have, the references or the bibliography usually have typos.

I know it by experience. Most of our clients' bibliographies are a huge mess. But 99.99% percent of the time it's because they do it manually… So I decided to avoid that hell.

They are several alternatives for bibliography format-

5. It exports [10] the MD to DOCX [11].
6. It moves the analytics to its correct directory.
7. It refreshes the modification date in the index [07].
8. It puts the new rolling release' hash in the index.
9. It puts the MD5 checksum of all outputs in the index.

And that's it. The process of developing a thesis as a automate repository allows me to just worry about three things:

1. Write the research.
2. Manage the bibliography.
3. Deploy all outputs automatically.

The legal stuff

That's how it works, but we still have to talk about how the thesis can *legally* be used…

This research was paid for by every Mexican through taxation. The National Council of Science and Technology (abbreviated Conacyt) granted me a scholarship to study a Master's in Philosophy at UNAM—yeah, American and British folks, more likely than not, we get paid here for our graduate studies.

This scholarship is a problematic privilege. So the least I can do in return is to liberate everything that was paid for by my homies and give free workshops and advice. I repeat: it is *the least* we can do. I disagree with using this privilege to live a lavish or party lifestyle

Bibliography pathway

Even with a simplified view you can see how this repo is a fucking mess. The bibliography [01] and the thesis [08] are the main directories in this repo. As a sibling you have the website [07].

The bibliography directory isn't part of the automation process. I worked on the BIB file [02] in different moments than my writing. I exported it to HTML [03] every time I used JabRef. This HTML is for queries from the browser. Over there it's also a simple script [04] to clean the bibliography with Biber and the configuration file [05] for Baby Biber. Are you a data hoarder? There is an special directory [06] for you with all the works used for this research ;)

Engine on

In the thesis directory [08] is where everything moves smoothly when you call to `generate-all` [19], the wrapper that turns on the engine!

The wrapper does the following steps:

1. It adds the bibliography [24] to the original MD file [25], leaving a new file [26] to act as input.
2. It generates [21] the HTML output [22].
3. It compiles [28] the PDF output [29].
4. It generates [13] the EPUB [17] and MOBI [18] according to their metadata [16] and Pecas config file [15].

ting and the most common one is Bib, the successor of Bib. With this type of format you can arrange your bibliography as an object notation. Here is a sample of an entry:

```
@book{proudhon1862a,
   author    = {Proudhon, Pierre J.},
   date      = {1862},
   file      = {:recursos/proudhon1862a.pdf:PDF},
   keywords  = {prio2,read},
   publisher = {Office de publicité},
   title     = {Les Majorats littéraires},
   url       = {http://alturl.com/fiubs},
}
```

At the beginning of the entry you indicate its type and id. Each entry has an array of key-value pairs. Depending on the type of reference, there are some mandatory keys. If you need more, you can just add them in. This could be very difficult to edit directly because PDF compilation doesn't tolerate syntax errors. For comfort, you can use some GUI like JabRef. With this software you can easily generate, edit or delete bibliographic entries as if they were rows in a spreadsheet.

So I have two types of input formats: BIB for bibliography and MD for content. I make cross-references by generating some additional syntax that invokes bibliographic entries by their id. It sounds complicated, but for writing purposes it's just something like this:

> @textcite[someone2020a] states… Now I am paraphrasing someone so I would cite her at the end @parencite[someone2020a].

When the bibliography is processed I get something like this:

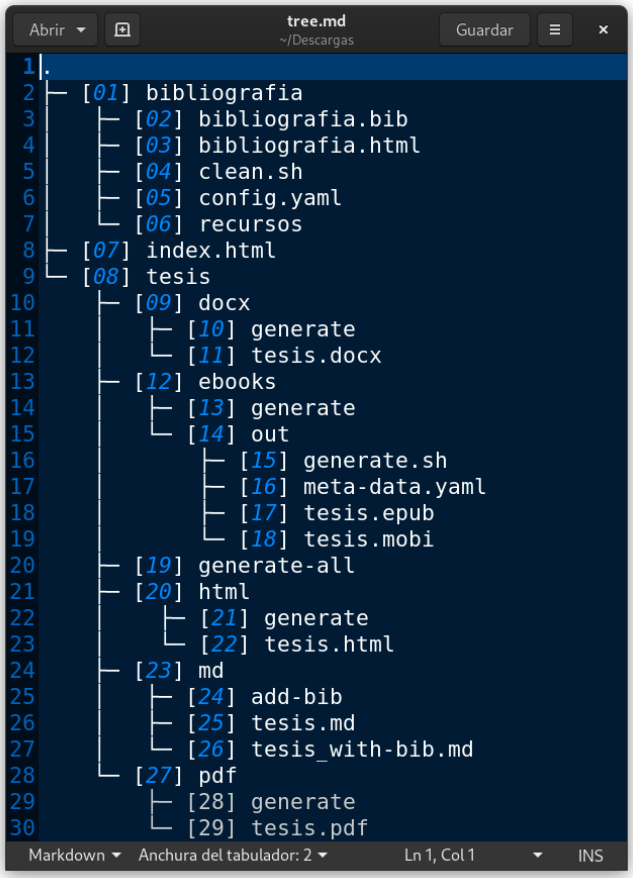
> Someone (2020) states… Now I am paraphrasing someone so I would cite her at the end (Someone, 2020).

This syntax is based on textual and parenthetical citations styles for Bib. The at sign (@) is the character I use at the beginning of any additional syntax for Pecas' Markdown. For processing purposes I could use any other kind of syntax. But for writing and editing tasks I found the at sign to be very accessible and easy to find.

The example was very simple and doesn't fully explore the point of doing this. By using ids:

- I don't have to worry if the bibliographic entries change.
- I don't have to learn any citation style.
- I don't have to write the bibliography section, it is done automatically!
- I *always* get the correct structure.

In a further section I explain how this process is possible. The main idea is that with some scripts these two inputs became one, a Markdown file with an added bibliography, ready for automation processes.



Simplified tree view.

Third parties

I already mentioned the third party software I used for this repo:

- Vim as a main text editor.
- Gedit as a side text editor.
- JabRef as a bibliography manager.
- Pandoc as a document converter.
- Luaas a PDF engine.
- Biber as a bibliography cleaner.

The tools I developed and this software are all FOSS, so you can use them if you want without paying or asking for permission—and without warranty xD

Deployment

There is a fundamental design issue in this research as automated repository: I should have put all the scripts in one place. At the beginning of the research I thought it would be easier to place each script side by side its input or output. Over time I realized that it wasn't a good idea.

The good thing is that there is one script that works as a wrapper. You don't really have to know anything about it. You just write the research in Markdown, fill the Bibbibliography and any time you want or your server is configured, call that script.

This is a simplified listing showing the places of each script, inputs and outputs inside the repo:

Outputs

I hate PDF as the only research output, because most of the time I made a general reading on screen and, if I wanted a more detailed reading, with notes and shit, I prefer to print it. It isn't comfortable to read a PDF on screen and most of the time printed HTML or ebooks are aesthetically unpleasant. That's why I decided to deliver different formats, so readers can pick what they like best.

Seeing how publishing is becoming more and more centralized, unfortunately the deployment of MOBI formats for Kindle readers is recommendable—by the way, FUCK Amazon, they steal from writers and publishers; use Amazon only if the text isn't in another source. I don't like proprietary software as Kindlegen, but it is the only *legal* way to deploy MOBI files. I hope that little by little Kindle readers at least start to hack their devices. Right now Amazon is the shit people use, but remember: if you don't have it, you don't own it. Look what happened with books in Microsoft Store…

What took the cake was a petition from my tutor. He wanted an editable file he could use easily. Long ago Microsoft monopolized ewriting, so the easiest solution is to provide a DOCX file. I would prefer to use ODT format but I have seen how some people don't know how to open it. My tutor isn't part of that group, but for the outputs it's good to think not only in what we need but in what we could need. People barely read research, if it isn't accessible in what they already know, they won't read.

So, the following outputs are:

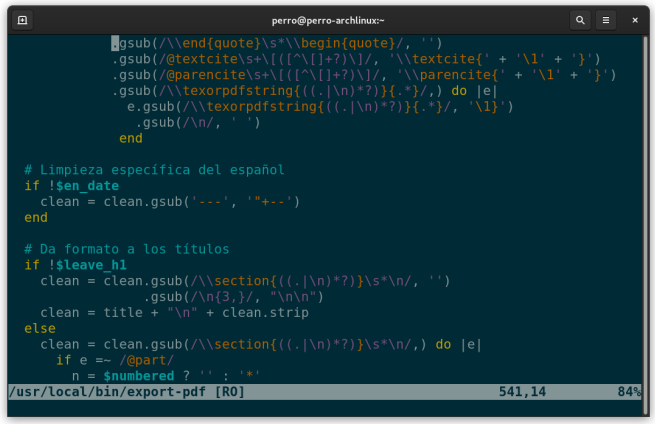
- EPUB as standard ebook format.
- MOBI for Kindle readers.
- PDF for printing.
- HTML for web surfers.
- DOCX as editable file.

Ebooks

I don't use Pandoc for ebooks, instead I use a publishing tool we are developing: Pecas. "Pecas" means "freckles," but in this context it's in honor of a pinto dog from my childhood.

Pecas allows me to deploy EPUB and MOBI formats from MD plus document statistics, file validations and easy metadata handling. Each Pecas project can be heavily customized since it allows Ruby, Python or shell scripts. The main objective behind this is the ability to remake ebooks from recipes. Therefore, the outputs are disposable in order to save space and because you don't need them all the time and shouldn't edit final formats!

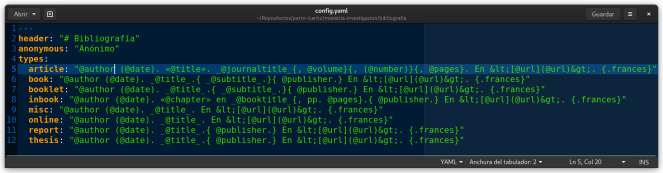
Pecas is rolling release software with GNU General Public License, so it's open, free and *libre* program. For a couple months Pecas has been unmaintained because this year we are going to start all over again, with cleaner code, easier installation and a bunch of new features—I hope, we need your support.



`export-pdf` Ruby code.

still want to experiment with Con.

I think Concould be a useful tool while using XML files for PDF outputs. I defend Markdown as input format for writers and publishers, but for automation XML format is way better. For the new version of Pecas I have been thinking about the possibility of using XML for any kind of standard output like EPUB, PDF or JATS. I have problems with TEX format because it generates an additional format just for one output, why would I allow it if XML can provide me with at least three outputs?



Baby Biber sample config file.

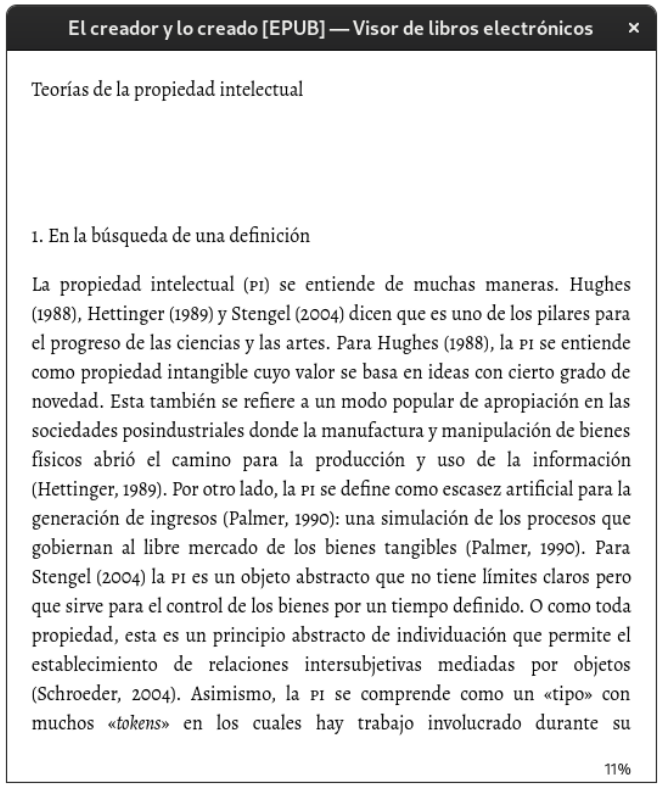
PDF exporter

The last script I wrote is for the automation of PDF compilation with Luaand Biber (optionally).

I don't like the default layouts of Pandoc and I could have read the docs in order to change that behavior, but I decided to experiment a bit. The new version of Pecas will implement PDF outputs, so I wanted to play around a little with the formatting, as I did with Baby Biber. Besides, I needed a quick program for PDF outputs because we publish sometimes fanzines.

So, `export-pdf` is the experiment. It uses Pandoc to convert MD to TEX files. Then it does some clean up and injects the template. Finally, it compiles the PDF with Luaand Biber—if you want to add the bibliographic entries this way. It also exports a PDF booklet with `pdfbook2`, but I don't deploy it in this repo because the PDF is letter size, too large for a booklet.

I have a huge debt here that I won't pay. It is cool to have a program for PDF outputs that I understand, but I



The research in its EPUB output.

PDF

For PDF output I rely on and Lua. Why? Just because it is what I'm used to. I don't have any particular argument against other frameworks or engines inside the family. It's a world I still have to dig more into.

Why don't I use desktop publishing instead, like InDesign or Scribus? Outside of its own workflow, desktop publishing is hard to automate and maintain. This approach is great if you just want a PDF output or if you desire to work with a GUI. For file longevity and automated and multiformat standardized publishing, desktop publishing simply isn't the best option.

Why don't I just export a PDF from the DOCX file? I work in publishing, I still have some respect for my eyes…

Anyway, for this output I use Pandoc as a middleman. I could have managed the conversion from MD to TEX format with scripts, but I was lazy. So, Pandoc converts MD to TEX and Luacompiles it into a PDF. I don't use both programs explicitly, instead I wrote a script in order to automate this process. In a further section I explain this.

HTML

The EPUB format is actually a bunch of compressed HTML files plus metadata and a table of contents. So there is no reason to avoid a HTML output. I already have it by converting the MD with Pecas. I don't think someone is gonna read 27K words in a web browser, but you never know. It could work for a quick look.

clear that this program is still in its baby stages.

What does Baby Biber do?

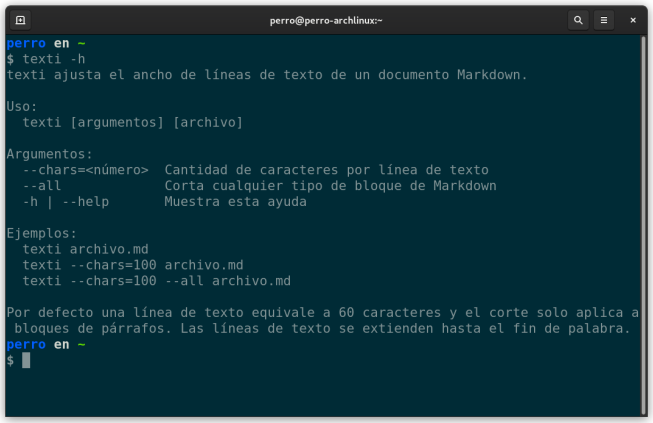
- It generates a new MD file with references and bibliography.
- It adds references if the original MD file calls to `@textcite` or `@parencite` with a correct Bibid.
- It adds the bibliography to the end of the document according to the called references.

One headache with references and bibliography styles is how to customize them. With Pandoc you can use `pandoc-citeproc` which allows you to select any style written in Citation Style Language (CSL). These styles are in XML and it is a serious thing: you should apply these standards. You can check different CSL citation styles in its official repo.

Baby Biber doesn't support CSL! Instead, it uses YAML format for its configuration. This is because of two issues:

1. I didn't take the time to read how to implement CSL citation styles.
2. My University allows me to use any kind of citation style as long as it has uniformity and displays the information in a clear manner.

So, yeah, I have a huge debt here. And maybe it will stay like that. The new version of Pecas will implement and improve the work done by Baby Biber—I hope.



`texti` sample help display.

I started to have issues with the references because our publishing method implies the deployment of outputs in separate processes from the same inputs, in this case MD and BIB formats. With Biber I was able to add the bibliographic entries but only for PDF.

The solution I came to was the addition of references in MD before any other process. In doing this, I merged the inputs in one MD file. This new file is used for the deployment of all the outputs.

This solution implies the use of Biber as a clean up tool and the development of a program that processes bibliographic entries of Bibinside Markdown files. Baby Biber is this program. I wanted to honor Biber and make

The research in its PDF output; I don't like justified text, it's bad for your eyes.

DOCX

This output doesn't have anything special. I didn't customize its styles. I just use Pandoc via another script. Remember, this file is for editing so its layout doesn't really matter.

Writing

Besides the publishing method used in this research, I want to comment on some particularities about the influence of the technical setup over the writing.

Text Editors

I never use word processors, so writing this thesis wasn't an exception. Instead, I prefer to use text editors. Between them I have a particular taste for the most minimalist ones like Vim or Gedit.

Vim is a terminal text editor. I use it on a regular basis—sorry Emacs folks. I write almost everything, including this thesis, with Vim because of its minimalist interface. No fucking buttons, no distractions, just me and the black-screen terminal.

Gedit is a GUI text editor and I use it mainly for RegEx or searches. In this project I utilized it for quick references to the bibliography. I like JabRef as a bibliography manager, but for getting the ids I just need access to the raw BIB file. Gedit was a good companion for that particular job because its lack of "buttonware"—the annoying

These programs are actually Ruby scripts that I put on my `/usr/local/bin` directory. You can do the same, but I wouldn't recommended it. Right now in Programando LIBREros we are refactoring all that shit so they can be shipped as a Ruby gem. So I recommend waiting.

With `texte` I am able to know the number of lines, characters, characters without spaces, words and three different page sizes: by every 1.800 characters with spaces, by every 250 words and an average of both—you can set other lengths for page sizes.

The MD beautifier is `texti`. For the moment it only works well with paragraphs. It was good enough for me, my issue was with the disparate length of lines—yeah, I don't use line wrap.

I also tried to avoid some typical mistakes while using quotation marks or brackets: sometimes we forget to close them. So `textu` is for this quality check.

These three programs were very helpful for my writing, that is why we decided to continue in its development as a Ruby gem. For our work and personal projects, MD is our main format, so we are obligated to provide tools that help writers and publishers also using Markdown.

Baby Biber

If you are into family, you probably know Biber, the bibliography processing program. With Biber we are able to compile bibliographic entries of Bibin PDF outputs and carry out checks or clean ups.

I think the thesis needs more rewriting, I could have written things in a more comprehensive way, but I was done—writing philosophy is not my thing, I prefer to speak or program (!) it. That is why I took my time on the review process—ask my tutor about that, LMFAO. It would have been easier for me to just add footnotes, but it would have been harder for you to read that shit. Besides that, footnotes take more space than rewriting.

So, with respect to the reader and in agreement with the text extension of my university, I decided not to use footnotes.

Programming

As you can see, I had to write some scripts and use third party software in order to have a thesis as an automated repository. It sounds difficult or perhaps like nonsense, but, doesn't Philosophy have that kind of reputation, anyway? >:)

MD Tools

The first challenges I had were:

- I needed to know exactly how many pages I had written.
- I wanted an easier way to beautify MD format.
- I had to make some quality checks in my writing.

Thus, I decided to develop some programs for these tasks: `texte`, `texti` and `textu`, respectively.

tendency to put buttons everywhere.

Citations

I want the research to be as accessible as possible. I didn't want to use a complicated citation style. That's why I only used parenthetical and textual citations.

This could be an issue for many scholars. But when I see typos in their complex citations and quotations, I don't have any empathy. If you are gonna add complexity to your work, the least you can do is to do it right. And let's be honest, most scholars add complexity because they want to make themselves look good—i.e. they conform with formation rules for research texts in order to be part of a community or "gain" some objectivity.

Block Quotations

You are not going to see any block quotes in the research. This isn't only because of accessibility—some people can't distinguish these types of quotes—but the ways in which the bibliography was handled.

One of the main purposes for block quotations is to provide a first and extended hand of what point a writer is making. But sometimes it's also used as text filling. In a common way to do research in Philosophy, the output tends to be a "final" paper. That text is the research plus the bibliography. This format doesn't allow to embed any other files, like papers, websites, books or data bases. If you want to provide some literal information, quotes and

block quotes are the way to go.

Because this thesis is actually an automated repository, it contains all the references used for the research. It has a bibliography, but also each quoted work for backup and educational purposes. Why would I use block quotes if you could easily check the files? Even better, you could use some search function or go over all the data for validation purposes.

Moreover, the university doesn't allow long submission. I agree with that, I think we have other technical capabilities that allow us to be more synthetic. By putting aside block quotes, I had more space for the actual research.

Take it or leave it, research as repository and not as a file gives us more possibilities for accessibility, portability and openness.

Footnotes

Oh, the footnotes! Such a beautiful technique for displaying side text. It works great, it permits metawriting and so on. But it works as expected if the output you are thinking of is, firstly, a file and secondly, a text with fixed layout. In other types of outputs, footnotes can be a nightmare.

I have the conviction that most footnotes can be incorporated into the text. This is due to three personal experiences. During my undergraduate and graduate studies, as a Philosophy student we had to read a lot

of fucking critical editions, which tend to have their "critical" notes as footnotes. For these types of text I get it, people don't want to confuse their words for someone else's, less if it's between a philosophical authority and a contemporary philosopher—take note that it's a personal taste and not a mandate. But this is a shitty Master's research thesis, not a critical edition.

I used to hate footnotes, now I just dislike them. Part of my job is to review, extract and fix other peoples' footnotes. I can bet you that half of the time footnotes aren't properly displayed or they are missing. Commonly this is not a software error. Sometimes it's because people do them manually. But I won't blame publishers nor designers for their mistakes. The way things are developing in publishing, most of the time the issue is the lack of time. We are being pushed to publish books as fast as we can and one of the side effects of that is the loss of quality. Bibliography, footnotes and block quotes are the easiest way to find out how much care has gone into a text.

I do blame some authors for this mess. I repeat, it is just a personal experience, but in my work I have seen that most authors put footnotes in the following situations:

- They want to add shit but not to rewrite shit.
- They aren't very good writers or they are in a hurry, so footnotes are the way to go.
- They think that by adding footnotes, block quotes or references they can "earn" objectivity.