LaTeX処理自動化ツール ClutTeXの近況と今後の計画

2024-11-30 (TeXConf 2024)

荒田実樹

1. **ClutTeXの紹介**
2. ClutTeXの動作の仕組み
3. ClutTeXの実装に使うプログラミング言語
4. ClutTeXの新機能

ClutTeXの紹介

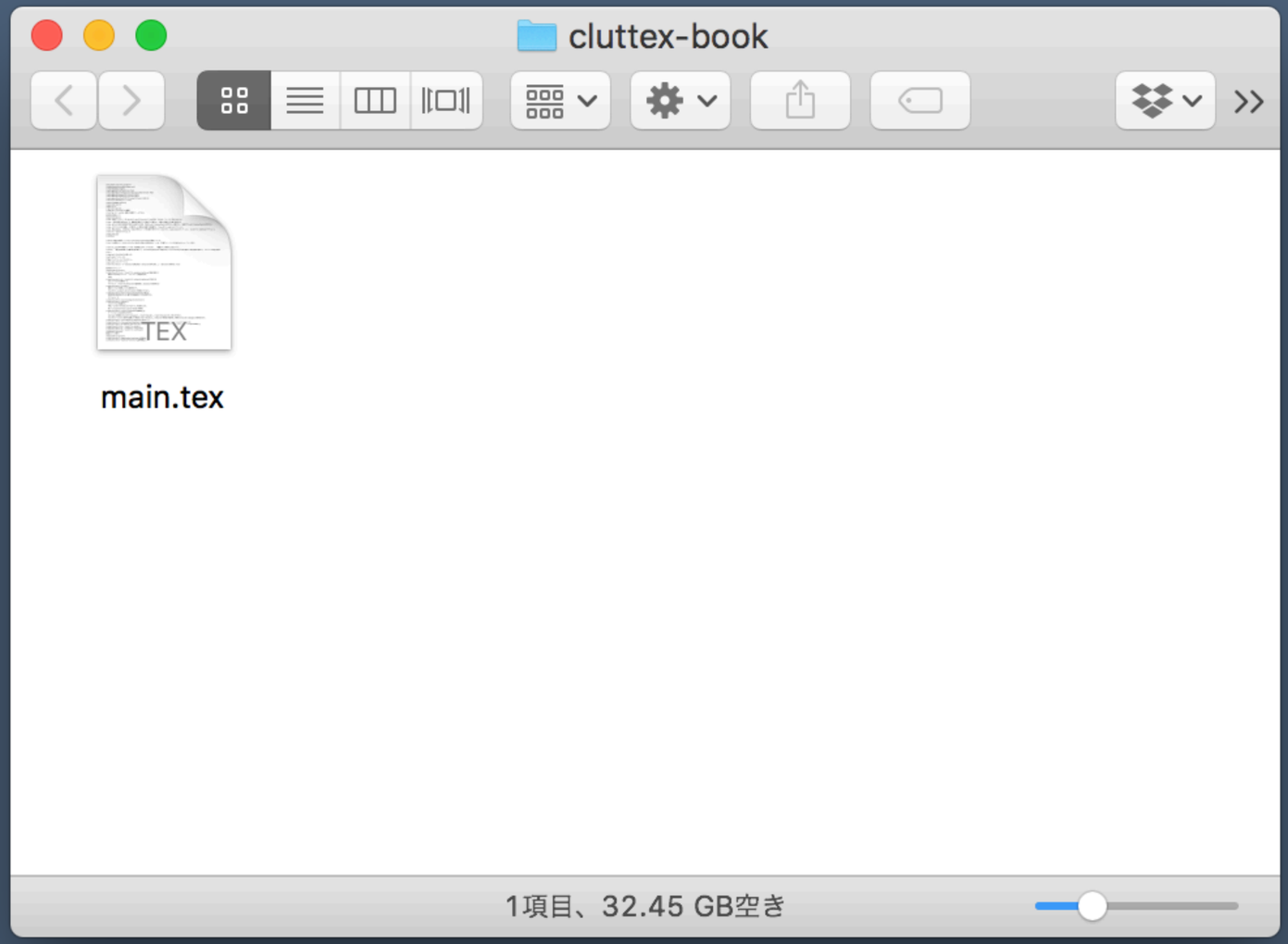
LaTeX文書の処理方法

- みなさんは普段どういうやり方でLaTeX文書を処理していますか？

  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. 自前の**シェルスクリプトや****Makefile**からplatex/dvipdfmx等のコマンドを叩いている
  3. latexmk等の**処理自動化ツール**を使っている
  4. TeXworksやTeXShop、LaTeX Workshop等の**統合環境**を使っている
  5. OverleafやCloud LaTeX等の**クラウド環境**を使っている

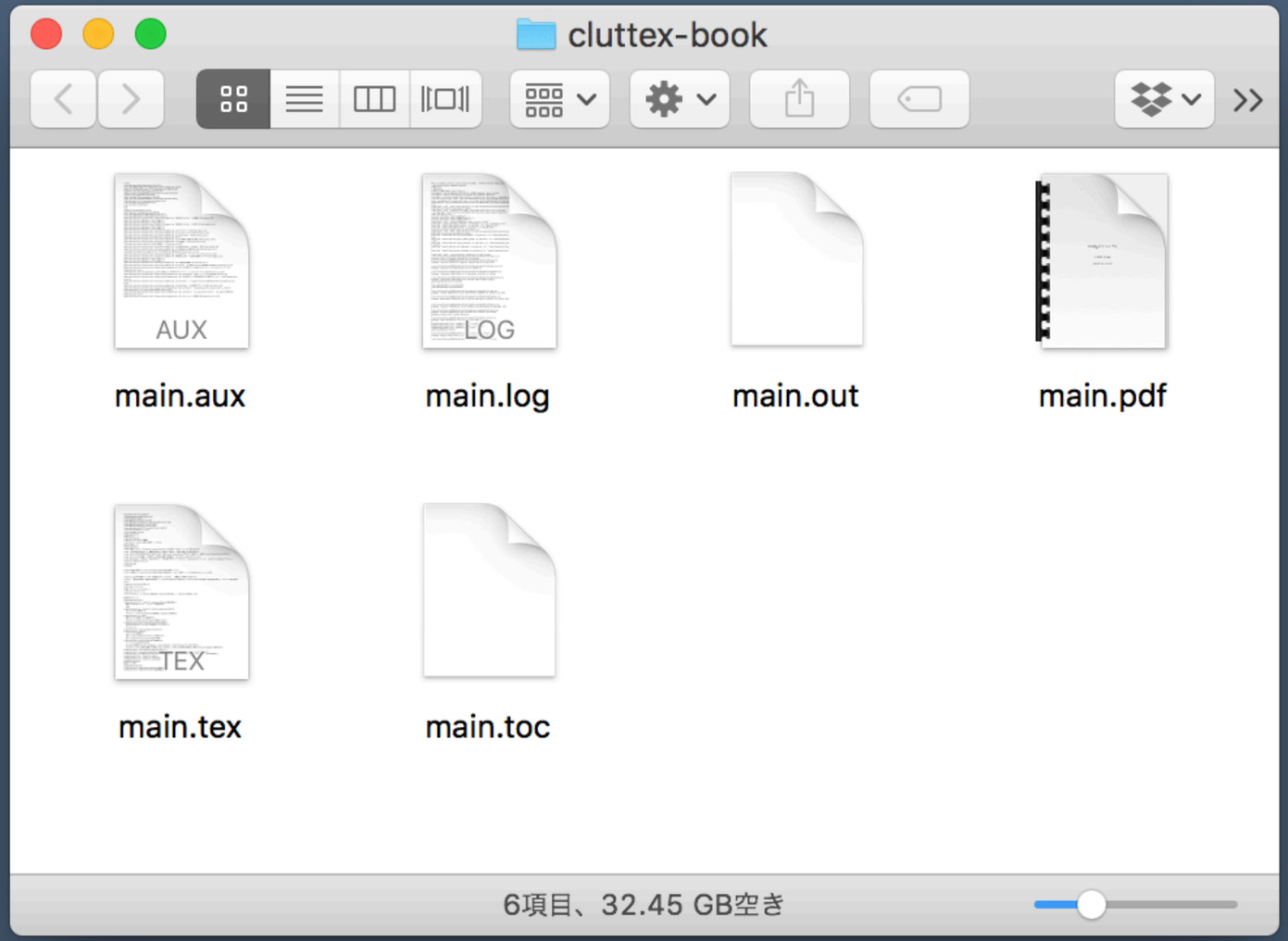
処理前

$ lualatex main.tex



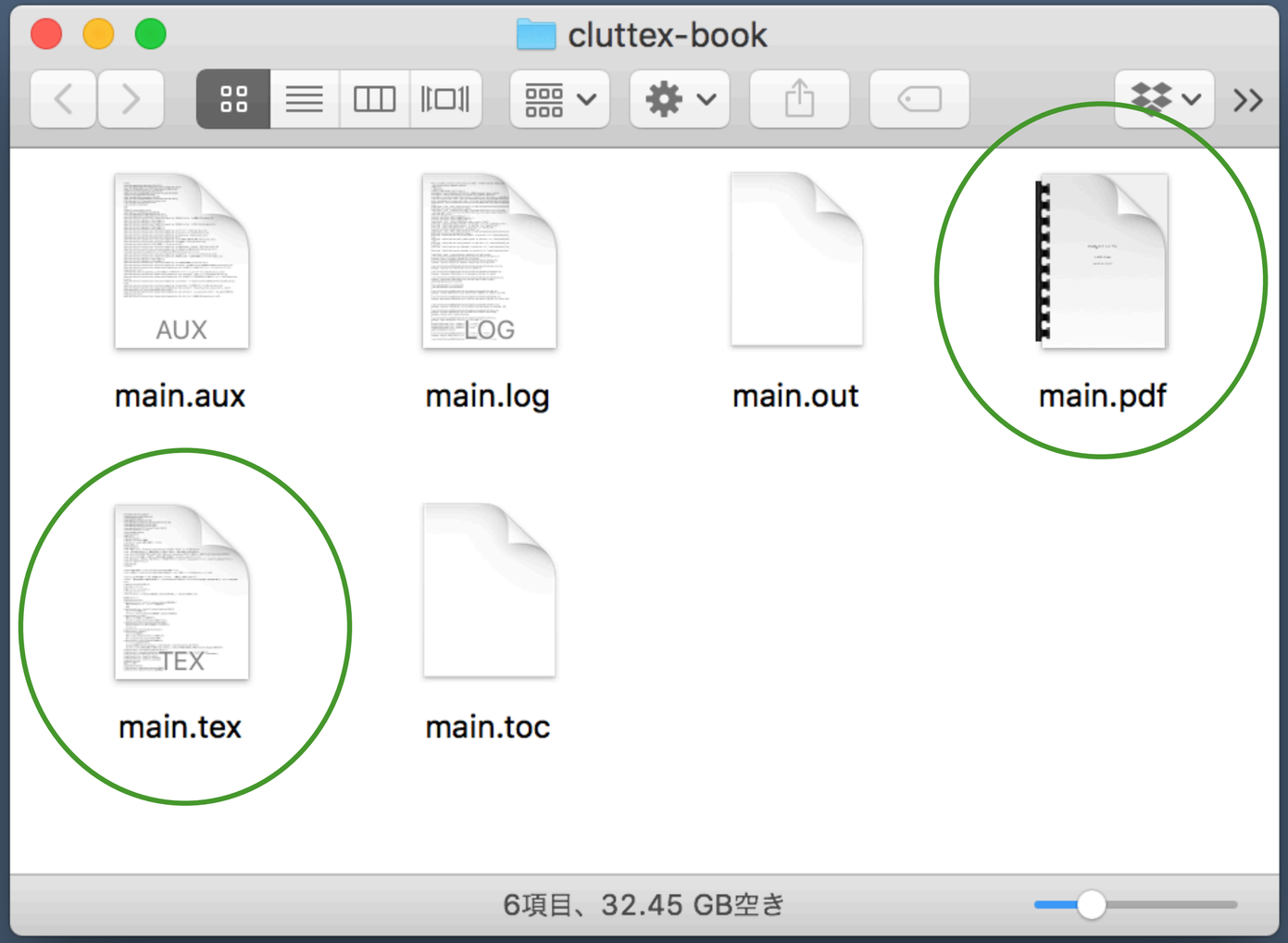
処理後

ファイルが多い！


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

処理後

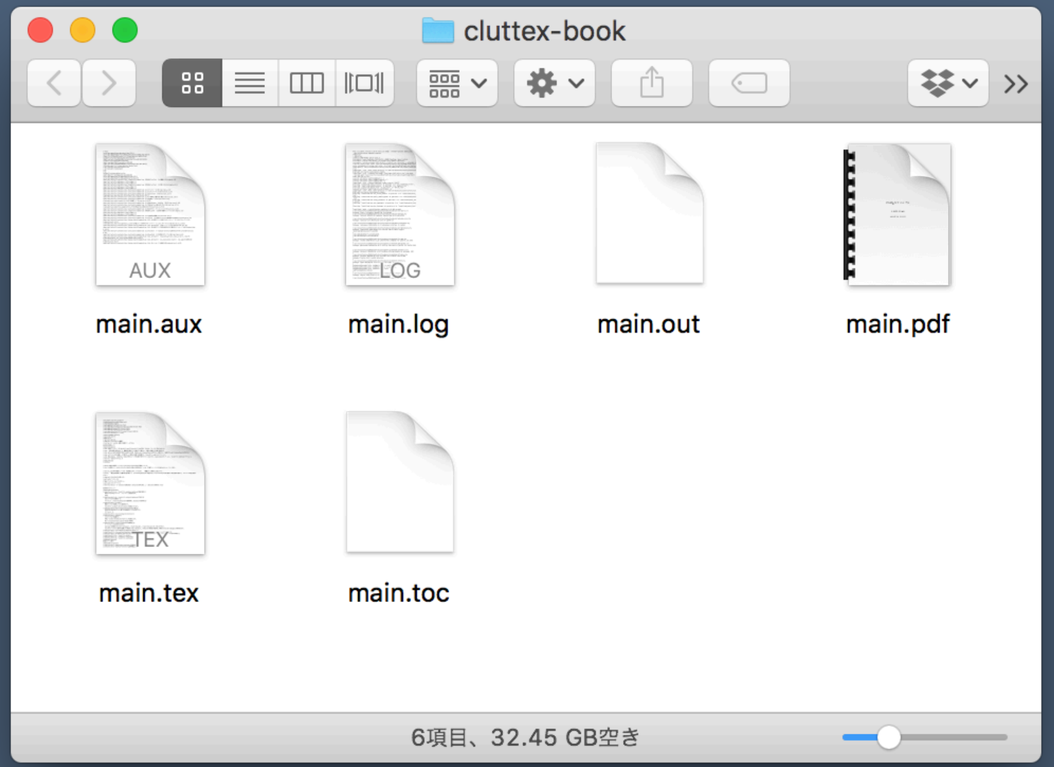
ファイルが多い！



ユーザーの興味があるのはこの2つだけではないのか

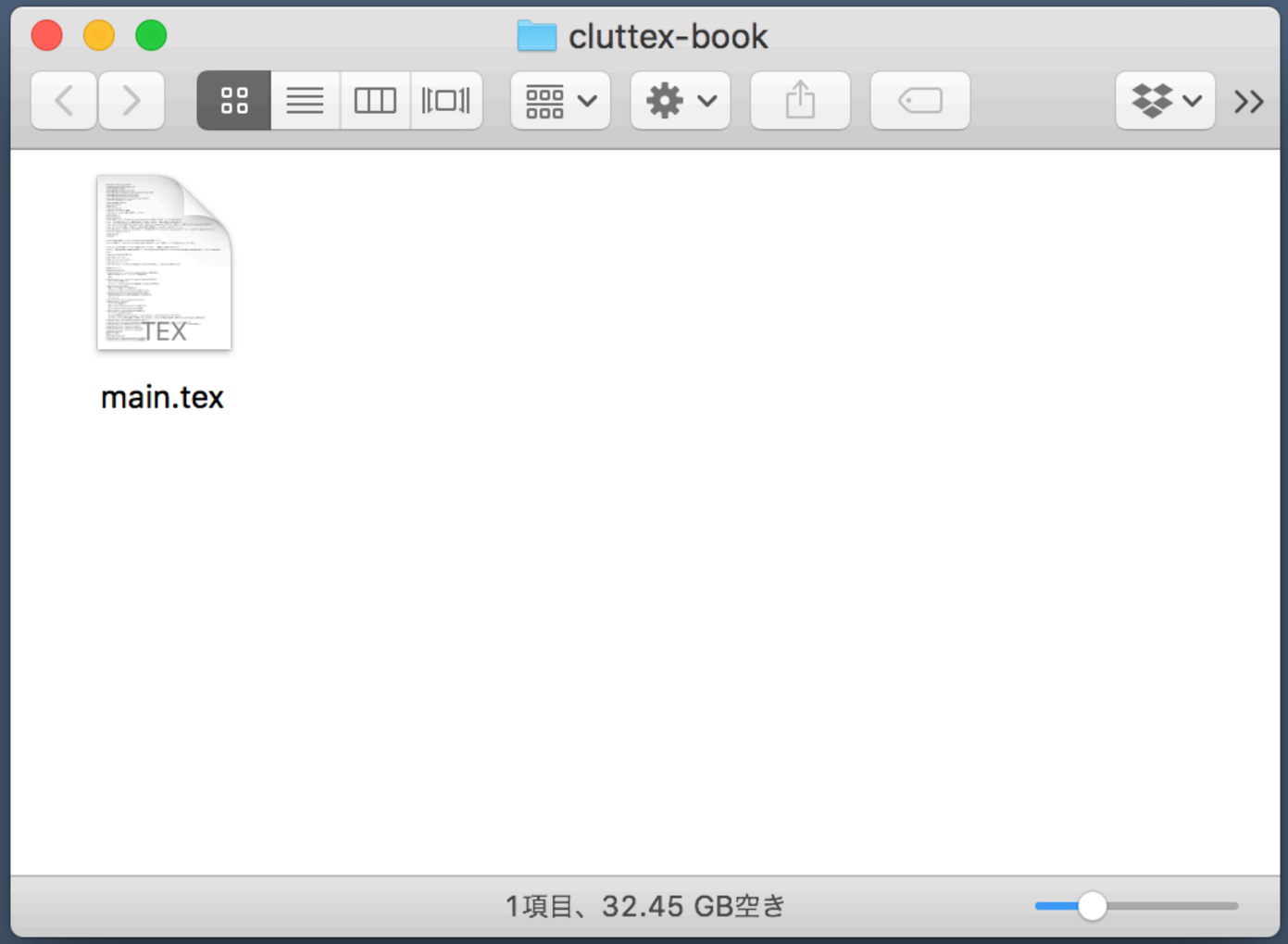
手元で処理する場合の問題点

- .auxや.log等の「余計な」ファイルができる
  - ※LaTeXの動作には.aux等は必要だが、一般ユーザーの興味があるのは.pdfだけ
- これらはクラウドストレージ等に同期されても嬉しくない



ClutTeX での処理前

```
$ cluttex -e lualatex  \
  main.tex
```

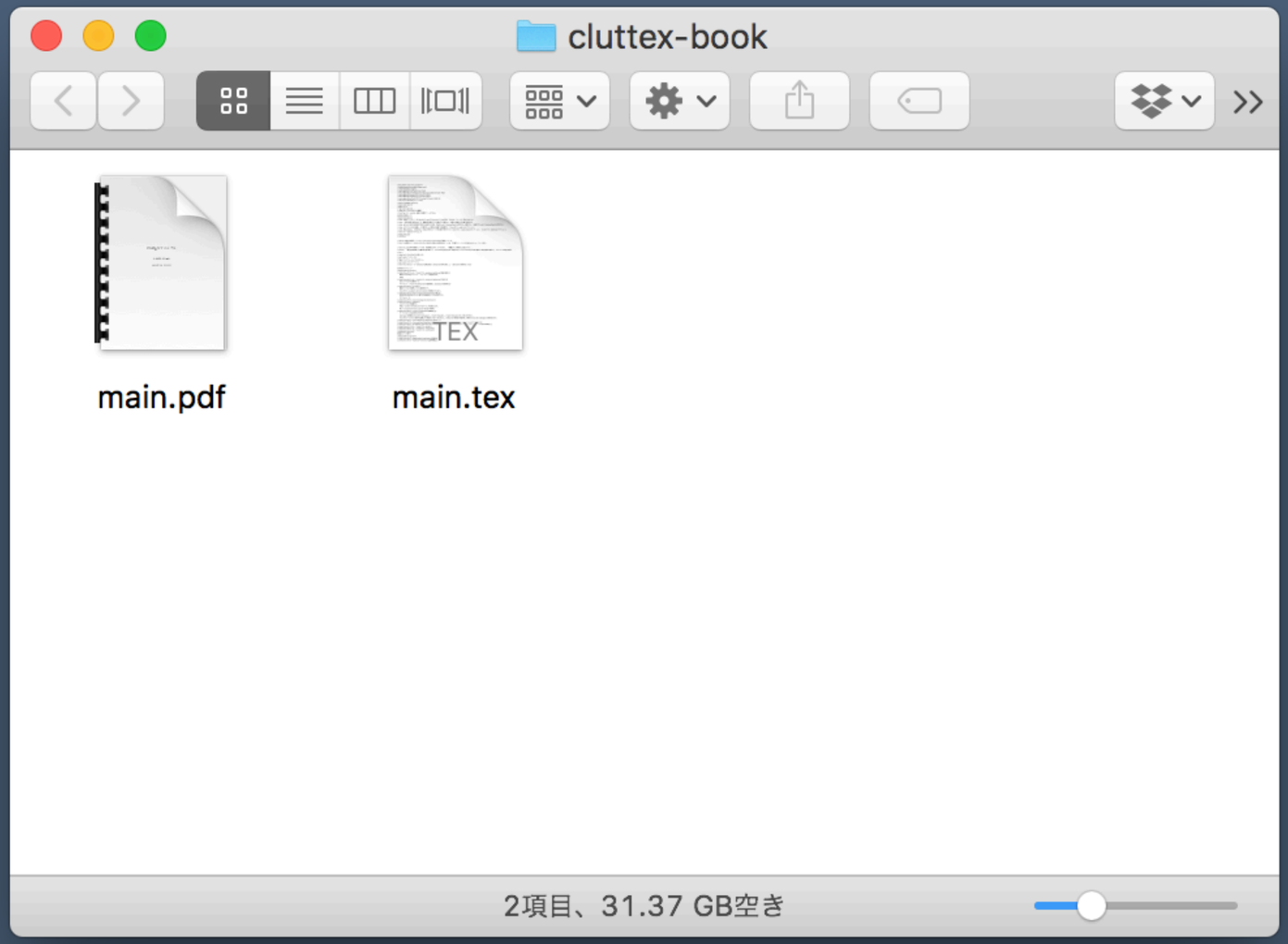


ClutTeX
での処理後

最終生成物の.pdfだけが

生成される

すっきり！



ClutTeXの最大の特徴

- 「余計な」ファイルをユーザーに見せない
  - 補助ファイルは削除されたわけではない。ユーザーから見えない位置に作る（隔離する）

ClutTeXの他の特徴

- 必要に応じてLaTeX処理を複数回行い、相互参照等を解決してくれる（latexmkの役割も兼ねる）
- 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するモード（監視モード）
- 2016年に開発開始した、比較的新しいツール

ClutTeXの他の機能

- --includeonlyオプション：ソースの先頭に\includeonlyを挿入できる
- パッケージに対するDVIドライバーの指定が正しいかチェックする機能
- ファイル名の特殊文字対策

ClutTeXのインストール方法

- TeX Live 2018以降に収録
- 開発中のバージョンを使いたい場合は https://github.com/minoki/cluttex を参照

新規開発した理由

なぜlatexmkのフォークではないのか

- Perlが不得手だったので

思想：一つのことをうまくやる

- ClutTeXは「補助ファイルを隔離する」という目的をうまく果たすことに注力する
- 提供しない機能：プロジェクトファイルやマジックコメント
  - 必要なら外部ツール（make等）を使う
  - 連携を補助する機能はClutTeX側で適宜実装する（例：Makefile用の依存関係の書き出し）
  - 本当は監視モードも外部ツールで実現するべきだったかもしれない
- 作者の好みによりデフォルト動作を変えていることがある（-interaction=nonstopmode, -halt-on-error）

1. ClutTeXの紹介
2. **ClutTeXの動作の仕組み**
3. ClutTeXの実装に使うプログラミング言語
4. ClutTeXの新機能

ClutTeXの動作の仕組み

基本的な仕組み

1. 作業用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. （必要に応じて）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

1. 作業用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. （必要に応じて）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くれない
- -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

-output-directoryがうまく動かない場面その1

- 大きな文書はパートごとにディレクトリを分割したいかもしれない
- 大元の文書ファイル（main.tex）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
- MiKTeXの--aux-directoryはこの辺をうまく対処しているようである
- ClutTeXはエラーメッセージを読み解いて足りないディレクトリを用意する

TeXの外でのファイル読み書き

-output-directoryがうまく動かない場面その2

- TeXの枠組みの中でファイルを読み書きする場合は-output-directoryが考慮されるので大体うまくいくが.....
- 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
  - 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

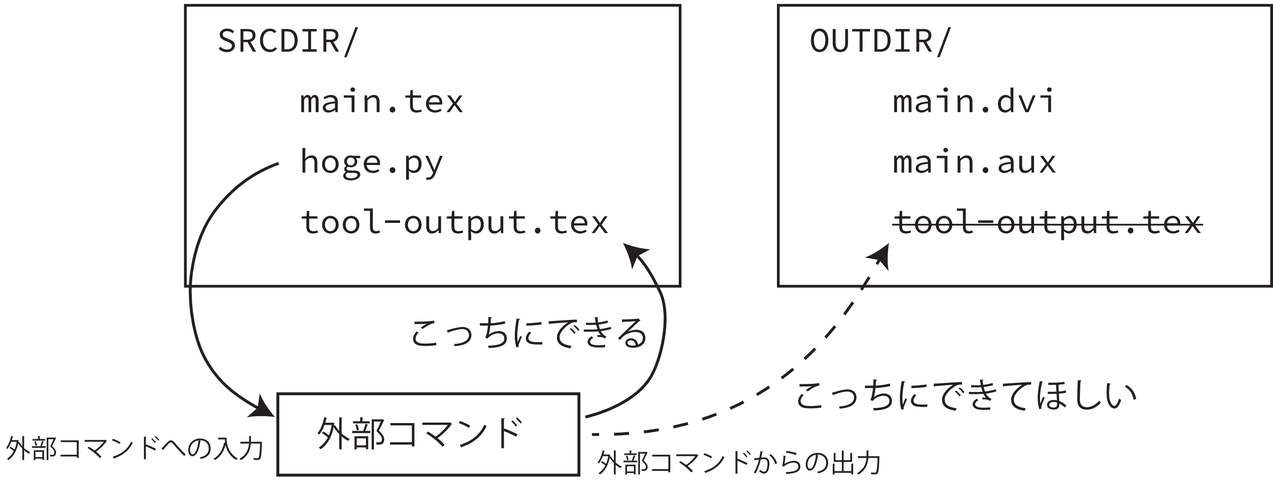
- 外部コマンドにファイルを渡す、あるいは外部コマンドの出力ファイルを文書で利用する場合に困る
  - ソースファイルと同じディレクトリのファイルを外部コマンドに渡して外部コマンドに処理させる場合
    - →外部コマンドの出力ファイルが「隔離」されない！
  - パッケージ側でファイルを書き出して外部コマンドに与える場合
    - →外部コマンドが書き出されたファイルを見つけられず、実行に失敗する！
- 関連：TeX Live 2024の環境変数

外部コマンド実行

うまくいかない場合1

- ソースファイルと同じディレクトリのファイルを入力として外部コマンドに渡す場合（例：\inputminted）

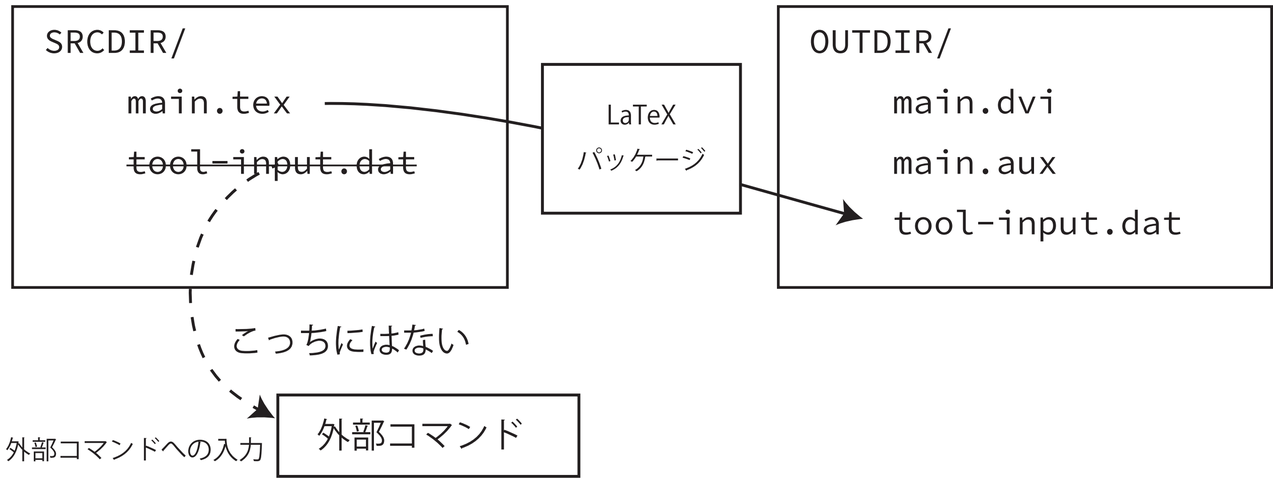
- 外部コマンドはSRCDIR/に出力ファイルを書き出す。これは「補助ファイルを隔離する」観点から、望ましくない



外部コマンド実行

うまくいかない場合2

- 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（要調査：TeX Live 2024）
- 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

ClutTeXの弱点

- 外部コマンド実行やLuaTeXのパッケージは、個別対応が必要になりがち
  - 正常に処理できない、あるいは文書のあるディレクトリにファイルが作られる
- TeX Live 2024からは環境変数TEXMF_OUTPUT_DIRECTORYが使えるようになったので改善していくのかもしれない
  - mintedはすでに対応している

複数回処理

- 一部の機能は、複数回処理（複数回latexを実行）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増える

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
  - 処理回数が足りない場合はそういうメッセージが出ることがある
- 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
  - ログ中のメッセージを参考にすることもできる
- latexmk：自動で必要な回数処理する

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
  - 処理回数が足りない場合はそういうメッセージが出ることがある
- 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
  - ログ中のメッセージを参考にすることもできる
- 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

複数回処理に対するスタンス

ClutTeXの考え

- 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
  - 前回処理からの変更が軽微であれば1回の実行で済むかもしれない。その場合に固定の回数処理するのは時間の無駄である（大きな文書は1回の処理に数分かかったりする）
- 補助ファイルを見る必要があるので、外部ツールに任せるのは難しい
- 複数回処理が望ましくない場合は「処理回数の最大値」を1に指定すればよい

複数回処理の必要性の判定方法

- TeX処理系が処理の過程で読み書きするファイルの一覧は-recorderオプションで得られる
- 「INPUT」のみに現れる→入力ファイル
  - タイムスタンプが新しくなっていたら追加の処理が必要
- 「OUTPUT」のみに現れる→出力ファイル
- 「INPUT」「OUTPUT」の両方に現れる→補助ファイル
  - MD5等のハッシュ値を比較して、変化していたら追加の処理が必要
- TeXの枠組みの外側で読み書きするファイルがあると-recorderだけでは済まない。外部コマンド実行とLuaTeX

1. ClutTeXの紹介
2. ClutTeXの動作の仕組み
3. **ClutTeXの実装に使うプログラミング言語**
4. ClutTeXの新機能

ClutTeXの実装に使うプログラミング言語

当初の選択肢：Lua

- TeX Liveでのバイナリーの配布はよくわからない
- スクリプト言語：多くの場合はインタプリタが別途必要となる
- 特筆すべき言語
  - Perl：多くのUnixで使える、WindowsにはTeX Liveにインタプリタが付属する
  - Lua：TeX Liveにインタプリタが付属する（TeXLua）
- PerlかLuaであればユーザーが使いやすそう
- 作者の好みでLuaを選択

規模が大きくなるとLuaは辛い

Luaには静的型がない

規模が大きくなるとLuaは辛い

- ClutTeX v0.6の段階で3600行程度のLuaコード
- 実装したい新機能はまだある→コードはもっと増える
- Luaには静的型がない→しょうもない型の取り違えがあっても静的に検出されない
- とはいえ、「TeX Liveの他に別途インタプリタを用意しなくても動かせる」というLuaのメリットは捨てがたい
  - →他のプログラミング言語からLuaへコンパイル（トランスパイル）しよう

Luaへのトランスパイル（altLua）

- 今でこそいくつか選択肢がある
  - Haxe
  - TypeScriptToLua
  - Teal
- 5年ほど前は選択肢が乏しかった
  - HaxeはClutTeXを開発した頃からあるが、外部ライブラリ依存の観点で不安がある

コンパイラがないなら作ればいいじゃない

- Luaをコンパイルターゲットとする言語処理系を自分で書くことにした
  - 2018年ごろから着手、本格的な開発は2020年ごろから
- コンパイル元としては既存の言語を使う（エコシステム整備の省力化）
- ML系言語やHaskellが好きだったのでML系言語をいくつか候補にして、Standard MLをコンパイル元として選定した

Standard MLってどんな言語？

- 言語機能は控えめ→実装しやすい
  - 型推論
  - 代数的データ型とパターンマッチ
  - プログラムの再利用を可能にするモジュールシステム
- 標準仕様があり、いくつかの処理系がある（SML/NJ, MLton, MLKitなど）
- 流行っているとは正直言い難い
  - パッケージなどのエコシステムは低調、最近Language Serverが登場した

LunarML

- Standard MLからLuaやJavaScriptへコンパイル（トランスパイル）する処理系
- https://github.com/minoki/LunarML
  - 2024年11月時点でのスター数は350程度
  - バージョン0.2.1

ClutTeXの移植

- ClutTeXをStandard MLで書き直した
- 一部のコードはLuaのまま残している（Standard ML側から呼び出して使う）
- バージョン0.7としてリリース予定

SMLで書き直したのは何のためか？

→新機能を実装するため

1. ClutTeXの紹介
2. ClutTeXの動作の仕組み
3. ClutTeXの実装に使うプログラミング言語
4. **ClutTeXの新機能**

ClutTeXの新機能

ClutTeXの新機能案

- 設定ファイル
- 正常終了／異常終了時に何らかのコマンドを実行する
- プロジェクトファイルを解釈するツール
- 各種パッケージのサポートの改善
  - BibTeXサポートの改善
  - glossariesサポートの改善

新機能：設定ファイル

- 提供しない機能：プロジェクトファイルやマジックコメント
- 一方で、ユーザーごとの（プロジェクトに依存しない）設定ファイルの余地はある
  - 作業ディレクトリのデフォルトの位置
  - ターミナル出力の色付けのカスタマイズ
- TOMLファイルで書けるようにする。v0.7.0で実装予定
  - Standard MLでTOMLパーサーを書いた：https://github.com/minoki/sml-toml

新機能案：終了時にコマンドを実行

- LaTeX処理が終了してPDFが生成された、あるいはエラーが出た場合にユーザーが指定したコマンドを実行する機能
- 用途
  - 正常終了した場合にビューワーを起動したい
  - 監視モードでは、エラーがあった場合にOSの通知機構で教えてくれると便利

新ツール案：プロジェクトファイル？

- ClutTeXと連携する別ツールがあると良いのでは
- 動機：Makefile連携は想定しているが、makeだけでは手間がかかる場合がある
  - 監視モードなど
- 形態：プロジェクトファイルか、マジックコメントか
  - 一つの文書ファイルから複数パターンの出力を得ることを考えると、文書とは別に記述されていた方が良い
  - 複数パターンの内訳：includeonly, プレビュー用の文書
- TOML形式でプロジェクトファイルを書いて、新ツールで解釈し、適宜ClutTeXを呼び出すイメージ

フィードバックをお待ちしています

(GitHubスターも頂けると嬉しいです)

https://github.com/minoki/cluttex

現地でいただいた反応・質問等

- 「ClutTeX」の読み方は？→作者は「クラテック」と読んでいる
- 補助ファイル・ログファイルを保存するニーズがある（原稿を受け取る側）。zip等で固められると良いかも
- longtableのことを考えるとmax-iterationsのデフォルト値は4以上が良い→会場で「デフォルト値は4です」と答えてしまったが、改めて確認すると3だった。次回のアップデートで考慮する
- SMLから（Luaコードではなく）実行可能バイナリにすることもできるのか？→現状は（全てSMLに移植したわけではなく）Luaのコードが残っているが、追加の労力をかければ原理的にはできる

おまけ：Apple Silicon対応の話

ClutTeXのApple Silicon Mac対応

- MacのCPUがIntelからApple Silicon (Arm64)に変わった
- ClutTeXは何か対応が必要か？
  - ClutTeXはLuaで書かれている（当時）ので特別な対応は不要かと思ったが……。

Lua FFIとApple Silicon Mac

- ClutTeXは「標準出力がターミナルに繋がっているか」を判定するためにFFIを使っている
  - 目的：標準出力を色付けするかどうか判断する
  - isatty関数を呼び出す
- LuaTeXのFFIはApple Silicon Macでは使えない！
  - LuaTeXのFFIはLuaJITのものと概ね互換だが、独自実装（本家LuaJITはもちろんApple Silicon Macに対応している）

ClutTeXでの対処

- シェルのtestコマンドを呼び出して「標準出力がターミナルに繋がっているか」を判定する
  - os.execute("test -t 0")