てふてふ

Sunday, March 19th, 2023

Systems and Control Laboratory, Department of Aeronautics and Astronautics,
Graduate School of Engineering, **Kyoto University**

H.A.*

* GitHub: https://github.com/ha-a

Table of Contents

1. はじめに

本資料は TEX や LATEX，latexmk の中身やコンパイルとはなにかについて，備忘録の意味も込めて記録するものである．インストールの方法などについて解説するものではなく，これを読んだら環境構築で間違わないわけでもなければ，tex が上手に書けるようになるわけでもない．普段気にせずに使っていたけど，よく考えると何も知らなくて，たまに理不尽に言うことを聞いてくれないから怖い “アイツ” について「頭をなでたことがあります」と他人に言っても差し支えないくらいの知識をつけることが目的である*.

TEX や LATEX の区別や使ってはいけない，もしくは強く推奨されないコマンドを紹介しているようなブログや Qiita 記事はたくさん溢れている．当時正しくても今となっては正しくないものも多い．そんなことを念頭に本資料ではなるべく嘘は書かないようにしているつもりである．しかし，私も “学生の中では比較的詳しい方” なだけで，“とても詳しい偉い人” ではないので間違えていることもあるかもしれない．間違いを見つけた場合はお近くの私までご連絡をお願いいたします．また，次節で説明する通り，TEX や LATEX とその周辺は今なお進化を続けている “生きた言語” であり，最新の情報やトレンドによって変化していくものだが，この資料の内容を常に最新情報にするだけの気概はないため，あくまで資料作成時の情報だということについてもご了承願いたい．

2. TEX と LATEX

まず，TEX と LATEX は異なるものである．しかし，そもそも名前が似てるし，両者の区別が曖昧のままでも所望の文書を作成することはとりあえずできてしまうので，別にどっちでもいいという考え方もできる．実際これらの単語を間違えたまま解説しているブログやネット記事も散見される．ここでは TEX と LATEX の何が違うのかを説明し，言葉として TEX がつく様々なものについて，今後混乱・誤用しないようにざっくりと解説していく．

ところで，TEX は TEX であり，tex でも TEX でもなければ Tex でもない．しかし，場合によって TeX と表記されることはある．また，本資料では拡張子が *.tex となっているものについては tex と表記することにする．LATEX も同様で，LATEX かもしくは LaTeX である．ちなみに，TEX の発音はアルファベットの X /ɛks/として読むのではなく，無声軟口蓋摩擦音/x/と発音するのが正しい（諸説あるが）†.

2.1. TEX とはなにか

TEX とは 1978 年に Donald E. Knuth[1] が発表した “組版‡ソフトウェアと組版言語” である[3]．つまり，“文章そのもの” と “文章の構造を指定する命令” が書かれたテキストファイル（この命令を記す言語も TEX という）を読み込み，その命令に従って文章を組版するソフトウェアが TEX である．組版結果は DVI 形式 (DeVice-Independent) に書

*あえて言うならば「抱きかかえる」ことや「素っ裸にする」こともできてはいないことには留意しなくてはならない．

†日本語で表記するなら「てふ」であり，歴史的仮名遣い的にその発音は/tʃoː/である．

‡原稿及びレイアウトの指定に従って，文字・図版・写真などを配置する作業の総称[2].

き出されるが，この dvi ファイルは文書の見た目のレイアウト（紙面のどの位置にどの文字を配置する，などの情報）を画像形式・表示デバイス・プリンタにまったく依存しない形で記録している[4]．TEX をとてもざっくりまとめると（Knuth の思う）美しい組版がデバイスによらず出力される，ということになろう．

2.2. LATEX とはなにか

TEX での命令は「横方向に何センチメートルの空白を作る」のように非常に原始的なものなので，使い勝手が良くない．そこで TEX を使ってもっと簡単に論文やレポートを作成したいという要望から LATEX は開発された[5]．TEX の原始的な命令を組み合わせて新しい命令（マクロと呼ばれる）を作ることができるが，TEX で文書作成をする際には多量のマクロが必要であり，それらの必要なマクロが一通り揃えられたマクロ体系の 1 つが LATEX である．

繰り返しになるが，この世にいくつも存在する TEX を楽に使うためのマクロ体系のひとつが LATEX である．現代の多くの tex ファイルはマクロが準備された LATEX で処理されることが前提であり，それを TEX で処理しようとして目的のものが得られることはまずない．次節で述べる通り，TEX にも LATEX にも沢山の種類があるため，それが tex ファイルが想定するそれと処理する LATEX が食い違っていればエラーが出るのも当然である．他にも，LATEX で定義されたマクロのおかげで成り立つ事象を TEX 全体で成り立つと理解するのは危険なのである．

2.3. その他のﾅﾝﾄｶ TEX とか TEXﾅﾝﾄｶとかﾅﾝﾄｶ TEXﾅﾝﾄｶとはなんなのか

TEX と LATEX 以外にも名前に TEX がつくものは数多くある．違いがわかるように簡単にまとめておこう[6]．

(1) TEX の仲間

組版を行うソフトウェアとして，もともとの TEX の機能が様々に拡張されたものがある．例えば，

- ε-TEX: オリジナルの TEX と 100% の互換性を保ちながら TEX の少々不便なところを補う実用性の高い拡張．
- pTEX: 縦組みに対応し，和文組版できる拡張 TEX．
- ε-pTEX: ε-TEX を取り入れた pTEX．
- upTEX: pTEX の Unicode*版．つまり，pTEX の内部コードを Unicode 化している．例えば Fig. 2.1 では同じ日本語で書かれた tex ファイルを pTEX と upTEX で処理した際の出力の違いである（正確には pTEX に対応したマクロ体系 pLATEX を利用した場合と upLATEX を利用した場合の違い）．pTEX の方では一部の文字（例えば踊り字「〻」や非常用漢字「沆瀣」など）が出力されていないことがわかる．Unicode を用いることでこの世に存在するより幅広い文字を扱えることになる．
- pdfTEX: ε-TEX の拡張．DVI ではなく，PDF を直接出力する．少しだけ触れたが，dvi ファイルとはデバイスに依存しない，TEX 固有の出力形式である．人間が読めるようになっているわけではないため，別に PDF

*様々な言語の文字を単一の文字コードで取り扱うために開発されている文字コード[7],[8]

などに変換する操作が必要である．pdfTEX は出力としてこの `dvi` ではなく，`pdf` ファイルを直接出力するため，ハイパーテキストリンクや目次といった PDF の機能を直接扱うことができる．

- XƎTEX(/ziːtɛx/)[9],[10] : ε-TEX の拡張．符号空間を Unicode 全体に拡大したもの（上述 upTEX の説明参照）．また，OpenType フォントに対応している．これはつまり，OS にインストールされているフォントをそのまま利用することが可能ということである[11]．もともとの TEX では組版を行うとき，組み立てる文字の情報を `tfm` ファイルと呼ばれるものから参照する．したがって好きなフォントを使うためにはそのための `tfm` ファイルを用意する必要があったのだが，XƎTEX ではその必要がなくなる．
- LuaTEX: pdfTEX の後継．OpenType フォント，Unicode に対応している他，Lua という軽量スクリプト言語を利用できる．この Lua の利用によりあらゆるものがカスタマイズ可能となり，究極の柔軟性が得られる[12]．(to-do) Lua を使ったおもちゃをいくつかとフォントの切り替えの楽さの確認．

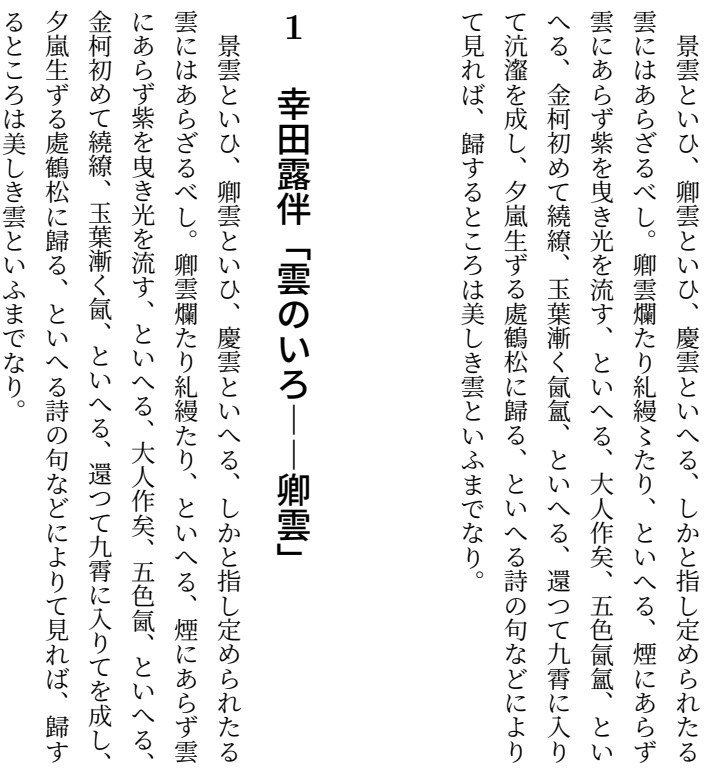
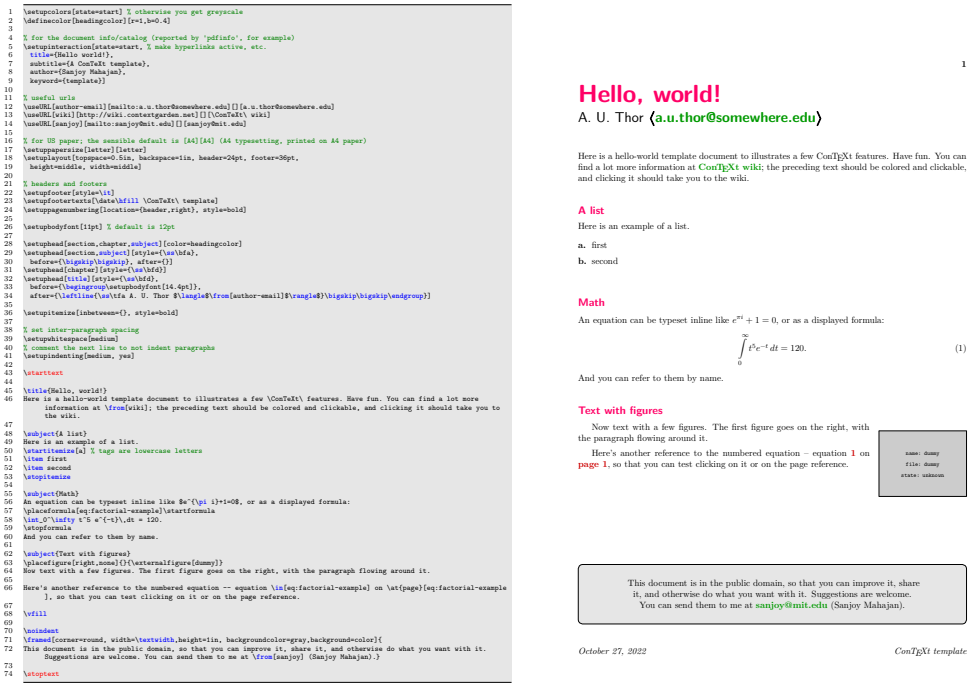


Fig. 2.1. pTEX engine (left) vs upTEX engine (right).[13]

(2) LATEX の仲間

TEX を使いやすくするためのマクロ体系は LATEX 以外にもあって，例えば以下のようなものがある：

- plain TEX: Knuth 自身が開発したもので，組版に最低限必要となる程度の機能しか持たない．
- pLATEX: pTEX に対応した LATEX．
- upLATEX
- pdfLATEX
- XELATEX
- LuaLATEX
- ConTEXt: 文書の体裁もユーザー自身によってすべて調整可能とすることを目標としているマクロ体系[14]．外部パッケージなしであらゆる部分の調整が可能となるようになっているため，LATEX で起こるようなパッケージ同士の衝突などはおこらない．Fig. 2.2 に ConTEXt を使った `tex` ファイルの例とその出力 `pdf` ファイルを示す．左のコードの中で赤く示した命令の間が本文であり，その前がいわゆるプリアンブルである．比較的単純な出力に対し，パッケージ等を用いず，すべての設定をユーザー自身が行う必要があり，プリアンブルが長くなっている印象である．



```
\setupcolors[state=start] % otherwise you get greyscale
\definecolor[headingcolor][r=1,b=0.4]

% for the document info/catalog (reported by 'pdfinfo', for example)
\setupinteraction[state=start, % make hyperlinks active, etc.
  title={Hello world!},
  subtitle={A ConTeXt template},
  author={Sanjoy Mahajan},
  keyword={template}]

% useful urls
\useURL[author-email][mailto:a.u.thor@somewhere.edu][][a.u.thor@somewhere.edu]
\useURL[wiki][http://wiki.contextgarden.net][][\ConTeXt\ wiki]
\useURL[sanjoy][mailto:sanjoy@mit.edu][][sanjoy@mit.edu]

% for US paper; the sensible default is [A4][A4] (A4 typesetting, printed on A4 paper)
\setuppapersize[letter][letter]
\setuplayout[topspace=0.5in, backspace=1in, header=24pt, footer=36pt,
  height=middle, width=middle]

% headers and footers
\setupfooter[style=\it]
\setupfootertexts[\date\hfill \ConTeXt\ template]
\setuppagenumbering[location={header,right}, style=bold]

\setupbodyfont[11pt] % default is 12pt

\setuphead[section,chapter,subject][color=headingcolor]
\setuphead[section,subject][style={\ss\bfa},
  before={\bigskip\bigskip}, after={}]
\setuphead[chapter][style={\ss\bfd}]
\setuphead[title][style={\ss\bfd},
  before={\begingroup\setupbodyfont[14.4pt]},
  after={\leftline{\ss\tfa A. U. Thor $\langle$\from[author-email]$\rangle$}\bigskip\bigskip\endgroup}]

\setupitemize[inbetween={}, style=bold]

% set inter-paragraph spacing
\setupwhitespace[medium]
% comment the next line to not indent paragraphs
\setupindenting[medium, yes]

\starttext

\title{Hello, world!}
Here is a hello-world template document to illustrates a few \ConTeXt\ features. Have fun. You can find a lot more
    information at \from[wiki]; the preceding text should be colored and clickable, and clicking it should take you to
    the wiki.

\subject{A list}
Here is an example of a list.
\startitemize[a] % tags are lowercase letters
\item first
\item second
\stopitemize

\subject{Math}
An equation can be typeset inline like $e^{\pi i}+1=0$, or as a displayed formula:
\placeformula[eq:factorial-example]\startformula
\int_0^\infty t^5 e^{-t}\,dt = 120.
\stopformula
And you can refer to them by name.

\subject{Text with figures}
\placefigure[right,none]{}{\externalfigure[dummy]}
Now text with a few figures. The first figure goes on the right, with the paragraph flowing around it.

Here's another reference to the numbered equation -- equation \in[eq:factorial-example] on \at{page}[eq:factorial-example
    ], so that you can test clicking on it or on the page reference.

\vfill

\noindent
\framed[corner=round, width=\textwidth,height=1in, backgroundcolor=gray,background=color]{
This document is in the public domain, so that you can improve it, share it, and otherwise do what you want with it.
    Suggestions are welcome. You can send them to me at \from[sanjoy] (Sanjoy Mahajan).}

\stoptext
```


Hello, world!

A. U. Thor ⟨a.u.thor@somewhere.edu⟩

Here is a hello-world template document to illustrates a few ConTEXt features. Have fun. You can find a lot more information at ConTEXt wiki; the preceding text should be colored and clickable, and clicking it should take you to the wiki.

A list

Here is an example of a list.

a. first

b. second

Math

An equation can be typeset inline like $e^{\pi i}+1=0$, or as a displayed formula:

$$\int_0^\infty t^5 e^{-t}\,dt = 120. \quad (1)$$

And you can refer to them by name.

Text with figures

Now text with a few figures. The first figure goes on the right, with the paragraph flowing around it.

Here's another reference to the numbered equation – equation 1 on page 1, so that you can test clicking on it or on the page reference.

name: dummy
file: dummy
state: unknown

This document is in the public domain, so that you can improve it, share it, and otherwise do what you want with it. Suggestions are welcome.
You can send them to me at sanjoy@mit.edu (Sanjoy Mahajan).

October 27, 2022 *ConTEXt template*

Fig. 2.2. Example of ConTEXt (left: source tex file, right: pdf file)

少し話は変わるが，LATEX のバージョンについても名前がついている：

- LATEX 2_ε: 現在（2023-03-19）の主流．
- LATEX2.09: 古いバージョン．

- LaTeX3: 開発中の次期バージョン．

(3) 文献管理ソフト

thebibliography 環境でひとつずつ文献情報を入力することも可能だが，数が増えると大変である上，表記に一貫性を持つための調整も手間である．そんなときに，LaTeX と組み合わせて文献データベースから自動的に参考文献リストを作るためのツールが文献管理ソフトである．よく耳にするのは

- BibTeX: 書誌情報のデータベースを用いて，自動で参考文献リストを作成する．このデータベース bib ファイルのなかで各書誌情報はまずエントリごとに種別分けされ（@article, book, proceedings, etc.），それぞれの情報を保存する（title, author, journal, etc.）．
- pBibTeX: pTeX に特化．日本語の文献を扱える．
- upBibTeX

などがあるが，最近は他にも

- BibLaTeX(+Biber) : LaTeX で用いるパッケージ．そもそも BibTeX は参考文献のフォーマットは bst ファイルというもので設定されるのだが，これまた編集が大変である．そのため，よりカスタマイズが容易なのが，本パッケージである．BibLaTeX のバックエンド（文献のソートを担当）として biber か bibtex を用いることになるが，biber のほうが bibtex を用いるよりさらにカスタマイズの柔軟性が上がる[15]．

が次世代文献管理ソフトとして挙げられる．

なお後述するが，文献情報がある際に LaTeX と BibTeX の処理の順番・回数は文献情報がない場合とは異なる．

(4) 統合開発環境

TEX 周りの作業がしやすいように整備されたエディタ．これさえあれば TEX がつかえる，というものではない．

- TEXShop: macOS 専用の LaTeX 統合環境
- TEXworks: TEXShop をモデルにすべてのシステム上で実現すべく開発された LaTeX 用統合環境
- TEXstudio

(5) ディストリビューション

TEX がまともに使えるようになるためには TEX さえあればいい，わけではない．TEX なのか pTEX なのか LuaTEX なのか．LaTeX なのか ConTEXt なのか．BibTeX なのか Biber なのか．統合開発環境はつかうのか．使うならどれか．さらに言えば，自分の文書作成では LuaLaTeX のみ使うと決心したとしても，学会によっては upLaTeX で動くことのみ想定したフォーマットを配っている場合もあり，そのフォーマットを LuaLaTeX 用に書き換えるのはただ面倒で生産性があまりなく，こうなると最初から upLaTeX を準備しておくのが望ましい．これ以外にも，特定のマクロ体系

を更に拡張するための “パッケージ” が必要なことがほとんどで，それらも新たに必要になるたびインストールしなくてはならない．結局，TEX を使えるようになるためには非常に多くのものをインストールしなくてはならないが，一つ一つ個別にインストールするのは大変であり，これらを一括でインストールできるのがディストリビューションである．

日本国内で多く用いられているディストリビューションは

- TEXLive[16]

であるが，他にも

- MacTEX: TEXLive をベースにした macOS 専用のディストリビューション．
- BasicTEX: LATEX の文書作成に必要最低限なものが含まれる macOS 専用ディストリビューション．ε-pTEX などが含まれない．
- MiKTEX: Windows，macOS，Linux で動作するディストリビューション．日本語 pTEX が含まれない．

などが存在する

3. 再三のコンパイルと自動化

ここまで tex ファイルから文書出力を得ることを曖昧に “処理” と表現してきた．この “処理” とは実際にはどのような “処理” をおこなっているのかについて説明していく．具体的には，pdfTEX 系なのか否かや相互参照・文献情報の有無によって必要な処理の種類・順番や回数が変わる．最近では latexmk というコマンドのおかげで，このことを意識しなくてもいいことが多いが，この処理について知っておくことは latexmk 丸投げよりも効率がいい処理も可能で，デバッグの役にも立つので，知っておいて損はないと思う．なお，latexmk という名前からも分かる通り，今後の話は LATEX まわりの話である．

3.1. tex ファイルから pdf ファイルまで

まず latexmk に丸投げして普段考えない，tex ファイルから pdf ファイルを出力するまでの処理の流れについて簡単に説明する[17]．なお，以下では作成した tex ファイルの名前を hoge.tex とする．ユーザは hoge.tex の中身に従い，適切なマクロ体系を選択する．というよりも，日本語を使うのか，Unicode を使うのか，ユーザ好みのフォントを選びたいのか，といった目的に応じて選択すべきマクロ体系があり，それに合わせて hoge.tex を作成する．今後，例として LuaLATEX+upBIBTEX を用いることにする．

(1) はじめての LATEX コンパイル

選択したマクロ体系で処理をする:

```
lualatex hoge.tex
```

これにより，次のファイルが生成される:

- 出力ファイル
  - `hoge.dvi`: LaTeX, (u)pLaTeX などを用いた場合
  - `hoge.pdf`: pdfLaTeX, XeLaTeX, LuaLaTeX などを用いた場合
- 補助ファイル `hoge.aux`: 相互参照・ラベルの情報を保存する．
- ログファイル `hoge.log`: コンパイル時の記録．エラーや警告も保存される．

このとき，もともと相互参照に関する `hoge.aux` を読み込まずに `hoge.tex` のコンパイルを行ったため，今回の出力で適切な相互参照はできていない．相互参照されるべきところでは “??” が表示され，`hoge.log` には参照が見つからなかった旨の警告*が記録される．これは一度の処理を高速・省メモリで終えるために，1 回あたりのコンパイルでは `tex` ファイルを最初から最後まで順に読み込みながら出力を行っているからである．このコンパイルの最中に補助ファイル（`hoge.aux`）にラベルを前から順に保存していく．

次のステップは文書の中に相互参照が含まれるかどうかによって以下のように場合分けされる:

	次の処理
参照したい文献`\cite` がある場合	(2) 参考文献処理
参考文献以外の相互参照`\ref` のみある場合	(3) 相互参照を処理する LaTeX コンパイル
相互参照も参考文献もない場合	(5) 文書の pdf 出力を得る

(2) 参考文献処理

所望の形に整形された参考文献の一覧を作る:

```
upbibtex hoge.aux
```

具体的には，(1) はじめての LaTeX コンパイルによって `hoge.aux` ファイル内に保存された文献リストの雛形から所望の形の文献リストが `hoge.bbl` に生成される[18]．これは，`thebibliography` 環境であり，BibTeX 等を使わずに手動で`\bibitem` などとちまちま書いていく “アレ” である．つまり，`hoge.bbl` にはそれぞれの情報を含めた文献項目（`bibitem` とか `entry`）が所望の形式で出力できるように収められている．

次のステップは (3) 文献リストを出力する LaTeX コンパイル．

(3) 文献リストを出力する LaTeX コンパイル

`hoge.bbl` を使って，文献リストを出力しつつ，補助ファイル `hoge.aux` に文献に関わる相互参照情報を保存する．

*LaTeX Warning: There were undefined references.

```
lualatex hoge.tex
```

ちなみに，(1) で作られた相互参照は今回の LaTeX コンパイル時に読み込まれた hoge.aux を使って，解決される．なぜならこのコンパイル時にはすでに 1 回目のコンパイルで作成された hoge.aux があり，そこに文献参照以外の必要な相互参照の情報が保存されているからである．

次のステップは (4) 相互参照を処理する LaTeX コンパイル．

(4) 相互参照を処理する LaTeX コンパイル

hoge.aux の相互参照を解決し，hoge.dvi か hoge.pdf を出力する:

```
lualatex hoge.tex
```

補助 aux ファイルの中に前回保存したラベルがあるため，その番号を採用・出力していく．御存知の通り，出力が dvi ファイルになるか pdf ファイルになるかはマクロ体系として何を使うかに依存する．直接 pdf ファイルが出力される系統を用いているのであれば，ここで文書は完成である．

マクロ体系	出力	次の処理
LaTeX, (u)pLaTeX	hoge.dvi	(5) 文書の pdf 出力を得る
pdfLaTeX, XeLaTeX, LuaLaTeX	hoge.pdf	-

(5) 文書の pdf 出力を得る

すでに述べたが，dvi ファイルとは DeVice-Independent file のことで，文書の見た目のレイアウトを画像形式・表示デバイス・プリンタなどにまったく依存しない形で記録している[19]．これは人間が読むように設計されておらず，DVI ドライバと呼ばれる別のプログラムに入力することで，画像として表示したり，文書形式に変換したりできる．主な DVI ドライバとしては次のようなものがある:

- dvipdfmx: dvi→pdf．日本語文書作成においては最もよく使われている．そもそも海外では pdfLaTeX とその周辺が主流であり，dvi ファイルは生成されないのである．
- dvips: dvi→ps (PostScript)．この ps ファイルとは pdf ファイルの基礎となったものである[20]．
- dviout: (u)pTeX 対応の DVI プレビューア．最近では推奨されない．

特別なスープ

文書作成の具体的な流れはここまでで説明したとおりである．基本的な相互参照にしろ，文献の参照にしろ，ラベルを適切に補助 aux ファイルに保存できれば，その後もう一回コンパイルすることですべての参照がきちんと対応して出力される．つまり，文献参照が必要ない（もしくは bbl ファイルに変更がない）場合には 2 回のコンパイルによって出力が得られることになる．これは基本的に間違ってはいないが，実際には 2 回では済まない場合がある．

具体的には，参照したいラベル先が不明でとりあえず “??” と出力されていた部分に，ラベルがついている数字や文献を表す記号などが正しく代入されることで改ページの場所や図表の位置が変わってしまった場合，ラベルと番号の対応付けが変わってしまい，参照番号が正しく振られていないことになってしまうことがある*．そこで次のような収束しないおもちゃを作ることができる[21]:

```
\documentclass{article}
\pagenumbering{roman}
\setlength{\textwidth}{5em}
\setlength{\textheight}{2\baselineskip}
\begin{document}
  \setcounter{page}{999}
  \noindent The Page Number \pageref{foo} is\label{foo} shaking up and down.
\end{document}
```

この TEX 文書をコンパイルした結果を Fig. 3.1 に示す．$n \in \mathbb{Z}_+$ 回コンパイルしたあとの出力において，ラベルが付けられる “is 直後” のページ番号とそれを参照しているはずの`\pageref{foo}`の出力は必ず一致しない．このおもちゃに面白い以上の意味は見いだせないが，これが理解できれば，相互参照のために `aux` ファイルが必要で，同様に複数回のコンパイルが必要なこともわかるであろう．

number of compile	page no. which contains `\label`	output of `\pageref`
1	999(cmxcix)	??
even	1000(m)	cmxcix
odd	999(cmxcix)	m

	former	latter
odd	The Page Number cmxcix cmxcix	is shaking up and down. m
even	The Page Number m is shak- cmxcix	ing up and down. m

Fig. 3.1. Example of not converging cross-referencing. Upper: former (left) and latter (right) page of the output pdf file after odd times compile. Lower: output after even times compile.

*LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.

3.2. 自動化

ここまででわかるように，文献を含む相互参照をきちんと機能させるためには，補助ファイル `hoge.aux` の存在も，それを使って複数回コンパイルすることも必要不可欠である．一方で，そのコンパイル回数はある程度目処がつくとは言え，正確に予測することは困難である．このような状況で，変更のたびに参照が失敗している箇所がないか確認しながら自分の手でコンパイルしていくのは大変面倒である．

この「複数回コンパイル等に起因する煩雑さ」を解決・軽減する方法として有名なのはもちろん

- `latexmk`: 現在最も有名な自動化ツール *．複数回の実行に加え，BibTeX 等の関連ツールの実行も自動で行える．`latexmk` は `aux` ファイルの内容が変化したときにコンパイルし直すことになっている．例えば `aux` ファイルに時刻の情報を含むようなコードを書けば，コンパイルのたびに `aux` ファイルが変化することになる†．

であるが，オリジナルの TeX の拡張が様々存在することや，TeX を使いやすくするためのマクロ体系も何種類も存在したことなどを考えれば，この自動化ツールが何種類も存在することは至極自然なことに感じられる．いくつかのコンパイルツールを下記に示すが，「煩雑だな」と感じるのは同じでもそれを解消するための設計指針や思想は異なっているものも多い．

- `latexn`:
- `ptex2pdf`: `(u)platex` と `dvipdfmx` をまとめて実行する．
- `llmk`: Light Latex MaKe.

4. 結局なにをどうするのか

ここまでで，TeX があって，その拡張としていくつか種類があり，TeX たちを楽に使うための LaTeX も何種類か存在することがわかった．説明はしていないが，文書には文書クラスというものがあり（`jsarticle` とか `jsbook` とか），適宜選ばなくてはならない．文献管理の方法も一つではなく，コンパイルを自動化してくれるツールも一つシェア率が高いものがあるとは言え，他にも選択肢がある．もはや説明していないが，エディタも何種類も存在する．もちろん，日本語を扱えないものや明らかに目的が違うものなどもあるため，その中から選択していく必要はあるが，それでも選択肢が何個もあるのは確かである．では我々はなにを選択するべきなのか．

結論

LuaLaTeX + upBibTeX (or BibLaTeX) + `latexmk`

これは結局カスタマイズ性を優先している方法である．その中で個人的には一番「特別なにも考えなくてもいいレ

*個人的には latexmk が広く使われる理由として「煩雑な操作に対して特別何も考えなくてもいつでもどこでも大体なんとなくうまくいく」ことが挙げられると思っている．しかしだからこそ冗長性を持ち，無駄が多いのではないかと感じる部分もある．

†latexmk では上限回数が設定されており，これにより無限の彼方に飛ばされることはない[22]．

シピ」である．LuaLaTeX を使う理由としては

- Unicode に対応しているので，出力できない文字を気にしなくて良い．
- `dvi` ファイルという TeX 固有のよくわからないものを生成する必要もない．

などが挙げられるが，さらにもう少し TeX を自在に扱いたいと思ったとして

- OpenType 対応なので，フォントの設定が楽
- 扱いが難しいとされる原始的な TeX 言語を扱わなくても Lua によって柔軟なカスタマイズが可能．

などもメリットであろう[23]．一方で，デメリットとして LuaLaTeX は処理が遅いことが挙げられるが，以前は 20 倍程度遅かったところ，最近では 2 倍程度に抑えられており[24]，今後さらに速くなっていくと考えられる．

逆に (u)pLaTeX を使わない理由としては，欧文と和文でフォントの扱いが違うためにせっかくの文書が美しくならなかったり，近いうちにまともに動かなくなるなんて話もあったり[25]*するからである．

一方で upBibTeX を用いるのは簡単に

- 学会であれば基本的に参考文献のフォーマットは決まっている（`bst` ファイルが配布される）．
- 自分で参考文献のところを必要以上に凝りたいと思うことが少ない．

ために，比較的処理の遅い BibLaTeX を用いる必要性を感じないからである．しかし，せっかくなら細かいところまで気にしたい，のであれば BibLaTeX を用いるのもよいだろう．

先に述べたコンパイルの煩雑さを回避するためにも自動化は必要不可欠であり，結局広く使われている `latexmk` をここではおすすめしている．というのも

- 広く使われているからこそ，使い方などの情報が豊富に見つかる．
- Visual Studio Code などで公式にサポートされている†．
- 一度設定ファイル `latexmkrc` を作ってしまえば，なにも考えずにその後使いまわすこともできる．

からである．

一方で，もしコンパイルの速さを優先するなら，

upLaTeX + upBibTeX + `dvipdfmx` (+ `latexmk`)

なども考えられよう．この場合，いつでも使える `latexmk` で自動化せずに，必要な回数コンパイルするレシピを作って処理するのがより速いだろう．例えばただの授業の課題など，体裁にこだわる必要のない，たかだか数ページのものを作るのならコンパイルの速度を優先するのがいい．ただ，どうせコンパイル中も編集は行えることを加味

*おそらく，結局誰かがどうにかして実際に使えなくなることはないのだろうが，それでもそのようなリスクを抱えるものをわざわざ使う理由は特にない．

†実は VScode 上でのエラーや警告の拾い方は latexmk だけ特別扱いされる[26]．

すれば LuaLaTeX を利用するのがいいように思える．
「ぼくがかんがえたさいきょうのてふかんきょう」を巻末に記す（予定な）ので，参考にしてほしい．

5. ちょっとした Tips

ここまでで，基本的な TeX の実行について幾分イメージがつきやすくなったと思う．これより先は「へえ、そうだったのか」と個人的に思ったような TeX 周りの何事か（学会原稿周辺のルールなども含む）を完全に備忘録として記していく．

5.1. LuaLaTeX でなんとやら

(1) フォント

本資料でもフォントの調整は行っている．このように途中で変えるのもお手の物である．

References

[1] “Don Knuth’s home page.” URL: https://www-cs-faculty.stanford.edu/~knuth/ (visited on Oct. 26, 2022) (cit. on p. 3).

[2] “組版 - Wikipedia.” URL: https://ja.wikipedia.org/wiki/%E7%B5%84%E7%89%88 (visited on Oct. 25, 2022) (cit. on p. 3).

[3] Arthur Reutenauer. “A brief history of TEX, volume II.” In: *EuroBachoTEX 2007* (2008), p. 68 (cit. on p. 3).

[4] “TeX と LaTeX の違い - ラング・ラグー.” URL: https://blog.wtsnjp.com/2016/12/19/tex-and-latex/ (visited on Oct. 26, 2022) (cit. on p. 4).

[5] “LaTeX- TeX Wiki.” URL: https://texwiki.texjp.org/LaTeX#f2915fa5 (visited on Oct. 26, 2022) (cit. on p. 4).

[6] “TeX Wiki.” URL: https://texwiki.texjp.org/ (visited on Oct. 25, 2022) (cit. on p. 4).

[7] “Unicode - Wikipedia.” URL: https://ja.wikipedia.org/wiki/Unicode (visited on Oct. 26, 2022) (cit. on p. 4).

[8] 師茂樹. “Unicode とのつきあい方―漢字文化圏を中心に―.” In: コンピュータ & エデュケーション 27 (2009), pp. 12–17 (cit. on p. 4).

[9] “XeTeX - Wikipedia.” URL: https://ja.wikipedia.org/wiki/XeTeX (visited on Oct. 26, 2022) (cit. on p. 5).

[10] “XeTeX - TeX Wiki.” URL: https://texwiki.texjp.org/?XeTeX (visited on Oct. 27, 2022) (cit. on p. 5).

[11] “XeLaTeX で日本語する件について.” URL: http://zrbabbler.sp.land.to/xelatex.html (visited on Oct. 26, 2022) (cit. on p. 5).

[12] 八登崇之. “日本人の知らない TEX” 2010 (cit. on p. 5).

[13] 幸田露伴. “雲のいろいろ.” 反省雑誌, 1987 (cit. on p. 5).

[14] “ConTEXt - TEX Wiki.” URL: https://texwiki.texjp.org/?ConTeXt (visited on Oct. 26, 2022) (cit. on p. 6).

[15] “Biber (LaTeX) - Wikipedia.” URL: https://en.wikipedia.org/wiki/Biber_(LaTeX) (visited on Oct. 30, 2022) (cit. on p. 7).

[16] “TeX Live をホンキで語る - Acetaminophen’s diary.” URL: https://acetaminophen.hatenablog.com/entry/texadvent2016-20161205#main-7 (visited on Oct. 27, 2022) (cit. on p. 8).

[17] “LaTeX コンパイル等 - Yamamoto’s Laboratory.” URL: http://www.yamamo10.jp/yamamoto/comp/latex/run/run.php (visited on Oct. 27, 2022) (cit. on p. 8).

[18] “参考文献の書き方 - 星野光.” URL: https://www.eng.u-hyogo.ac.jp/faculty/hoshino/pc/latex/bibtex/ (visited on Oct. 27, 2022) (cit. on p. 9).

[19] “DVI - Wikipedia.” URL: https://ja.wikipedia.org/wiki/DVI_(%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%83%95%E3%82%A9%E3%83%BC%E3%83%9E%E3%83%83%E3%83%88) (visited on Oct. 27, 2022) (cit. on p. 10).

[20] “PostScript - Wikipedia.” URL: https://ja.wikipedia.org/wiki/PostScript (visited on Oct. 27, 2022) (cit. on p. 10).

[21] “LATEX のの相互参照はいつでも解決（収束）するのか？.” URL: https://qiita.com/kuroky_plus/items/28fc1f4e5610aa5c956e (visited on Oct. 27, 2022) (cit. on p. 11).

[22] “TeX の実行あれこれ - 雑記帳.” URL: https://blog.miz-ar.info/2016/12/running-tex/ (visited on Oct. 27, 2022) (cit. on p. 12).

[23] “「日本語 LaTeX」が多すぎる件について.” URL: https://www.slideshare.net/zr-tex8r/latex-239371115 (visited on Oct. 30, 2022) (cit. on p. 13).

[24] “LuaTeX が実はそんなに遅くないかも知れない件 - マクロツイーター.” URL: https://zrbabbler.hatenablog.com/entry/20170730/1501402087 (visited on Oct. 30, 2022) (cit. on p. 13).

[25] “pLaTeX が本格的にやばいかもという話 - Acetaminophen’s diary.” URL: https://acetaminophen.hatenablog.com/entry/2021/06/18/022108 (visited on Oct. 30, 2022) (cit. on p. 13).

[26] “LaTeX Workshop で llmk してみたらアレだった件.” URL: https://zrbabbler.hatenablog.com/entry/2020/09/23/190840 (visited on Oct. 30, 2022) (cit. on p. 13).

付録

A. ぼくがかんがえたさいきょうのてふかんきょう

タイトルのとおりである．僕の考える最強であって，実際の最強ではないし，日々ぼくのなかのさいきょうはこうしんされつづける！ さらに言えば，ここは特に丁寧に説明する気もないので，個別の環境や目的ごとに場合分けをしていない．書いてあることをそのままコピペではなく，慎重に行うことをすすめる．

(1) distribution

- Windows: 知らない
- MacOS: Homebrew を使う．知らない人は知らない．TeXShop とかはいらないからついてない方．

(2) 確認と準備

TeX のインストールが上手く行かない話はよくあるが，最後までやってから何かでテストしようとするのが良くないと思う．当たり前だがこの時点でコンパイルはできるのでとりあえずやってみるべし．

(3) latexmkrc

本来的には，文書ごとにどの LaTeX を使っているか意識するのがよいのだが，取りあえず何でも使えるやつをつくる．

ここでも一応 `latexmk` が思い通りに動くか確認しておこう．

(4) Visual Studio Code (LaTeX Workshop)

List of Figures

List of Tables