LaTeX 処理自動化ツール ClutTeX

@mod_poppo

~~2019-10-12~~
~~(TeXConf 2019)~~
2019-12-10
(TeX & LaTeX Advent Calendar 2019)

LaTeX 処理

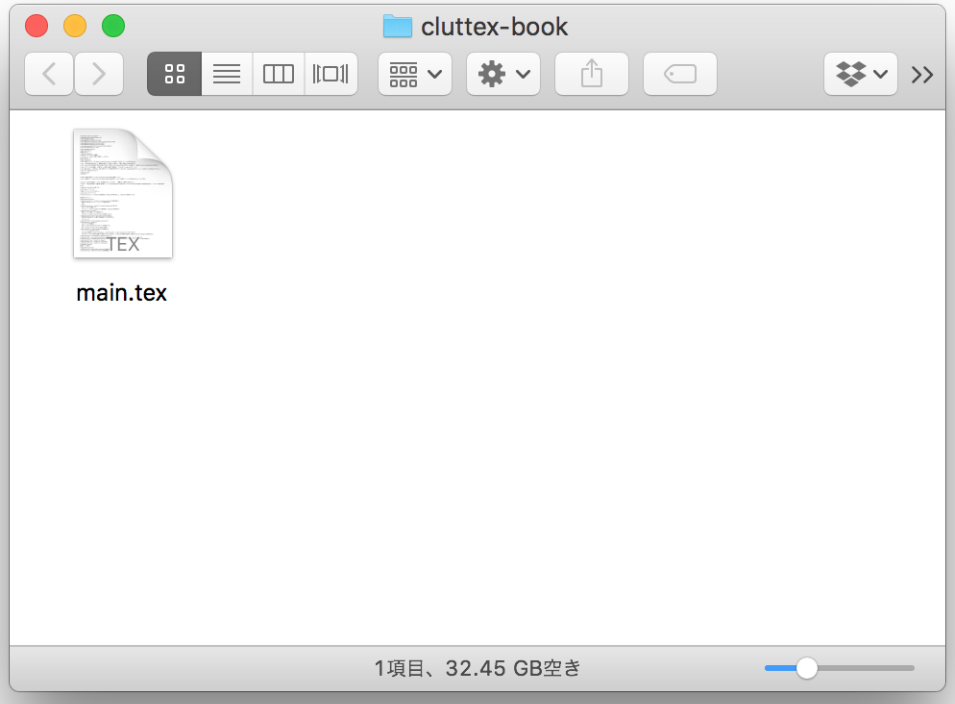
みなさん、LaTeX 文書はどうやって処理していますか？

LaTeX 処理

みなさん、LaTeX 文書はどうやって処理していますか？

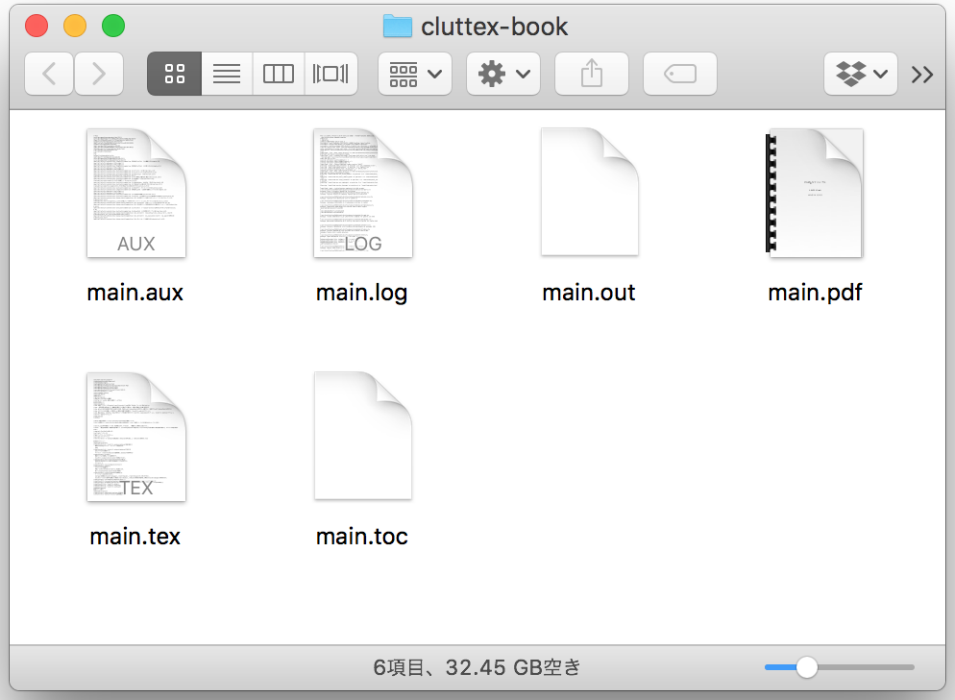
1. コマンドラインで platex/dvipdfmx 等のコマンドを直接叩いている
2. latexmk 等の処理自動化ツールを使っている
3. TeXworks や TeXShop 等の統合環境を使っている
4. Overleaf や Cloud LaTeX 等のクラウド環境を使っている

処理前




```
$ lualatex main.tex
```

処理後


cluttex-book
AUX
main.aux
LOG
main.log
main.out
main.pdf
TEX
main.tex
main.toc
6項目、32.45 GB空き

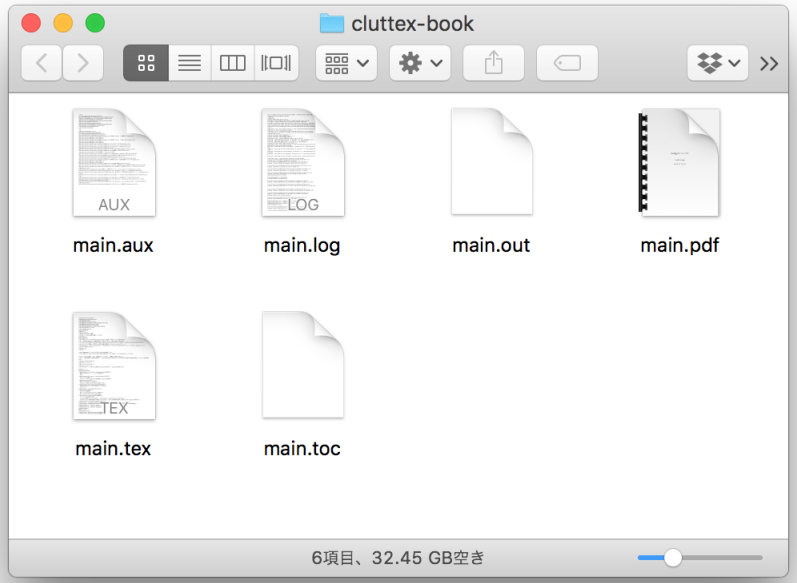
ごちゃごちゃ・・・

LaTeX 処理の問題点

手元で処理させる（1. から 3.）場合、

問題点

.aux や .log 等の「余計な」ファイルができる



※ LaTeX の動作に必要かどうかは関係なく、ユーザーがほしい出力ファイ

ClutTEX の紹介

なにこれ

LATEX 処理自動化ツール

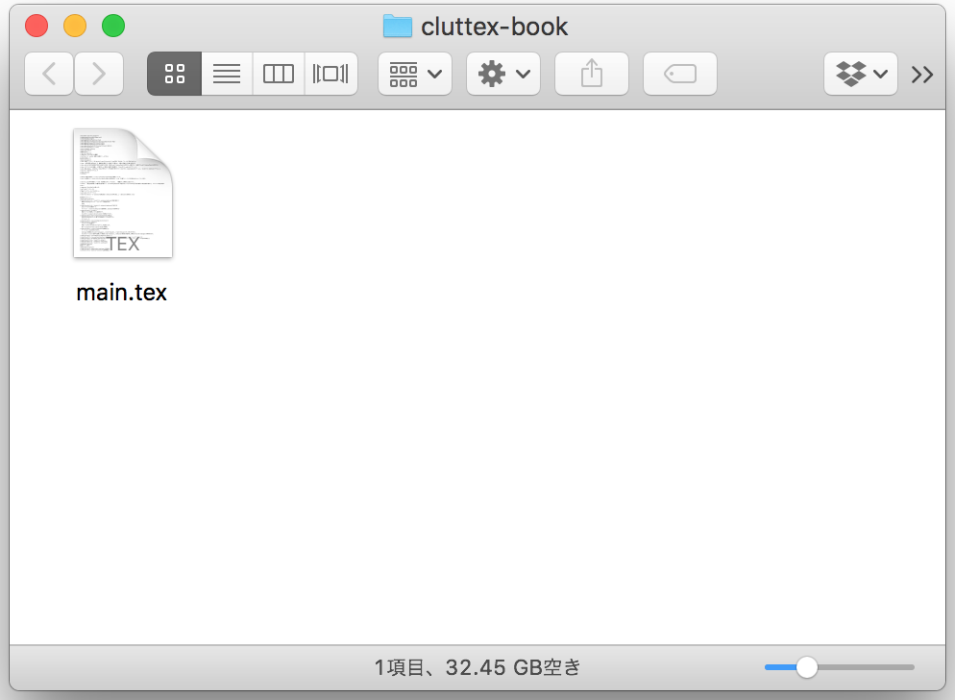
特徴（主な機能）

- **余計なファイルを作らない**
- 必要に応じて LATEX 処理を複数回行い、相互参照等を正しくしてくれる
- pTEX 系でも一発で PDF を生成できる（`dvipdfmx` を呼び出してくれる）
- MakeIndex や BIBTEX と連携できる
- 入力ファイルが変化したら自動で再処理できる（監視モード）

作者

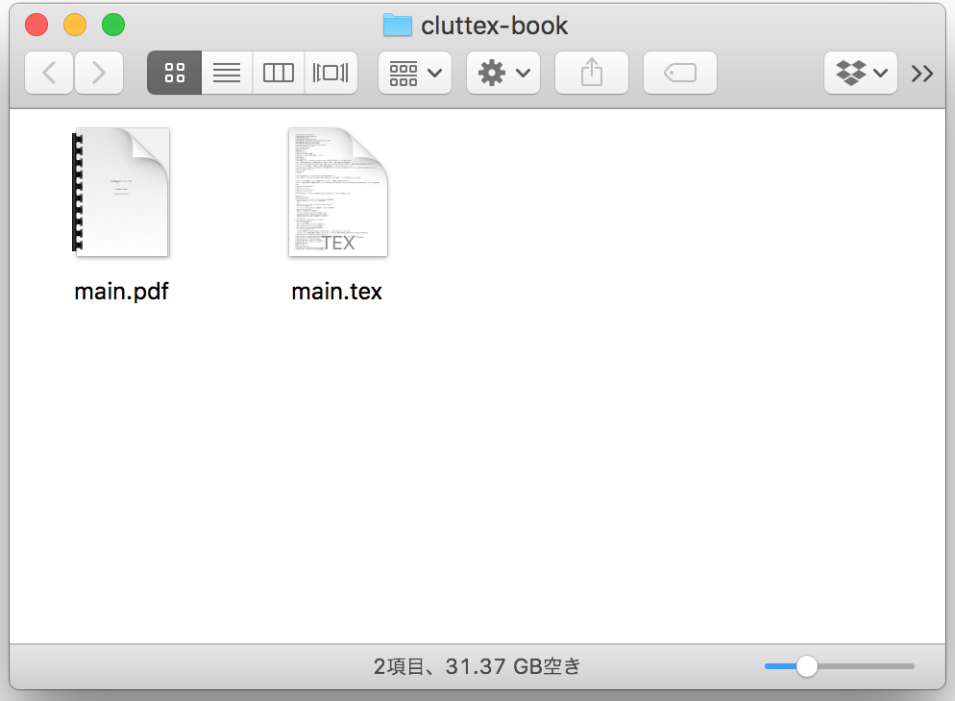
私（@mod_poppo）

処理前（ClutTEX 利用）




```
$ cluttex -e lualatex main.tex
  または $ cllualatex main.tex
```

処理後（ClutTEX 利用）



すっきり！！！

インストール方法

TEX Live

TEX Live 2018 以降に収録。最新の TEX Live であれば`$ tlmgr update --all` で最新版がインストールされるはず。

手動

開発中のバージョンを使いたい場合は、

1. `https://github.com/minoki/cluttex/releases` を開く
2. `.zip` か `.tar.gz` をダウンロード、展開
3. `bin/cluttex` か `bin/cluttex.bat` を適切な場所にコピー

ClutTEX の動作の仕組み

基本的な仕組み

基本的な動作

1. テンポラリディレクトリを用意（以下 OUTDIR）
2. TEX 処理系に -output-directory=OUTDIR を指定して文書を処理する
3. （pTEX 系の場合は）dvipdfmx を呼び出して PDF を作る
4. OUTDIR/ に生成された PDF ファイル（または DVI ファイル）をソースファイルと同じディレクトリにコピーする

基本的な仕組み

基本的な動作

1. テンポラリディレクトリを用意（以下 OUTDIR）
2. TEX 処理系に -output-directory=OUTDIR を指定して文書を処理する
3. （pTEX 系の場合は）dvipdfmx を呼び出して PDF を作る
4. OUTDIR/ に生成された PDF ファイル（または DVI ファイル）をソースファイルと同じディレクトリにコピーする

これだけでうまくいけば苦労はしない

-output-directory をうまく動かすために

LaTeX のエコシステムは -output-directory のことをそこまで考慮していない。

-output-directory がうまくいかない場面

- サブディレクトリのファイルの \include
- 外部コマンド実行
- Lua (\directlua) によるファイル読み書き

\include とサブディレクトリ

大元の文書ファイルから part1/chap1.tex を \include しているとする。LaTeX は補助ファイルを OUTDIR/part1/chap1.aux に書き出そうとするが、OUTDIR/part1 が存在しないと書き込みエラーとなる。

ソースディレクトリ	補助ファイルの出力先	
SRCDIR/	OUTDIR/	
main.tex	main.aux	
part1/chap1.tex	part1/chap1.aux	<-- error
part1/chap2.tex	part1/chap2.aux	<-- error

MiKTEX の --aux-directory はこの辺をうまく対処しているようである。

TEX の外でのファイル読み書き

TEX の枠組みの中でファイルを読み書きする場合
は -output-directory が考慮されるのでだいたいうまくいくが…。

TEX の外でのファイル読み書き

TEX の枠組みの中でファイルを読み書きする場合は -output-directory が考慮されるのでだいたいうまくいくが…。
実際の TEX 文書処理では TEX の枠組みから外れたところでファイルを読み書きする場面がある。

TEX の枠組みの外でのファイルの読み書き

- 外部コマンド実行 (\write18)
- Lua によるファイル読み書き (\directlua, io.open)

外部コマンド実行

外部コマンドにファイルを渡す、あるいは外部コマンドの出力ファイルを文書で利用する場合に困る。

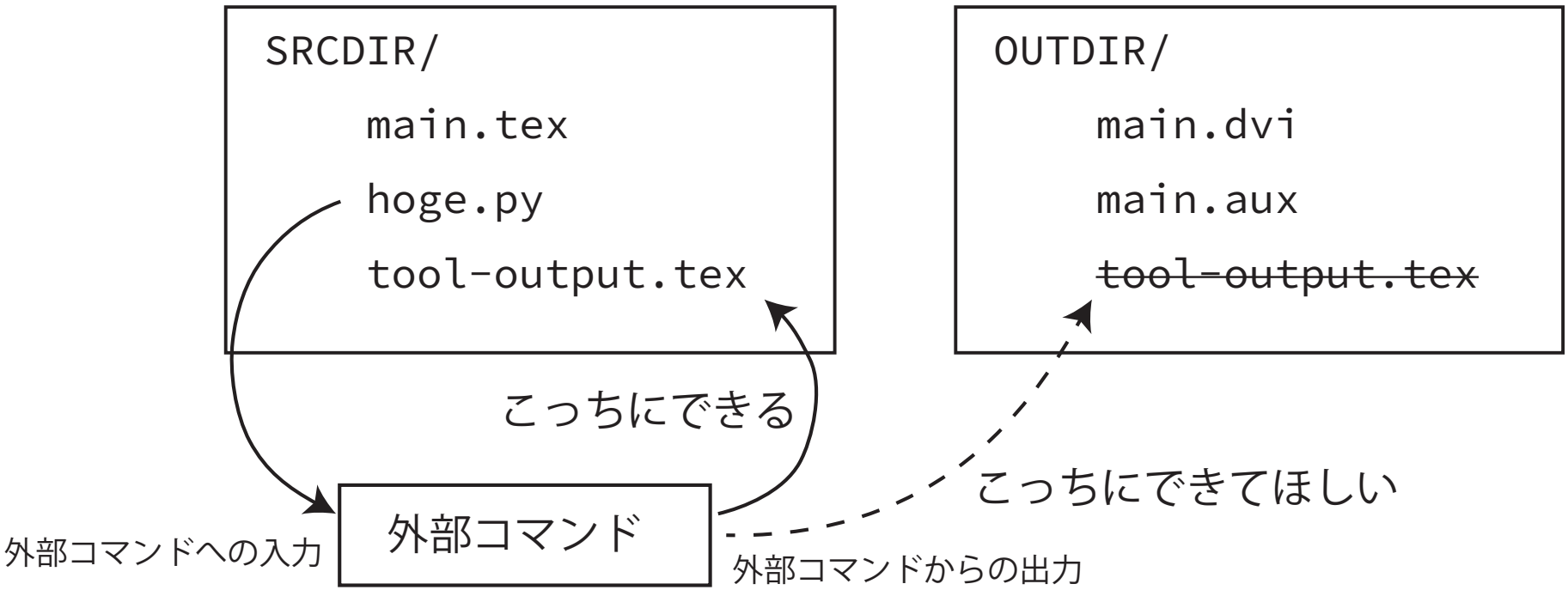
1. ソースファイルと同じディレクトリのファイルを外部コマンドに渡す場合→外部コマンドの出力ファイルが「隔離」されない！
2. パッケージ側でファイルを書き出す場合→外部コマンドの実行に失敗する！

外部コマンドを実行するパッケージ・処理の例

1. シンタックスハイライト（minted パッケージ）
2. gnuplot（TikZ その他）
3. SVG ファイルを EPS や PDF に変換する
4. EPS ファイルを PDF に変換する

外部コマンド実行

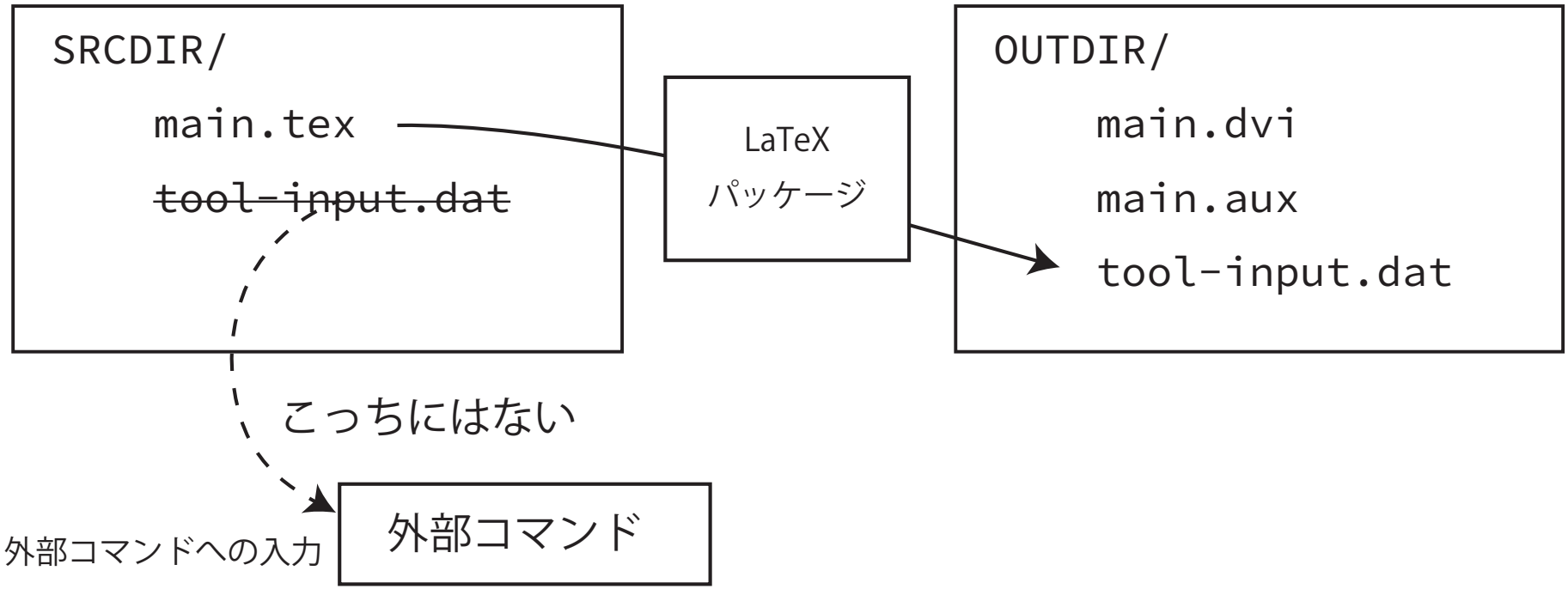
ソースファイルと同じディレクトリのファイルをコマンドに渡す場合（例：\inputminted）。



外部コマンドは SRCDIR/ に出力ファイルを書き出す。これは「補助ファイルを隔離する」観点から、望ましくない。

外部コマンド実行

LaTeX パッケージ側でファイルを書き出す場合（例：minted 環境）。



パッケージが書き出す、外部コマンドへの入力ファイルは OUTDIR/ に書き出される。外部コマンドは SRCDIR/ から入力ファイルを探すので、処理に失敗する。

LuaTEX でのファイル読み書き

Lua の io.open 関数は -output-directory を考慮しない。筆者の知る限り、TEX または Lua コードで -output-directory の値を取得する方法はなさそうなので、パッケージ側でできることは少ない。

例：luatexja-ruby

LuaTEX-ja の luatexja-ruby によって生成される .ltjruby ファイルは -output-directory の影響を受けない。そのため、「補助ファイルを作らない」という目標を達成できない。

ClutTEX では io.open を乗っ取ることで .ltjruby ファイルを OUTDIR/ に書き出すようにしている。

複数回処理

一部の機能は、複数回処理しないと正しい結果が得られない。

- 相互参照（.aux）
- 目次（.toc）
- PDF bookmarks（.out）
- longtable
- TikZ の remember picture
- biblatex の defernumbers を使うと必要な回数が 1 回増える

複数回処理

一部の機能は、複数回処理しないと正しい結果が得られない。

- 相互参照（.aux）
- 目次（.toc）
- PDF bookmarks（.out）
- longtable
- TikZ の remember picture
- biblatex の defernumbers を使うと必要な回数が 1 回増える

何回処理すれば十分？

複数回処理に対するスタンス

- 素の TEX：ユーザーが複数回実行する必要がある
- 自作シェルスクリプト：一定回数（必要と思われる回数）実行する
- latexmk：自動で必要な回数処理する

ClutTEX は latexmk と同じく、「自動で必要な回数処理する」方針。複数回処理したくない場合はオプションを指定すれば良い。

複数回処理に対するスタンス

なぜ「自動で必要な回数」？

「必要な回数」を正しく見積もるのは難しい。
変更が軽微であれば複数回実行する必要はないかもしれない。その場合に何回も実行するのは時間の無駄である。(でかい文書は一回の処理に数分かかったりする)

複数回処理の必要性の判定

TEX 処理系が処理の過程で読み書きするファイルの一覧を得る。この「一覧」は -recorder で得られる。
処理の前後で入力ファイルが変化していなければ、「収束した」と考える。
.aux のような補助ファイルは、「INPUT」「OUTPUT」の両方に現れる。こういうファイルは補助ファイルとして扱う。補助ファイルの変化を検出するのにタイムスタンプは使えなくて、MD5 等のハッシュを取って比較する。

複数回実行の判定の罠

TEX の枠組みの外側で読み書きするファイルがあるとうまくいかない。
つまり、外部コマンド実行や LuaTEX。

隔離か、削除か

`latexmk -c` みたいなやつを使うと補助ファイルを削除してくれたりする。ClutTeX の方式（補助ファイルを隔離された場所に生成する）とどちらが良いか？
補助ファイルの拡張子はパッケージによってまちまち。よく使われている拡張子はツールのデフォルトで対応しているかもしれないが、マイナーなパッケージの補助ファイルは対応していない場合がある。`.ltjruby` とか。
ClutTeX は、TEX の上で読み書きされる補助ファイルは、未知のパッケージであろうが原理的には全て隔離できる。ただ、TEX の外側で読み書きされる補助ファイルには個別対応が必要となる。`.ltjruby` とか。

隔離か、削除か

`latexmk -c` みたいに補助ファイルを削除したら、その後の処理が最初からやり直しとなる。
隔離する方式であれば、明示的に削除しない限り「前回の補助ファイル」を引き継いだ状態で処理を行える。
ただ、補助ファイルを引き継ぐことにはデメリットもある。「TEX ソースは正しいのに前回の `.aux` が悪さをしてエラーになる」という状況が起こりうる。

ClutTEX では

基本的には前回の補助ファイルを引き継ぐが、`--fresh` オプションを指定すると補助ファイルを（`rm -r OUTDIR/` により）一掃する。また、デフォルトではテンポラリディレクトリに補助ファイルを書き出すので、OS の再起動や不要ファイルの削除等で補助ファイルが消える。
ちなみに、`-output-directory` を明示的に指定した場合は `--fresh` オプションは使えない（事故防止のため）。

雑多な話題

OS の API を叩く必要性

Lua の os.execute や io.open、lfs などの関数・ライブラリーを使えばシェルスクリプト並みのことはできるが…。

- ターミナル出力を色付けしたい
- ファイル監視を実装したい

等を考えると OS の API を叩く必要性が出てくる。

ターミナル出力の色付け

1. 標準出力がターミナルかどうか判定する ←環境依存
   - Unix の場合は `isatty(fileno(stdout))`
   - Windows にも `_isatty`, `_fileno` みたいな関数があるが MinTTY を検出できないので API を呼んでごにょごにょ
2. Windows ではそのまま ANSI エスケープシーケンスを出力しても認識されるとは限らない。ちゃんとやるなら、`SetConsoleMode` API を叩いて Virtual Terminal を有効化する ←環境依存
   - Windows 10 のどこかのバージョンで実装されたやつ
3. ANSI エスケープシーケンスを出力して色付けする

詳しくは Qiita に書いた「Windows 向けのプログラムで ANSI エスケープシーケンスを使うには」を参照。

ファイル監視

ファイルの変更を検出して自動で再処理できると便利。どうやってファイルの変更を検出する？

1. 外部のファイル監視プログラムを使う
   - fswatch: クロスプラットフォーム。
   - inotifywait: Linux 向け。Windows 用の互換プログラム (inotify-win) が GitHub に落ちているが互換性が低い。
2. OS の API を叩く
   - Windows の場合は `ReadDirectoryChangesW` や `FindFirstChangeNotification`
   - Linux の場合は inotify
   - macOS の場合は FSEvents

ClutTEX では

Windows では fswatch や inotify-win の動作が怪しかったので自前で API を叩くことにした。それ以外は fswatch コマンドや inotifywait コマンドを使う。

ファイル名の文字コードの問題

Unix で

```
cluttex -e lualatex 円円対応.tex
```

Windows で

```
cluttex -e lualatex 円円対応.tex
```

グッバイ、Windows（完）

Windows の文字コード問題

どれ？

1. Shift_JIS のダメ文字対策をする
2. 「ベータ: ワールドワイド言語サポートで Unicode UTF-8 を使用」を有効にしてもらう
3. ファイル名やコマンドライン引数を UTF-8 で扱うように手を加えた Lua インタープリターを使う
4. 諦める。Windows には WSL があるし？

ClutTEX では

基本的にノーガード戦法。一応、3 番の案を GitHub の `windows-u8` ブランチで試している。

Windows の文字コード問題

どれ？

1. Shift_JIS のダメ文字対策をする
   - →前世紀。日本語以外でおかしくなる可能性がある。却下。
2. 「ベータ: ワールドワイド言語サポートで Unicode UTF-8 を使用」を有効にしてもらう
   - → pTEX が動かなくなる。却下。
3. ファイル名やコマンドライン引数を UTF-8 で扱うように手を加えた Lua インタープリターを使う
   - → TEX Live（W32TEX）次第。あるいは、ClutTEX 側でバイナリ配布する。
4. 諦める。Windows には WSL があるし？

ClutTEX では

基本的にノーガード戦法。一応、3 番の案を GitHub の `windows-u8` ブランチで試している。

アンケート

Windows、使ってますか？

- 使っている（最新の Windows 10）
  - 普段は WSL を使っている
  - 普段は Windows 側のコマンドラインツールを使っている
- 使っている（古い Windows）
- 使っていない

Windows 10 を使っている人へ：「ベータ: ワールドワイド言語サポートで Unicode UTF-8 を使用」は有効にしていますか？

- 普段から有効にしている
- 試したことがあるがうまく動かないので無効化した
- 試していない

Lua 同士の互換性

独断と偏見で選ぶ Lua の更新履歴

Lua 5.1 正式版は 2006 年 2 月リリース。

Lua 5.2 正式版は 2011 年 12 月リリース。

- モジュールの文化が変わった
- `_ENV`, `goto`, `bit32` library, UTF-8 BOM
- `os.execute` の返り値の仕様が変わった（後述）

Lua 5.3 正式版は 2015 年 1 月リリース。

- 64 ビット整数とビット演算子
- `utf8` library
- `"\u{1F986}"`

LuaJIT 2.0 正式版は 2012 年 11 月リリース。基本は Lua 5.1 ベースで、ABI 互換性を壊さない更新を Lua 5.2 から取り込んでいる。

LuaJIT 2.1 beta Lua 5.3 の `\u` が取り込まれているようだ。

オフトピ：Lua 5.4

Lua の話題ついでに

Lua 5.4 2019 年 10 月現在、beta 版が出ている。

- generational GC
- to-be-closed variables
  ```
  local f <close> = io.open("f.txt")
  ```
- const variables
  ```
  local a <const> = 42
  ```

詳しくは `http://www.lua.org/work/doc/#changes` をチェック！

LuaTEX の Lua

- いくつかの定数や関数が追加されている。`os.type`, `os.setenv` 等
- 追加の付属ライブラリー：`lfs` や `md5`
- LuaJIT 互換の FFI ライブラリー（luaffifb がベースっぽい）が付属する。
  - ヌルポインターの扱いが違う（`nil` vs `ffi.NULL`）
  - 64 ビット整数：本家 LuaJIT および Lua 5.2 では独自の boxed integer (signed or unsigned), Lua 5.3 では Lua 標準の 64 ビット整数 (signed only)
- `os.execute` の返り値の仕様が Lua 5.1 時代のまま。

```
print(os.execute("exit 42"))
--> nil  exit  42    (Lua 5.2 or later)
--> 10752            (Lua 5.1 or LuaTeX, on Unix)
```

- LuajitTEX では `require "lfs"` が動かない。

Lua での開発の限界

Lua での開発の限界

静的型が欲しい。特に FFI を使っていると暗黙の型変換が動いたり動かなかったりして辛い。

考えられる対策

- 他のコンパイル型言語に移行する？
- 他の言語から Lua にトランスパイルする？

今後

機能追加

- ビューワーの起動
- 設定のカスタマイズ（ビューワー、色づけなど）
- `cluttex-support` みたいな LaTeX パッケージを用意して、`-output-directory` を考慮していないパッケージにパッチを当てる？

普及

- ~~マニュアルを書く（英語と日本語）~~ →書いた
- 使い方の解説記事を書く（英語~~と日本語~~）→日本語記事は書いた
- ~~同人誌を出す（日本語）~~ →出した（技術書典 6）
- `latexmk` と並ぶ選択肢として認知させていきたい

リンク

GitHub https://github.com/minoki/cluttex
CTAN https://www.ctan.org/pkg/cluttex
ブログ記事 https://blog.miz-ar.info/2016/12/cluttex/
同人誌 https://lab.miz-ar.info/cluttex-book/