# *Cómo está hecha: tesis de Maestría*

Perro Tuerto

2020

*Cómo está hecha: tesis de Maestría*
Perro Tuerto

Última edición: 16 de febrero del 2020

Este material es una entrada del *blog*
Publishing is Coding: Change My Mind
www.perrotuerto.blog



Hecho en México / *Made in Mexico*

# Cómo está hecha: tesis de Maestría

Uff, después de seis meses de estar escribiendo, revisando, suprimiendo, gritando y casi darme por vencido, por fin he concluido la tesis de investigación de la Maestría. Puedes verla en `maestria.perrotuerto.blog`.

La tesis es sobre propiedad intelectual, bienes comunes y producción cultural y filosófica. La Maestría en Filosofía la realicé en la Universidad Nacional Autónoma de México (UNAM). Esta investigación consiste en aproximadamente veintisiete mil palabras y casi cien páginas.

Desde el principio decidí que no la escribiría usando procesadores de textos como LibreOffice ni Microsoft Office. Esta decisión es debido a que:

- El *software* de ofimática fue diseñado para un tipo de trabajo en particular, no para fines de investigación.
- El manejo bibliográfico y la revisión de la redacción puede ser muy pero muy engorroso.
- Necesitaba diversas salidas, lo que hubiera implicado una labor fuerte de formateo si hubiera escrito la investigación en formatos ODT o DOCX.
- Quería ver qué tan lejos podía llegar con el puro uso

de Markdown, una terminal y FOSS.

De manera general la tesis en realidad es un repositorio automatizado donde puedes verlo todo —incluyendo la bibliografía entera, el sitio y la historia de escritura—. Esta investigación usó un modelo de liberación continua —«un concepto de lanzamiento frecuente de actualizaciones»—. La metodología consiste en edición estandarizada, automatizada y multiformato, o como prefiero denominarla: edición ramificada.

Este no es el espacio para discutir el método, pero estas son algunas ideas generales:

- Tenemos algunos datos de entrada que son nuestros archivos de trabajo.
- Necesitamos diversas salidas que serán nuestros archivos listos para distribuir.
- Queremos automatizar para solo concentrarnos en escribir y editar, en lugar de perder nuestro tiempo con el formato o tener pesadillas con la maquetación.

Para tener éxito es necesario evitar cualquier tipo de enfoques MYSIWYG o de publicación de escritorio. En su lugar, la edición ramificada emplea el enfoque MYSIGYM y sistemas de composición tipográfica.

¡Así que empecemos!

## Datos de entrada

Cuento con dos archivos principales como datos de entrada: el contenido de la investigación y la bibliografía. Para el contenido usé Markdown. Para la bibliografía decidí usar Bib.

### Markdown

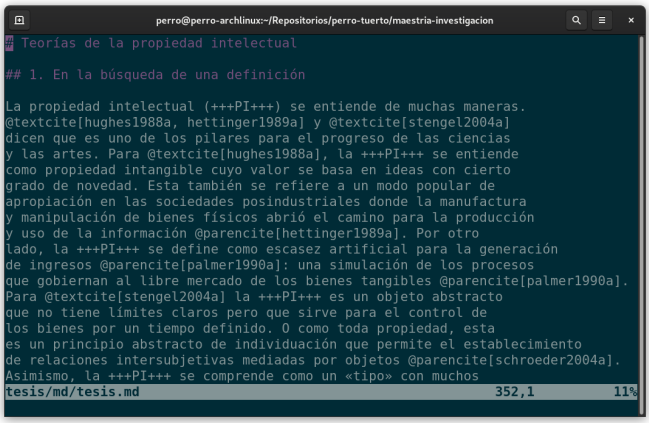
¿Por qué Markdown? Debido a que es:

- fácil de leer, escribir y editar
- fácil de procesar
- un formato ligero
- un formato abierto y de texto plano

El formato Markdown fue planteado para la escritura de *blogs*. Así que la versión «*vanilla*» de Markdown no es suficiente para la escritura de investigación o académica. Además no soy fan de Pandoc Markdown.

No lo tomes a mal, Pandoc *es* la navaja suiza para la conversión de documentos, su nombre le queda a la perfección. Pero para el tipo de edición que llevo a cabo, Pandoc es parte del proceso de automatización y no para los datos de entrada o las salidas. Pandoc lo uso como intermediario para algunos formatos ya que me ayuda a ahorrar mucho tiempo.

Para los datos de entrada y los formatos de salida pienso que Pandoc es una gran herramienta de propósito general, pero no satisface las necesidades de un editor quisquilloso como este perro. Además, amo el *scripting*




La investigación en su dato de entrada original en MD.

así que prefiero emplear mi tiempo en eso en lugar de configurar las salidas de Pandoc —me permite aprender más—. Así que para este proceso de publicación usé Pandoc cuando no había resuelto algo o fui muy flojo para hacerlo, LOL.

A diferencia de los formatos de texto procesado como ODT o DOCX, MD es muy fácil de personalizar. No necesitas instalar *plugins*, ¡solo tienes que generar más sintaxis!

Así que Pecas Markdown fue el formato base para el contenido. La sintaxis adicional fue para citar la bibliografía según su identificador.

Bib

El formateo de bibliografía es uno de los mayores dolores de cabeza para muchos investigadores. El aprendizaje de cómo citar y referenciar requiere de mucho tiempo y energía. Y no importa cuánta experiencia se tenga, las referencias o la bibliografía usualmente tienen erratas.

Lo sé por experiencia. Mucha de la bibliografía de nuestros clientes son un enorme desmadre. Pero el 99.99 % de las veces es debido a que lo hacen de manera manual... Así que decidí evadir ese infierno.

Existen diversas alternativas para el manejo bibliográfico y la más común es Bib, el sucesor de Bib. Con este tipo de formato puedes gestionar tu bibliografía como una notación de objetos. Esta es una muestra de una ficha:

```
@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},
}
```

Al principio de la ficha indicas su tipo y su identificador. Cada una tiene un conjunto de pares llave-valor. Según el tipo de referencia, hay algunas llaves necesa-

rias. Si necesitas más, solo basta con que las añadas. Esto podría ser muy difícil de editar directamente porque la compilación a PDF no tolera errores en la sintaxis. Por comodidad puedes usar una interfaz gráfica como JabRef. Con este *software* de manera muy sencilla puedes generar, editar o eliminar fichas bibliográficas cual si fueran filas en una hoja de cálculo.

Así que tengo dos tipos de formatos para los datos de entrada: BIB para la bibliografía y MD para el contenido. Las referencias cruzadas las llevé a cabo al generar sintaxis adicional que invoca a la ficha bibliográfica a partir de su identificador. Esto suena complicado, pero para fines de redacción es únicamente algo como esto:

> @textcite[alguien2020a] dice… Ahora estoy parafraseando a alguien, así que la citaré al final @parencite[alguien2020a].

Cuando la bibliografía es procesada, tengo algo como esto:

> Alguien (2020) dice… Ahora estoy parafraseando a alguien, así que la citaré al final (Alguien, 2020).

Esta sintaxis está basada en los estilos de citas textuales y parentéticos de para Bib. La arroba (@) es un carácter que empleo al inicio de cualquier sintaxis adicional de Pecas Markdown. Para propósitos de procesamiento podría usar cualquier otro tipo de sintaxis. Pero para las tareas

de redacción y edición me he percatado que la arroba es muy accesible y fácil de localizar.

El ejemplo fue muy sencillo y no demuestra por completo el punto de hacer esto. Al usar identificadores:

- No tengo que preocuparme de que la ficha bibliográfica cambie.
- No tengo que aprender ningún estilo de citas.
- No tengo que escribir la sección de la bibliografía, ¡se genera automáticamente!
- *Siempre* tengo la estructura correcta.

Más adelante explico cómo es posible este proceso. La idea principal es que con un par de *scripts* estos dos datos de entrada se convierten en uno, un archivo Markdown con la bibliografía añadida, listo para el proceso de automatización.

## Archivos de salida

Me molesta que el PDF sea el único archivo de salida para la investigación, la mayoría del tiempo realizo una lectura general en la pantalla y, si quiero ahondar en detalles, con notas y chingaderas, prefiero imprimirla. No es muy cómodo leer un PDF en la pantalla y casi sin excepción la impresión de HTML o de libros electrónicos es estéticamente desagradable. Esos son los motivos por los que decidí proporcionar diferentes formatos para que los lectores puedan escoger el que más le convenga.

A como la edición se está centralizado cada vez más, desafortunadamente es recomendable suministrar el formato MOBI para los lectores con Kindle —por cierto, A LA MIERDA Amazon, le roba a escritores y editores; úsalo solo si el texto no está en otra fuente—. No me agrada el *software* propietario como Kindlegen, pero es el único medio *legal* para proveer archivos MOBI. Ojalá poco a poco los lectores con Kindle al menos empiecen a *hackear* sus dispositivos. Por el momento Amazon es la mierda que usa la gente, pero recuerda: si no lo tienes, no te pertenece. Mira lo que le pasó a los libros en la Microsoft Store…

La cereza del pastel fue una petición de mi tutor. El quería un archivo editable que le fuera fácil de usar. Mucho tiempo atrás Microsoft monopolizó la escritura digital, así que la solución más sencilla fue la distribución de un archivo DOCX. En lo personal hubiera preferido usar el formato ODT pero he visto cuántas personas desconocen cómo abrirlo. Mi tutor no es parte de ese grupo, pero para los archivos de salida es buena idea pensar no solo en lo que necesitamos, sino en lo que podríamos requerir. Las personas a duras penas leen investigaciones, si no es accesible en lo que ya conocen, no leerán nada.

Así que los archivos de salida son:

- EPUB como libro electrónico estándar.
- MOBI para lectores con Kindle.
- PDF para impresión.
- HTML para internautas.

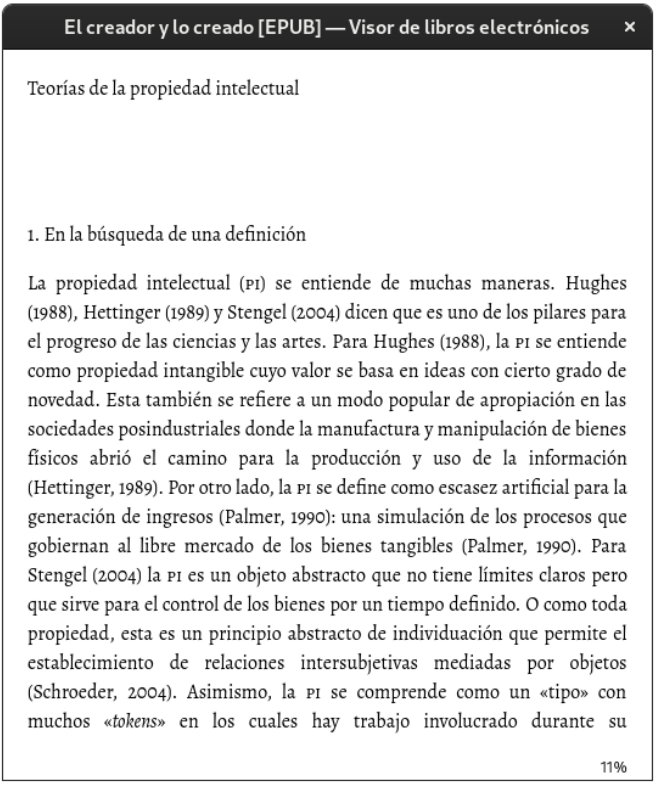
- DOCX como archivo editable.

## Libros electrónicos

No usé Pandoc para los libros electrónicos, en su lugar empleé la herramienta editorial que estamos desarrollando: Pecas. En este contexto «Pecas» es en honor a un perro pintito de mi infancia.

Pecas me permite generar formatos EPUB y MOBI a partir de un MD, además de realizar estadísticas del documento, validación de archivos y manejo sencillo de metadatos. Cada proyecto de Pecas puede ser fuertemente personalizado porque permite *scripts* de Ruby, Python y Shell de Unix. El objetivo principal detrás de ello es la capacidad de rehacer libros electrónicos a partir de recetas. Por lo tanto, los archivos de salida son desechables con el fin de ahorrar espacio y porque ¡no los necesitas todo el tiempo ni deberías hacer ediciones sobre los formatos finales!

Pecas es *software* en liberación continua con Licencia Pública General de GNU, así que es gratuito, abierto y libre. Desde hace meses Pecas no ha estado en mantenimiento porque este año vamos a empezar de nuevo, con código más limpio, con maneras más sencillas de instalarlo y con muchas nuevas características —eso espero, necesitamos tu apoyo—.




La investigación en su salida EPUB.

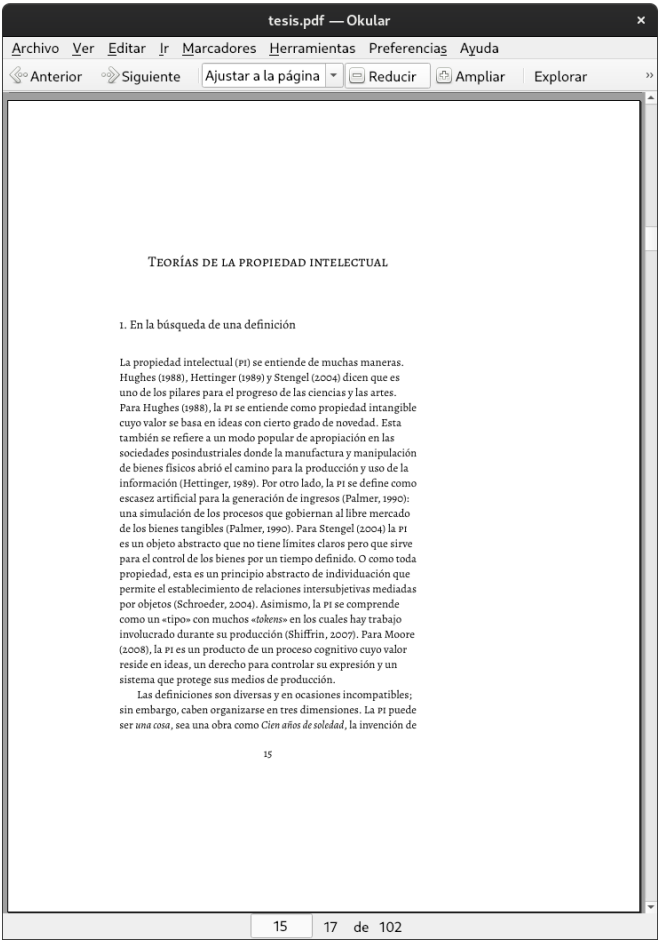
## PDF

Para la salida PDF me fío de y Lua. ¿Por qué? Simplemente por costumbre. No cuento con algún argumento en particular en contra de otros *frameworks* o motores dentro de la familia . Se trata de un mundo que aún tengo que indagar más.

¿Por qué no usé publicación de escritorio en su lugar, como InDesign o Scribus? Afuera de su propio flujo de trabajo, la publicación de escritorio es difícil de automatizar y mantener. Esta aproximación es estupenda si solo quieres una salida PDF o si deseas trabajar con una interfaz gráfica. Para la conservación de los archivos y para una edición estandarizada, automatizada y multiformato, la publicación de escritorio sencillamente no es la mejor opción.

¿Por qué no solo exporté el PDF a partir del archivo DOCX? Mi campo de trabajo es la edición, aún le tengo respeto a mis ojos…

Como sea, para esta salida usé Pandoc como intermediario. Podría haber realizado la conversión de MD a formato TEX con *scripts*, pero fui flojo. Así que Pandoc convierte el MD a TEX y Lualo compila a PDF. No uso ambos programas de manera explícita, en su lugar escribí un *script* que automatiza este proceso. Más adelante explico cómo es posible.




La investigación en su salida PDF; no me agrada el texto justificado, es dañino para nuestros ojos.

### HTML

El formato EPUB en realidad consiste en un conjunto de archivos HTML comprimidos más metadatos y una tabla de contenidos. Así que no hay motivo para evadir una salida HTML. Ya cuento con esta al convertir el MD con Pecas. No creo que alguien vaya a leer casi veintisiete mil palabras en un explorador *web*, pero uno nunca sabe. Este formato podría servir para dar un vistazo.

### DOCX

Esta salida no tiene nada en especial. No personalicé sus estilos. Solo usé Pandoc mediante otro *script*. Recuerda, este archivo es para editar, así que su maquetación no es relevante.

## Redacción

Además del método de publicación empleado en esta investigación, quiero hacer unos comentarios particulares sobre la influencia de la disposición técnica sobre la escritura.

### Editores de texto

Nunca uso procesadores de texto, así que la escritura de esta tesis no fue la excepción. En su lugar prefiero el empleo de editores de texto. Entre estos tengo un gusto particular por los más minimalistas como Vim o Gedit.

Vim es un editor de texto para la terminal. Lo uso de manera regular —lo siento, compas de Emacs—. Casi todo lo escribo con Vim, incluyendo esta tesis, debido a su interfaz minimalista. No hay pinches botones, no tengo distracciones, solo soy yo y una terminal con fondo negro.

Gedit es un editor de texto con interfaz gráfica y principalmente lo uso para expresiones regulares o búsquedas. En este proyecto lo empleé para vistazos rápidos a la bibliografía. Me encanta JabRef como gestor bibliográfico, pero para obtener los identificadores solo necesito acceso directo al archivo BIB. Gedit fue un buen acompañante para esta labor en particular por su carencia de «*buttonware*» —esa fastidiosa tendencia de poner botones por todos lados—.

## Citas

Quiero que la investigación sea lo más accesible posible. No quise usar un sistema complicado de estilos de cita. Por ello únicamente usé citas parentéticas o textuales.

Esto podría ser un problema para varios académicos. Pero cuando veo erratas en sus rebuscadas citas y referencias, no puedo tener ninguna empatía. Si vas a añadir complejidad a tu trabajo, lo mínimo que puedes hacer es ejecutarlo de manera correcta. Y seamos honestos, la mayoría de los académicos agregan complejidad porque quieren verse chingones —es decir, se conforman con las reglas de formación para textos de investigación con la finalidad de ser parte de una comunidad o de «ganar»

algo de objetividad—.

## Bloques de cita

En la investigación no vas a encontrar ni un bloque de cita. Esto no es solo por accesibilidad —algunas personas no pueden distinguir este tipo de citas—, sino también por la manera en como se gestionó la bibliografía.

Uno de los motivos principales para los bloques de cita es el ofrecimiento extendido y de primera mano de lo dicho por un escritor. Pero de manera ocasional también se usan para rellenar cuartillas. En la manera común de hacer filosofía, los archivos de salida tienden a ser un artículo «definitivo». Este texto se compone por la investigación más la bibliografía. Este formato no permite embeber otro tipo de archivos como más artículos, sitios *web*, libros o bases de datos. Si lo que deseas es el suministro de información literal, las citas y los bloques de cita son el medio para llevarlo a cabo.

Debido a que esta tesis en realidad es un repositorio automatizado, contiene todas las referencias usadas para la investigación. Tiene la bibliografía, pero también cada trabajo citado para fines pedagógicos y de conservación. ¿Por qué habría de emplear bloques de cita si fácilmente puedes acceder a los archivos? Aun mejor, podrías utilizar alguna función de búsqueda o ir sobre todos los datos con la intención de validar información.

Además la universidad no permite la entrega de textos amplios. Concuerdo con ello, considero que tenemos

otras capacidades técnicas que nos permiten ser más sintéticos. Al poner a un lado los bloques de cita, tuve más espacio para la investigación.

Tómalo o déjalo, la investigación como repositorio y no como archivo nos da mayores posibilidades de accesibilidad, portabilidad y apertura.

## Notas al pie

¡Oh, las notas al pie! Qué técnica tan más hermosa para mostrar texto secundario. Funciona de manera maravillosa, permite la metaescritura y más. Pero solo funciona como se espera si la salida que estás pensando es, primero que nada, un archivo y, de manera secundaria, un texto con formación fija. Con otro tipo de salidas las notas al pie pueden ser una pesadilla.

Tengo la convicción de que casi todas las notas al pie pueden incorporarse al texto. Esto es por tres experiencias personales. En los estudios universitarios como estudiante de filosofía tenemos que leer un chingo de ediciones críticas, las cuales tienden a implementar sus notas «críticas» al pie. Para este tipo de textos lo entiendo, a las personas no les gusta que confundan sus palabras con las de otro, menos si es entre una autoridad filosófica y un filósofo contemporáneo —tomen nota: es un gusto personal, no un mandato—. Pero esta es una pirruchenta tesis de investigación de Maestría, no una edición crítica.

Solía odiar las notas al pie, ahora solo me desagradan. Parte de mi trabajo consiste en revisar, extraer y enmen-

dar notas al pie de otras personas. Puedo apostar que la mitad de las veces las notas al pie no están presentadas de la manera correcta o están desaparecidas. Por lo general no se trata de un error de *software*. En algunas ocasiones es porque las personas las elaboran de manera manual. Pero no voy a culpar a editores o diseñadores por sus errores. Por el modo en como las cosas se están gestando en la edición, la mayoría de las veces es por falta de tiempo. Nos están presionando para publicar libros tan rápido como podamos y uno de los daños colaterales es la pérdida de calidad. De la manera más sencilla en la bibliografía, las notas al pie y los bloques de cita puede observarse qué tanto cuidado se le ha dado a un texto.

Sí culpo a algunos autores por este desastre. Vuelvo a repetir, esto es solo una experiencia personal; sin embargo, en mi trabajo he visto que la mayoría de los autores colocan notas al pie en las siguientes situaciones:

- Quieren agregar más chingaderas pero no quieren reescribir ni mierda.
- No son buenos escritores o están en apuros, por lo que las notas al pie son el camino a seguir.
- Piensan que por la adición de notas al pie, bloques de cita o referencias podrán «obtener» objetividad.

En mi opinión la tesis necesita más reescritura, pude haber redactado de una manera mas comprensiva, pero ya estaba hasta el colmo —escribir filosofía no es lo mío, prefiero hablarla o programarla (!)—. Por ello me tomé mi tiempo en su revisión —pregúntenle a mi tutor sobre ello,

lmfao—. Para mí hubiera sido más sencillo solo añadir notas al pie, pero para ti habría sido más embrollo leer esa chingadera. Aparte de eso, las notas al pie ocupan más espacio que la reescritura.

Así que por respeto al lector y en acuerdo con la extensión del texto establecido por mi universidad, decidí no usar notas al pie.

## Programación

Como puedes observar, tuve que escribir algunos *scripts* y usar *software* de terceros para tener una tesis como un repositorio automatizado. Se escucha complicado o quizá como un sin-sentido, pero ¿acaso la filosofía no tiene la misma reputación? >:)

### Herramientas MD

Los primeros desafíos que tuve fueron:

- Necesitaba saber con exactitud cuántas páginas llevaba escritas.
- Quería una manera simple de embellecer el formato MD.
- Tenía que hacer controles de calidad en mi redacción.

En consecuencia, decidí desarrollar algunos programas para estas tareas: `texte`, `texti` y `textu`, respectivamente.

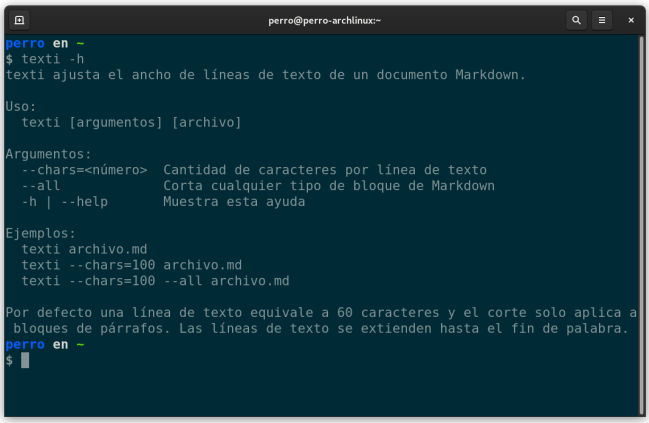
Estos programas en realidad son *scripts* de Ruby que coloqué en mi directorio `/usr/local/bin`. Tú puedes hacer lo mismo pero lo desaconsejo. Ahora mismo en Programando LIBREros estamos refactorizando toda esa basura para que pueda ser despachada como una gema de Ruby. Así que recomendaría que esperaras.

Con `texte` puedo saber cuántas líneas, caracteres, caracteres sin espacios y palabras tiene un archivo, además de conocer tres tipos de tamaños de cuartilla: cada mil ochocientos caracteres con espacios, cada docientas cincuentas palabras y un promedio de ambas —puedes establecer otros tamaños—.

El embellecedor de MD es `texti`. Por el momento solo funciona bien con párrafos. Esto es suficiente para mí porque mi problema fue con las longitudes dispares de líneas —sí, no uso ajuste de línea—.

También intenté evadir típicos errores al usar comillas o paréntesis: en algunas ocasiones olvidamos cerrarlos. Así que `textu` fue para este control de calidad.

Estos tres programas fueron de mucha ayuda para mi escritura, por ello decidimos continuar con su desarrollo como una gema de Ruby. Para nuestro trabajo o proyectos personales, MD es nuestro formato principal, así que tenemos la obligación de proveer de herramientas que ayuden a escritores y editores que también usen Markdown.




Impresión de la ayuda de `texti`.

## Baby Biber

Si conoces la familia , con seguridad sabes de Biber, el programa de procesamiento bibliográfico. Con Biber puedes compilar las fichas bibliográficas de Biben salidas PDF así como hacer verificaciones y limpiezas.

Las referencias empezaron a ser un problema porque nuestro método de publicación implica el desdoblamiento de salidas en procesos independientes desde los mismos datos de entrada, en este caso los formatos MD y BIB. Con Biber puedo añadir las fichas bibliográficas pero solo al PDF.

La solución que llevé a cabo fue la adición de referencias en el MD antes de cualquier otro proceso. Así se unen

los datos de entrada en un archivo MD. Este nuevo fichero se utiliza para despachar todas las salidas.

Esta solución implica el uso de Biber como herramienta de limpieza y el desarrollo de un programa que procese las fichas bibliográficas de Bibdentro de archivos MD. Baby Biber es este programa. Con este nombre quise honrar a Biber y poner en claro que este programa está en sus fases iniciales.

¿Qué hace Baby Biber?

- Genera un nuevo archivo MD con las referencias y la bibliografía.
- Añade las referencias si el MD original llama a `@textcite` o `@parencite` con un identificador correcto de Bib.
- Añade la bibliografía al final del documento según las referencias invocadas.

La personalización de los estilos bibliográficos y de referencias es un dolor de cabeza. Con Pandoc puedes usar `pandoc-citeproc`, lo cual te permite seleccionar cualquier estilo compuesto en Citation Style Language (CSL). Estos estilos están en XML y son de armas tomar: deberías aplicarlo como estándar. Puedes revisar diferentes estilos de citas CSL en su repositorio oficial.

¡Baby Biber no soporta CSL! En su lugar usa el formato YAML para su configuración. Esto se debe a dos cuestiones:

1. No me tomé el tiempo para leer cómo implementar los estilos de cita CSL.


Archivo de configuración de muestra de Baby Biber.

2. Mi universidad me permite usar cualquier tipo de estilo de cita siempre y cuando tenga uniformidad y muestre la información de manera clara.

Así que, sí, aquí tengo una gran deuda. Y es probable que así se quede. La nueva versión de Pecas implementará y mejorará el trabajo hecho por Baby Biber —eso espero—.

## Exportador PDF

El último *script* que escribí fue para automatizar la compilación de PDF con Luay Biber (opcional).

No me gusta la plantilla por defecto de Pandoc y podría haber leído la documentación para cambiar este comportamiento, pero decidí experimentar un poco. La nueva versión de Pecas permitirá salidas PDF así que quise juguetear con el formateo, como lo hice con Baby Biber. Además, necesitaba con urgencia un programa para salidas PDF porque a veces publicamos *fanzines*.

Entonces, `export-pdf` es este experimento. Este usa

Pandoc para convertir archivos MD a TEX. A continuación hace una limpieza e inyecta la plantilla. Por último, compila el PDF con Luay Biber —si quieres añadir las fichas bibliográficas de esta manera—. También exporta un folleto PDF con `pdfbook2`, lo cual no implementé en este repositorio porque el PDF es de tamaño carta, muy grande para un folleto.

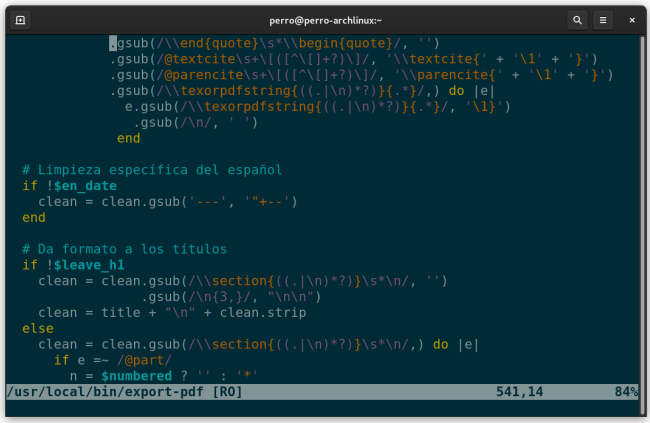
Tengo una enorme deuda que no voy a pagar. Es muy chido tener un programa para salidas PDF cuyo funcionamiento entiendo, pero todavía quiero experimentar con Con.

Pienso que Conpodría ser una herramienta muy útil en el uso de archivos XML para salidas PDF. Mi postura es la defensa de Markdown como formato de entrada para escritores y editores, pero para automatización el XML es superior. Para la nueva versión de Pecas he estado pensando en la posibilidad de usar XML para cualquier tipo de salida estándar como EPUB, PDF o JATS. Mi problema con los archivos TEX es se trata de un formato adicional para una sola salida, ¿por qué lo permitiría si el XML puede suministrarme al menos tres?

### *Software* de terceros

Ya mencioné los programas de terceros que utilizo en este repositorio:

- Vim como editor de texto principal.
- Gedit como editor de texto secundario.
- JabRef como gestor bibliográfico.




Código de Ruby de `export-pdf`.

- Pandoc como conversor de documentos.
- Luacomo motor compilador de PDF.
- Biber como limpiador bibliográfico.

Todas las herramientas que desarrollé y estos programas son FOSS, por lo que puedes usarlos si quieres y sin tener que pagar o pedir permiso —y sin garantía xD—.

## Desarrollo

Hay un problema fundamental de diseño para esta investigación como repositorio automatizado: tuve que haber colocado todos los *scripts* en un solo lugar. Al principio de la investigación pensé que sería más sencillo poner cada

*script* lado a lado a su dato de entrada o archivo de salida. Con el tiempo me di cuenta de que no fue buena idea.

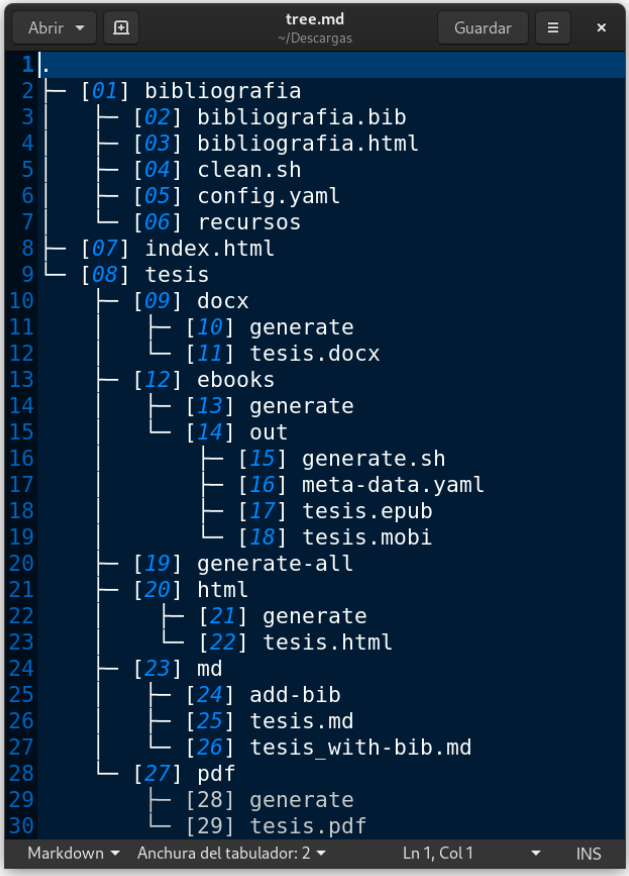
Lo bueno es que hay un *script* que funciona como *wrapper*. En realidad no tienes que saber nada de esto. Únicamente escribes tu investigación en Markdown, llenas tu bibliografía con Biby cada vez que quieras, o tu servidor esté configurado, ejecutas este *script*.

En la página siguiente está una lista simplificada que muestra la ubicación de los *scripts*, sus datos de entrada y sus archivos de salida en el repositorio.

## Senda de la bibliografía

Incluso desde una vista simplificada puedes observar que este repositorio es un desmadre. La bibliografía [01] y la tesis [08] son los directorios principales de este repositorio. Como hermano tienes al sitio [07].

El directorio de la bibliografía no forma parte del proceso de automatización. El archivo BIB [02] lo trabajé en momentos distintos a mi redacción, así como lo exportaba a HTML [03] cada vez que usaba JabRef. Este HTML es para consultas desde el explorador. Ahí mismo hay un simple *script* [04] para limpiar la bibliografía con Biber y el archivo de configuración [05] para Baby Biber. ¿Eres un acumulador de datos? Existe un directorio [06] especial para ti con todos los trabajos usados para esta investigación ;)




Vista en árbol simplificada.

Motor encendido

En el directorio de la tesis [08] es donde todo se mueve plácidamente cuando ejecutas `generate-all` [19], ¡el *wrapper* que pone al motor en funcionamiento!

Este *wrapper* lleva a cabo las siguientes tareas:

1. Añade la bibliografía [24] al archivo MD [25] original, generando un nuevo archivo [26] que funciona como dato de entrada.
2. Genera [21] la salida HTML [22].
3. Compila [28] la salida PDF [29].
4. Genera [13] el EPUB [17] y el MOBI [18] según los metadatos [16] y el archivo de configuración de Pecas [15].
5. Exporta [10] el MD a DOCX [11].
6. Mueve la analítica al directorio correcto.
7. Refresca la fecha de modificación en el index [07].
8. Imprime el *hash* de la versión de la liberación continua en el index.
9. Imprime las sumas de comprobación MD5 de todos los archivos de salida en el index.

Y eso es todo. El proceso de desarrollo de una tesis como repositorio automatizado me permite solo preocuparme por tres cosas:

1. Escribir la investigación.
2. Gestionar la bibliografía.
3. Despachar todas las salidas de manera automatizada.

## Las cuestiones legales

Así es como está hecho, pero todavía tenemos que hablar sobre cómo *legalmente* se puede usar esta tesis…

Esta investigación fue pagada con los impuestos de todos los mexicanos. El Consejo Nacional de Ciencia y Tecnología (Conacyt) me otorgó una beca para estudiar una Maestría en Filosofía en la UNAM —así es, compas de otras latitudes, es común que aquí nos paguen por los estudios de posgrado—.

Esta beca es un privilegio problemático. Así que lo mínimo que puedo dar a cambio es la liberación de todo lo que fue pagado por mis compas, así como dar asesorías y talleres gratuitos. Lo repito: es lo *mínimo* que podemos hacer. Me encuentro en desacuerdo con el empleo de este privilegio para ostentar un estilo de vida abundante o parrandero que culmina con el abandono de los estudios. En un país con tantas crisis, las becas son para mejorar tu comunidad y no solo a ti.

En general tengo la convicción de que si eres un investigador o un estudiante de posgrado y ya recibes un pago —no importa que sea un salario o una beca, que estés en una universidad pública o privada, o que el dinero venga de fondos públicos o privados—, tienes un compromiso con tu comunidad, nuestra especie y nuestro planeta. Si quieres hablar de trabajo gratuito o de explotación —que sí sucede—, por favor mira hacia abajo. En este mundo de mierda estás en los escaños superiores de esta pirámide sin sentido.

Como investigador, científico, filósofo, teórico, artista y más, tienes la obligación de ayudar a otras personas. Aún podrías alimentar tu ego y creer que eres la chingonería o el próximo pensador, filósofo o artista categoría AAA. Ambas cuestiones no se sobreponen —aunque no le quita lo fastidioso—.

Por estos motivos esta investigación tiene licencia *copyfarleft* para su contenido y licencia *copyleft* para su código. En realidad se trata del mismo esquema de licenciamiento de este *blog*.

Con la Licencia Editorial Abierta y Libre (LEAL) eres libre de usar, copiar, reeditar, modificar, distribuir o comercializar bajo las siguientes condiciones:

- Los productos derivados o modificados han de heredar algún tipo de LEAL.
- Los archivos editables y finales habrán de ser de acceso público.
- El contenido no puede implicar difamación, explotación o vigilancia.

Podrías quitar mi nombre y poner el tuyo, está permitido. Incluso podrías modificar el contenido y escribir que AMO la propiedad intelectual: no hay medio técnico que impida semejante difamación. Pero las sumas de comprobación MD5 muestran si los archivos fueron modificados por otros. Aunque el archivo difiera por un bit, la suma de comprobación MD5 arrojará un resultado distinto.

El *copyfarleft* es el camino —mas no la solución— que se ajusta a nuestro contexto y nuestras posibilidades de

libertad. No vengas aquí con tu noción liberal e individualista de libertad —como los weyes de Weblate que expulsaron este *blog* de su servidor porque la licencia de su contenido «no es libre», sin importar que ellos digan que el código, pero no el contenido, debe usar una licencia «libre», como la pinche GPL que emplea este *blog* para su código—. Este tipo de libertad liberal no funciona en lugares donde ningún Estado ni corporación puede garantizarnos un conjunto mínimo de libertades individuales, como acontece en Asia, África y la otra América —América Latina y la América que no es retratada en la publicidad del «sueño americano»—.

## Últimos pensamientos

Así como una tesis funciona a partir de una hipótesis, el sendero técnico y legal de esta investigación actúa según la posibilidad de obtener una tesis como repositorio automatizado, en lugar de una tesis como archivo. Al final esto fue posible, pero de manera limitada.

Pienso que la idea de una tesis como repositorio automatizado es realizable y podría ser una mejor manera de distribuir el conocimiento en lugar de subir un simple archivo. No obstante, esta implementación contiene muchas fugas que la hacen inadecuada para escalarla.

Más trabajo es necesario para poder desdoblarla como práctica estandarizada. Esta técnica también podría ser aplicada para la automatización y la homologación de publicaciones, como artículos en una revista o una colección

de libros. El esfuerzo necesario no es considerable y *tal vez* lo retome durante el doctorado. Pero por ahora, ¡es todo lo que puedo ofrecer!

Gracias a @hacklib por incitarme a escribir esta entrada y, de nueva cuenta, gracias a mi pareja por persuadirme a estudiar la Maestría y por corregir esta publicación en su versión inglesa. Agradezco a mi tutor, a Programando LIBREros y a Gabi por su apoyo académico. No puedo olvidar dar gracias a Colima Hacklab, al Rancho Electrónico y al grupo Miau por su soporte técnico. ¡Gracias también a todas las personas y organizaciones que menciono en la sección de agradecimientos de la investigación!

*Cómo está hecha: tesis de Maestría* se terminó de componer el 16 de febrero del 2020. Documento hecho con LaTeX.