# 表紙の PDF

cover.pdf

こんにちは、@d_etteiu8383 です。この記事は traP 夏のブログリレー 9 月 30 日の記事です。この記事では私のレポート制作環境 (Pandoc と Latexmk を使った PDF 作成の自動化) の紹介をします。既に似た内容の記事が多く存在するため新規性の無い記事ではありますが、皆さんの参考になったら幸いです。

# 目 次

# 1　目的

以下この記事を書こうと思った経緯↓

1. コロナウィルスの影響により、私が所属している系の必修科目である実験科目が延期される
2. 夏季期間講義として (夏休みを削って) 本来のスケジュールを圧縮して週四回の実験科目が開講されることになる
3. 週四回以上のレポート提出を要求される
4. くるしい

普段は Word か LaTeX を用いてレポートを書いていたのですが、Word 文書の見た目を整えるのが苦手だったり、そもそも Word 自体動作が重かったり、LaTeX はコマンド書くのが大変だったりして、レポート 1 つ仕上げるのに非常に時間がかかっていました。そこでレポートの制作環境を見直し、ここにまとめることにしました。手書き、LaTeX 直書き、Word で作る... いろいろ手段はあると思いますが、「そういう方法もあるんだなぁ」程度に読んでもらえると嬉しいです。ここでは最低限講義レポートととして提出できるレベルの PDF ファイルを (なるべく楽して) 出力することを目的とします。

レポートに追われながら書いているので (特に後半) かなり雑な記事になってしまっています。わかりづらい点などありましたらコメントで教えてもらえるとありがたいです。レポート倒したら必ず更新しに来ます。

### 環境

- Windows10
- Pandoc 2.10.1
- pandoc-crossref-Windows-2.10.1
- TeXLive 2020

# 2　原理

以下の手順でレポートを作成していきます。(ほとんど https://blog.8tak4.com/post/168232661994/know-how-writing-thesis-markdown で紹介されている手順を参考にしているのでこちらを見ていただいたほうがわかりやすいかもしれません...)

1. Markdown でレポート本文を書く
2. BibTeX で文献情報をまとめる
3. Pandoc で 1 の Markdown ファイルを tex ファイルに変換する
4. 3 の tex ファイルから Latexmk を使って PDF を作成する
5. (表紙 PDF が指定されている場合)Python で表紙 PDF と結合

以下で各手順の概要を説明します。環境構築や実際に作成するファイルの内容はセクション 3 で紹介します。

## 2.1 Markdown でレポート本文を書く

まず初めに、読み書きが非常に楽な Markdown 記法を用いてレポート本文を作成します。Markdown のコマンドは LaTeX に比べて非常に簡単なので、コマンドミス等に気を取られずレポート本文に集中して執筆ができます。実はこのブログ記事も Markdown 記法を用いて執筆しています。めっちゃ楽。Markdown を使う事のメリットとしては、git diff 等で差分の確認が簡単にできることも挙げられます。MSWord の docx 形式はバイナリ形式なのでテキストに変換しないと diff

具体的なコマンドや Markdown の活用法についてまとまっているこちらの記事もぜひご覧ください→情報系以外の方にもおすすめしたい Markdown

本記事とは関係ない話ですが、弊サークルの部内 SNS traQ[1]ではメッセージ中で Markdown 記法を利用できます。すごい。

## 2.2 BibTeX で文献情報をまとめる

BibTeX とは、LaTeX における参考文献の整形ツールです。`.bib` の拡張子を持つ参考文献の情報を記述したファイルを作成することで、著者名等を自動で整形し、相互参照 (後述) したうえで適切に出力してくれます。

## 2.3 Pandoc で Markdown ファイルを tex ファイルに変換する

Markdown は本来プレーンテキスト形式で書かれた文章から HTML を生成するために開発されたものでした。しかし、現在では HTML 以外の形式のファイルへ変換するソフトも多数存在します。今回はその 1 つである **Pandoc** というコンバータを用いてセクション 2.1 で作成した Markdown ファイルから tex ファイルへの変換を行います。

### 2.3.1 Pandoc の仕組み

そもそも Pandoc は Markdown や LaTeX に限らず、HTML や Word など多種多様なフォーマット間の変換が可能なツールです。対応フォーマットの一覧や、より詳しい使い方については公式ページや、Pandoc ユーザーズガイド日本語版をご覧ください。

Pandoc は以下のような手順で文書を変換しています。(`https://github.com/jgm/pandoc/blob/master/README.md` より引用)

> Pandoc has a modular design: it consists of a set of readers, which parse text in a given format and produce a native representation of the document (an abstract syntax tree or AST), and a set of writers, which convert this native representation into a target format. Thus, adding an input or output format requires only adding a reader or writer.

まず Pandoc は入力された変換前の文書を、**リーダー**というプログラムを用いて**抽象構文木** (abstract syntax tree, AST) と呼ばれる形式に変換します。抽象構文木についておおざっぱに説明すると「もとの文章の意味を保持したまま、その各要素を種類によって区別し (見出しなのか？本文なのか？表なのか？図なのか？など)、いい感じに整理整頓したもの」といった感じ。次に**ライター**というプログラムを用い、この抽象構文木を指定の出力形式に変換します。つまり Pandoc は入力文書を直接出力形式に変換しているのではなく、一度汎用的な形式に変換してから再度目的形式に変換する、という二度の変換を行っています。

[1] traQ について詳しく知りたい方はこちらの記事をご覧ください→爆☆誕 traQ-S【新歓ブログリレー 2020 54 日目】

リーダーとライターを分けているので、新しい形式を利用したい場合は必要なリーダー/ライターを追加するだけで対応できるようになっています。クレバー。


図 1: Pandoc の変換動作例

### 2.3.2 Pandoc-crossref による相互参照

「Pandoc は一度入力ファイルを AST 形式に変換している」と説明しましたが、この AST 形式のドキュメントに手を加えることで変換処理をカスタマイズできる**フィルター**という仕組みが存在します。(以下 https://github.com/jgm/pandoc/blob/master/README.md より引用)

> Users can also run custom pandoc filters to modify the intermediate AST (see the documentation for filters and Lua filters).

具体的には図 2 に示すように、AST 形式の文書を入力として受け取り、AST 形式の文書を出力するプログラムを途中に挟むことで変換処理のカスタマイズを実現しています。こうすることで、入力形式に依存しないカスタマイズと、カスタマイズの容易化を実現しています。スマート。

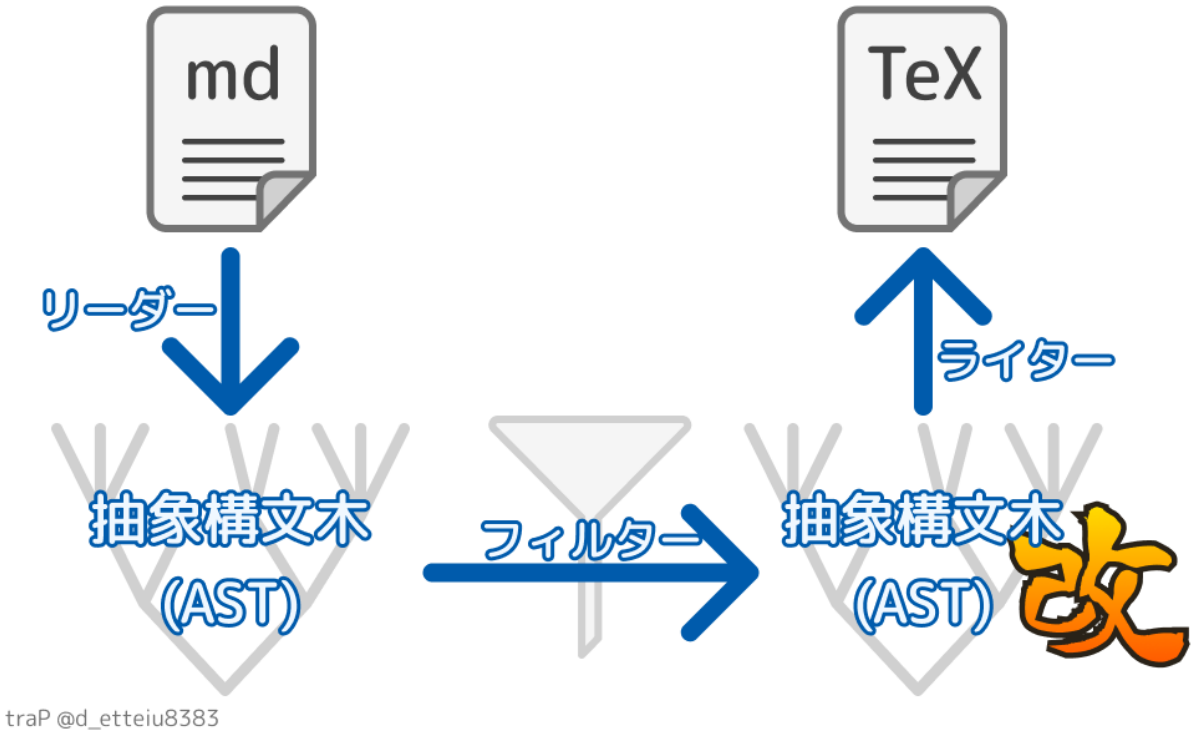


図 2: フィルターの仕組み

フィルターに関する詳細は https://pandoc.org/filters.html をご覧ください。

今回は Pandoc のフィルターの 1 つである **pandoc-crossref** というフィルターを用いて文書内での相互参照ができるようにします。相互参照とは、「図 1 に Pandoc の変換動作例を示す。」の「図 1」のように、図や表、数式、コードブロックの参照番号を自動でつけてくれる機能です。手入力で「図 X に~~を示す。」などと記述してしまうと、編集過程で図やページ順番が変わってしまった場合にいちいち直さなくてはなりません。LaTeX では `\label{}` と `ref{}` を用いることで相互参照が可能ですが、通常の Markdown 形式に相互参照機能は無いため pandoc-crossref を用いた相互参照を行います。

## 2.4 tex ファイルから Latexmk を使って PDF を作成する

次にセクション 2.3 で作成した tex ファイルから Latexmk を利用して PDF を生成します。そもそも tex ファイルから PDF を作成するには、基本的に以下の手順を踏むことになります。

1. tex ファイルから dvi ファイルを作成する
2. dvi ファイルから PDF ファイルを作成する

手順 1 で生成する dvi ファイルには文書のレイアウトに関する情報が記録されており、これをもとに手順 2 で組版を行います。手順 1 を行うために pLaTeX や upLaTeX というエンジンを、手順 2 を行うために dvipdfmx などのソフトを用います。tex ファイルから直接 PDF ファイルを生成する pdfLaTeX、XeLaTeX、LuaLaTeX 等も存在します。どの処理系を使うかは好みの問題になりますが、私は手順 1 に upLaTeX を、手順 2 に dvipdfmx を使用しています[2]。

[2]実は Pandoc 単体でも Markdown 形式の文書から内部で勝手に LaTeX を経由して PDF を生成することもできるのですが、これでは少し融通が利かない部分があったりしたので私は一度 tex ファイルを生成してから PDF を作っています。

と、ここでは最大 2 回の操作で PDF が完成するかのように書いていますが、実際はこの操作を何度か繰り返さないと相互参照が正しく表示されません。これらの操作をいちいち手動で繰り返すのは面倒なので、**Latexmk** で自動化します。Latexmk は、上に述べた PDF 作成までに必要な操作を必要回数自動で行ってくれるツールです。Latexmk についてはこちらのサイトでより詳しく紹介されています→Latexmk から学ぶ PDF 化までの処理の流れ

### 2.5 Python で表紙 PDF と結合

表紙 PDF が指定されている場合は表紙 PDF の結合も行います。ウェブアプリの利用などでも出来ますが、Python のライブラリ **PyPDF2** を使ってサクッと出来たのでこれも自動化しています。参考:Python, PyPDF2 で PDF を結合・分割（ファイル全体・個別ページ）、https://github.com/mstamy2/PyPDF2

## 3 方法

以下に具体的な環境構築・レポート作成手順を示します。(https://github.com/detteiu8383/Markdown2PDF から引っ張ってきていい感じにカスタマイズしてもらえると嬉しい)

### 3.1 各種インストール

#### 3.1.1 Markdown

Markdown ファイルの作成に特別な準備はほとんど必要ありません。一般的なテキストエディタには Markdown のプレビュー機能やショートカットがあることが多いです。私は VSCode か Typora で書いています。Typora は Markdown 専用のエディタで、リアルタイムのプレビュー機能が便利なのでよく使っています。Typora の特徴やインストールは公式サイトを参照してください。

#### 3.1.2 Pandoc のインストール

Pandoc のインストールは Pandoc の比較的簡単なインストール方法で詳しく説明されているので紹介させていただきます。基本的には https://github.com/jgm/pandoc/blob/master/INSTALL.md に示されている手順に従えば大丈夫です。

#### 3.1.3 pandoc-crossref のインストール

こちらも基本的には https://github.com/lierdakil/pandoc-crossref#installation に示されている手順に従えば OK です。リリースページからダウンロードし、パスが通っている所に配置します。

#### 3.1.4 TeXLive(uplatex,dvipdfmx,biber,Latexmk) のインストール

LaTeX の環境構築は TeXLive で行うのが多分楽だと思います。詳しくは TeX Wiki の https://texwiki.texjp.org/?TeX%20Live で説明されているのでこちらを参照してください。

## 3.2 実際の手順

### 3.2.1 ディレクトリ構造

実際に作成するファイルは以下のようにしています。

```
./
|   .latexmkrc    <- Latexmk の設定ファイル
|   build.bat     <- コマンドの自動化
|
├── dest
|       output.pdf <- 生成される PDF
|
└── src
    |   cover.pdf          <- 表紙 PDF
    |   references.bib     <- 文献情報
    |   report.md          <- レポート本文の Markdown
    |
    ├── img
    |       pandoc_description.png <- レポートに挿入する画像
    |       ...
    |
    └── templates
            config.yml     <- Pandoc の設定ファイル
            merger.py      <- 表紙 PDF が指定されている場合、これで PDF の結合をする
            template.tex   <- テンプレートの tex ファイル
```

### 3.2.2 Markdown でレポート本文を書く

残念ながらこの手順は手動です。頑張ってください。今回例として作成した Markdown ファイルは https://github.com/detteiu8383/Markdown2PDF/blob/master/src/report.md にあります。これを report.md として保存。

### 3.2.3 BibTeX で文献情報をまとめる

文献データを作成します。フォーマットについては https://ja.wikipedia.org/wiki/BibTeX で詳しく紹介されています。以下具体例。

```
@book{
  Laala,
  author="プリパラ制作委員会",
  title="プリパラ＆アイドルタイムプリパラ設定資料集＜上＞ ＜プリズムボイス編＋ドリームパレード編＞",
  publisher="小学館",
  year="2019",
  pages="96--97"
}
```

このように、著者名やタイトルをまとめたファイルを作り、本文内で

```
真中らぁらのキャラクターデザインがほぼ決定したのは 2013 年 7 月のことであり、制作初期段階では"ことり"と名づけられていた\cite{Laala}。
```

と書くと、以下のようにコンパイルされます。

真中らぁらのキャラクターデザインがほぼ決定したのは 2013 年 7 月のことであり、制作初期段階では “ことり” と名づけられていた [1]。

最後に書いた `\cite{Laala}` が “[1]” に書き変わっていますね。さらに参考文献欄に自動的に整形された書誌情報が記載されます。`\cite{}` は LaTeX のコマンドですが、Markdown 内で LaTeX のコマンドを使用した場合もちゃんと LaTeX のコマンドとして処理してくれます。

### 3.3 メタデータの記述

Pandoc での変換に関する設定ファイルを作成します。図表を相互参照したとき、デフォルトでは “fig.1” のように参照されてしまうので、これを日本語化するために次のようなファイル `config.yml` を作成し、コマンド実行時に指定します。

```
figureTitle: "図"
tableTitle: "表"
listingTitle: "コード"
figPrefix: "図"
eqnPrefix: "式"
tblPrefix: "表"
lstPrefix: "コード"
secPrefix: "セクション"
```

### 3.4 Pandoc で Markdown ファイルを tex ファイルに変換する

次に上で記述した `report.md` を `main.tex` に変換します。基本的には以下のコマンドを実行することになります。

```
pandoc --filter pandoc-crossref \
  --top-level-division=section \
  -M "crossrefYaml=.\src\templates\config.yml" \
  .\src\report.md -o .\src\main.tex
```

1 行目:Pandoc を使用、フィルターとして pandoc-crossref を用いる。2 行目:トップレベルの見出しを `section`(節) にする。このほか `default`,`chapter`,`part` が使用可能 3 行目:メタデータの設定 4 行目:`report.md` を `main.tex` に変換

これによって生成される `main.tex` はドキュメント部分のみ (LaTeX の `\begin{document}` の中身部分) なので、プリアンブルを記述した `template.tex` 内でこれを input します。

私が使用している `template.tex` は https://github.com/detteiu8383/Markdown2PDF/blob/master/src/templates/template.tex にあるのでご覧ください。

### 3.5 tex ファイルから Latexmk を使って PDF を作成する

Latexmk の詳しい使い方はセクション 2.4 で紹介させていただいたサイトで説明されているのでそちらをご覧ください。私は以下のような.mklatexrc を作っています。

```
#!/usr/bin/env perl
$latex = 'uplatex -halt-on-error';
$latex_silent = 'uplatex -halt-on-error -interaction=batchmode';
$biber = 'biber --bblencoding=utf8 -u -U --output_safechars';
$dvipdf = 'dvipdfmx %O -o %D %S';
$makeindex = 'mendex %O -o %D %S';
$max_repeat = 10;
$pdf_mode = 3;
```

各オプションは https://qiita.com/Rumisbern/items/d9de41823aa46d5f05a8#latexmk で詳しく説明されています。このような設定ファイルを作成することで、面倒なコンパイルが

```
latexmk template.tex
```

とコマンドを実行するだけで終わります。

### 3.6 Python で表紙 PDF と結合

PyPDF2 を pip install PyPDF2 でインストールし、次に示す merger.py を作成します。

```
import sys
import PyPDF2

args = sys.argv

merger = PyPDF2.PdfFileMerger()

for path in args[1:-1]:
    merger.append(path)

merger.write(args[-1])
merger.close()
```

これで

```
python merger.py cover.pdf page1.pdf page2.pdf page3.pdf output.pdf
```

のように実行すれば、1 ページ目から順に cover.pdf、page1.pdf、page2.pdf、page3.pdf が結合された output.pdf が生成されます。

### 3.7　バッチファイルにまとめる

以上の操作を次に示すバッチファイルにまとめ、コマンド実行も自動化します。

```
@echo off
setlocal EnableDelayedExpansion

set PROJECT_DIR=%cd%
mkdir tmp
xcopy /e src tmp
copy .latexmkrc tmp
cd tmp
pandoc --filter pandoc-crossref ^
--top-level-division=section ^
-M "crossrefYaml=templates\config.yml" ^
report.md -o main.tex
move templates\template.tex .\
latexmk template
python .\templates\merger.py cover.pdf template.pdf output.pdf
move output.pdf %PROJECT_DIR%/dest/output.pdf
cd %PROJECT_DIR%
rd /S /Q tmp
endlocal
pause
```

やっていることとしては、

1. 作業用に tmp フォルダを作成し、src フォルダ内のファイルを全部コピー
2. tmp フォルダ内でコンパイル
3. 表紙 PDF が指定されている場合、merger.py で PDF の結合を行う必要なかったら rem でコメントアウトするか消す
4. 生成された output.pdf だけ出力用フォルダの dest に移動して、tmp フォルダは削除

といった感じです。これで dest にできたてほかほかの output.pdf が産まれます。

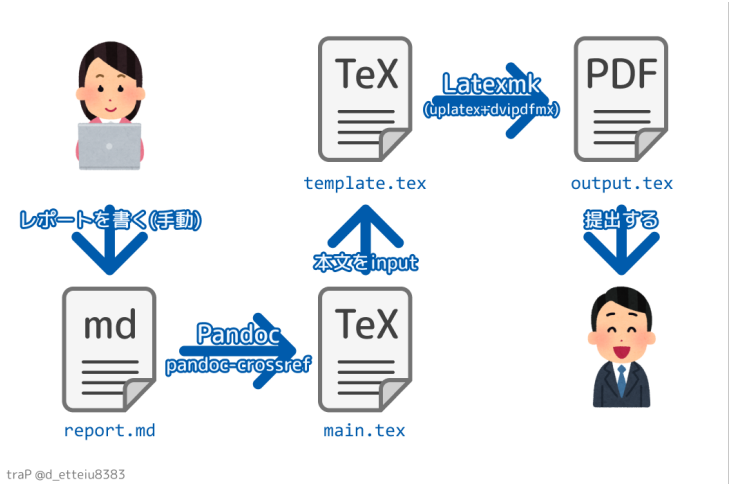


図 3: 作業の流れをまとめるとこんな感じ

## 4 結果

以上の手順を用いて実際に Markdown から PDF に変換したファイルが、今ご覧になっている**この** PDF です。変換元の Markdown は https://github.com/detteiu8383/Markdown2PDF/blob/master/src/report.md でご確認ください。

## 5 考察

- コードブロックのシンタックスハイライトと背景色変更が上手く働いてくれていない
  - LaTeX 力が足りな過ぎて Shaded 環境とか Highlighting 環境がちゃんと動いてない...?レポート終わったら直します。
  - (生命系の学生なのでレポートにコード貼る機会がそもそも無くてちゃんと整備していない)
- 画像横ならべ時の画像感覚が狭い (というか 0)
  - subfig コマンドについてもうちょっと調べて追記します... レポート終わったら

## 6 感想

真のレポート製造機は私だったというオチ。レポート執筆部分がボトルネックになっているので誰かここも自動化してください。

この記事執筆時も未提出レポートが数件溜まっていて本当は記事とか書いている場合じゃないのですがせっかくのブログリレーなので書かせていただきました。未提出レポートが片付いたらもっと丁寧に書き直します。最後まで読んでくださりありがとうございます。

明日の担当は temma さんです。たのしみ〜

## 参考文献

[1] プリパラ制作委員会. **プリパラ＆アイドルタイムプリパラ設定資料集＜上＞ ＜プリズムボイス編＋ドリームパレード編＞**. 小学館, 2019, pp. 96–97.