

最大限 $\text{T}_{\text{E}}\text{X}$ 入門

インストールと利用法

北海道大学理学院 宇宙理学専攻 M1 人見祥磨

令和 2 年 5 月 8 日

[改訂第 7 版] $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 美文書作成入門
(奥村晴彦・黒木祐介 著 技術評論社 (2017))

- 3 年毎に改版
(2020 年に改版)
- 「とりあえずこれを読め」



👤 美文書何章に記述があるか適宜参照します 👤

扱うことと扱わないこと

扱うこと

- $\text{T}_{\text{E}}\text{X}$ とはなにか
- $\text{T}_{\text{E}}\text{X}$ のインストール方法
- $\text{T}_{\text{E}}\text{X}$ の初歩的な使い方

扱わないこと

- 文書を書くのに使う^{コマンド}命令 (美文書 3, 5-11 章の大半)
- ^{コマンド}命令の作成 (美文書 4 章)
- 🍣 $\text{T}_{\text{E}}\text{X}$ 言語 🤢

目次

T_EX 概観

T_EX のインストール

T_EX の使い方

L^AT_EX の書き方

エラーへの対処

T_EX のディレクトリ構成

TeX 概観

👤 美文書 1 章 👤

T_EX できること、特徴

文字を並べた PDF を作ることができる。

T_EX できること、特徴

文字を並べた PDF を作ることができる。

- きれいな数式

$$\int_0^{\infty} \frac{\sin x}{\sqrt{x}} dx = \sum_{k=0}^{\infty} \frac{(2k)!}{2^{2k(k!)^2}} \frac{1}{2k+1} = \frac{\pi}{2}$$

- 相互参照、処理の自動化
- フリーソフト
- 様々な OS で利用可能
- 実体はテキストファイル

T_EX でできないこと

- 見たまま編集

Word などを使えば良い、もしくは LyX?

- 図の描画

ほかソフトで作ってから埋め込めば良い、もしくは TikZ?

- フォントを自在に扱う

LuaT_EX や X_YT_EX で使える fontspec パッケージを使えば.....

T_EX とは何か

- 1978 年に Donald E. Knuth が発表
 - 相当に古い
- 組版システム
 - 組版するためのソフトウェア
 - 組版するためのプログラミング言語

T_EX とは何か

- 1978 年に Donald E. Knuth が発表
 - 相当に古い
- 組版システム
 - 組版するためのソフトウェア
 - 組版ためのプログラミング言語
- L^AT_EX
 - T_EX とは別物
 - T_EX のマクロ体系（フォーマット）

T_EX の仲間にはたくさんある (ナトカT_EX)

- 処理系.....T_EX (ソフトウェア) を拡張したもの
ε-T_EX, pdfT_EX, X_YT_EX, LuaT_EX, pT_EX, upT_EX など
- フォーマット.....マクロ体系
L^AT_EX, plain T_EX, ConT_EXt など

全部まとめてT_EXと呼ぶことも多い

T_EX ディストリビューション

T_EX に関する成果物は、CTAN に集められる

- <https://www.ctan.org>
- ボランティアで成り立っている

CTAN から様々なディストリビューション (配布元) へ

- T_EX Live (<http://www.tug.org/texlive/>)
- W32TeX (<http://w32tex.org/>)¹
- MiK_TE_X (<https://miktex.org>)²

¹T_EX Live がベース

²日本語できない模様

T_EX ディストリビューション

T_EX 本体やパッケージ以外にも、関連するバイナリも収録されている

- dvi ウェア (後述)
- kpathsea (後述)
- texdoc

texdoc → ドキュメントを検索するコマンド

例: `texdoc latex`

`texdoc platexsheet-jsclasses`

でコマンド一覧を表示

ワトソン氏 (朝倉卓人) 作成

最も普及している T_EX ディストリビューション

膨大な数のパッケージやバイナリが含まれる

晩春に名前が変わる大型アップデート

2 月頃に更新停止 (frozen) ・次年度版の pretest

2020 年 4 月 10 日 T_EX Live 2019 → T_EX Live 2020

バイナリの更新は原則**大型アップデート時のみ**

パッケージ (テキストファイル) の更新は frozen 時以外はいつでも

大型アップデート時はインストールし直す必要

TEXのインストール



美文書 付録 A



T_EX のインストール

Windows なら W32TeX
それ以外なら T_EX Live をインストール

詳しくは T_EX Wiki³
または <http://www.circle9.work/tex/install.html>

インストールには数時間かかります。

³<https://texwiki.texjp.org>

最新の T_EX Live をインストールする—UNIX 系の場合

TUG⁴ からインストーラをダウンロード⁵

```
wget http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unix.tar.gz
```

インストーラを起動してインストール

```
sudo ./install-tl -no-gui \  
-repository http://mirror.ctan.org/systems/texlive/tlnet
```

ネットワーク経由なので電波の良いところで

⁴T_EX User Group <ftp://ftp.tug.org/texlive/tlnet/>

⁵ミラーサイトを利用しましょう

T_EX Live のアップデート

(T_EX Live をインストールした場合)

```
sudo tlmgr update --self --all
```

上のコマンドで T_EX Live をアップデート

定期的にやろう

年度が変わる大型アップデート時には再インストール

TEXの使い方

👤 美文書 2 章 👤

T_EX の使い方

自分で書いた T_EX ソースを、T_EX 処理系に処理させることで、PDF ファイルを得る。

⁶新しい処理系には、直接 PDF を出力するものもある。
例: pdfT_EX, LuaT_EX, X_YT_EX

T_EX の使い方

自分で書いた T_EX ソースを、T_EX 処理系に処理させることで、PDF ファイルを得る。

のだが

⁶新しい処理系には、直接 PDF を出力するものもある。
例: pdfT_EX, LuaT_EX, X_YT_EX

T_EX の使い方

自分で書いた T_EX ソースを、T_EX 処理系に処理させることで、PDF ファイルを得る。

のだが

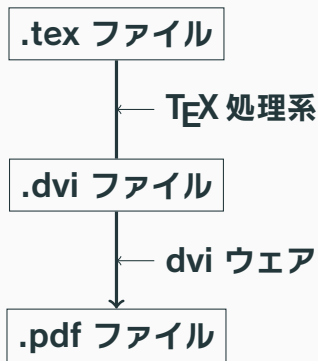
歴史的経緯で、T_EX 処理系は、PDF ファイルではなく、**dvi ファイル**を出力する⁶。

出てきた dvi ファイルを dvi ウェアで処理することによって、最終的な PDF を得ることができる。

⁶新しい処理系には、直接 PDF を出力するものもある。
例: pdfT_EX, LuaT_EX, XeT_EX

PDF を得る流れ

T_EX 処理系 文字の座標を決める
dvi ウェア 実際に文字を配置する



dvi ウェア

dvi ← Device Independent (装置非依存)

dvi ウェア.....dvi ファイルを変換するソフトウェア

- **dvipdfmx (PDF に変換)**
- **dvips (PostScript に変換)**

dvi の仕様

標準仕様 装置非依存な部分。dvi ウェアで共通。

拡張仕様 装置依存な部分⁷。dvi ウェアごとに異なる。

⁷色とか用紙サイズとか

T_EX ソースを書くうえでの注意点

使う T_EX 処理系、dvi ウェアによって書き方が微妙に違う

どの処理系、どの dvi ウェアを利用するか気に留める必要

⁸Lua_AT_EX は (u)p_AT_EX と書き方がそれなりに異なるので注意

T_EX ソースを書くうえでの注意点

使う T_EX 処理系、dvi ウェアによって書き方が微妙に違う

どの処理系、どの dvi ウェアを利用するか気に留める必要

日本で一般的な方法

- pL_AT_EX + dvipdfmx
- upL_AT_EX + dvipdfmx
- LuaL_AT_EX (最近広まりつつある)

以下、主に pL_AT_EX + dvipdfmx を例にして話す⁸

⁸LuaL_AT_EX は (u)pL_AT_EX と書き方がそれなりに異なるので注意

L^AT_EX の書き方

👤 美文書 3 章 👤

Listing 1: sample.tex

```
1 \documentclass[12pt,dvipdfmx]{jsarticle}
2 \usepackage[T1]{fontenc}
3 \usepackage{graphicx,xcolor}
4 \usepackage{otf}
5 \usepackage{newpxtext,newpxmath}
6 \usepackage{amsmath}
7 \usepackage[a6paper]{geometry}
8 \begin{document}
9 吾輩は\TeX である。名前はまだない。
10 \[e^{i\pi}=-1\]
11 \end{document}
```

BOM なし UTF-8 で保存しましょう

L^AT_EX 文書の作り方

コマンドラインで以下を実行

```
platex sample
```

```
dvipdfmx sample
```

吾輩は T_EX である。名前はまだない。

$$e^{i\pi} = -1$$

コマンド

命令 \ で始まる、英文字（と和文文字）の列

もしくは \ のあとに数字か記号ひとつ

\TeX や \^ など（コントロールワード 制御語 と コントロールシンボル 制御文字）

環境 \begin{ナントカ} と \end{ナントカ} で囲まれたもの

コメント % から行末まではコメント扱い（無視される）

特殊な文字

以下の文字は特殊文字

% \ ^ _ ~ { } # & \$

コマンド 命令についての注意

- ^{コマンド}命令の引数は { } で括る
- オプショナルな引数は [] で括る

{ } はカッコの対応を確認されるが、
[] はカッコの対応を確認しない

例:

`\lstinputlisting[caption=[1]]{foo.tex}` は
caption=[1 **だけが** [] に入っている判定
→ [] に含めたい全体を { } で括ると解決
`\lstinputlisting[{caption=[1]}]{foo.tex}`

L^AT_EX 文書の構造

```
1 % クラスファイル (jsarticle.cls) を読み込む
2 \documentclass[dvipdfmx]{jsarticle}
3
4 % プリアンブル
5 %     パッケージ (nntoka.sty) の読み込みや
6 %     文書全体の設定
7
8 \begin{document}
9 %     文書本体
10 \end{document}
```


L^AT_EX 文書の構造 (ドキュメントクラス)

```
\documentclass[dvipdfmx]{jsarticle}
```

クラスファイルを読み込む → 版面構成の定義など
実体は `natbib.cls` というテキストファイル

主要なクラスファイル

- `jsarticle`, `jsreport`, `jsbook` (新ドキュメントクラス)
- `jlreq` (日本語組版処理の要件⁹対応)
- `beamer` (スライド用 日本語するには工夫が必要)
- `jarticle`, `jreport`, `jbook` (s なし) は**非推奨**

⁹<https://www.w3.org/TR/jlreq/ja/>

L^AT_EX 文書の構造 (ドキュメントクラス)

```
\documentclass[dvipdfmx]{jsarticle}
```

[] 中はオプション設定
フォントサイズ、見開きの設定など

```
\documentclass[12pt,dvipdfmx]{jsarticle}
```

必ず使う dvi ウェアをオプションに設定する
(ドライバオプション)¹⁰

¹⁰dvi 拡張仕様の命令を dvi ファイルに埋め込む必要があるため

L^AT_EX 文書の構造 (文書本体)

```
\begin{document}  
:  
\end{document}
```

文書本体は `\begin{document}` と `\end{document}` の間に書く

打ち込んだ文字がそのまま出力される (特殊文字は除く)

コマンド
命令を利用できる

文書を書くときの注意

改行の扱い

- 改行は空白扱い
- 和文文字直後の改行は無視（空白にもならない）
- 連続した改行 → 改段落
- % は改行文字も含めて、行末まで無視する
→ 空白は入らない

メール的なフォーマットで書ける

1 行が長くなったら改行、改段落は空行

文書を書くときの注意

コントロールシーケンス

制 御 綴 や空白の扱い

- 空白はいくつつなげても 1 つに吸収される
- 行頭行末の空白は無視される
- `_` や `~` で空白を出力できる (`~` は行分割されない)
- コントロールワード 制 御 語 コントロールワード 直後の空白は 制 御 語 の区切りでしかない
→ 無視される
- コントロールシンボル 制 御 文 字 直後の空白は無視されない

<code>\TeX_Live</code>	→	<code>T_EXLive</code>
<code>\TeX_Live</code>	→	<code>T_EX Live</code>
<code>A_&B</code>	→	<code>A & B</code>

<code>\TeX_Live</code>	→	<code>T_EXLive</code>
<code>{\TeX}_Live</code>	→	<code>T_EX Live</code>
<code>A\&B</code>	→	<code>A&B</code>

文書を書くときの注意

その他の注意、使える命令、環境は

- `texdoc platexsheet-jsclasses`
- 美文書作成入門

を参照

ググるより先に上を読みましょう

ググって出てくる情報は軒並み古くて怪しい¹¹

¹¹ディスプレイ数式を `$$ ~ $$` で囲んだり、`\begin{eqnarray}` を使ったり

L^AT_EX 文書の構造 (プリアンブル)

`\documentclass` から `\begin{document}` の間

→ **プリアンブル (preamble)**

パッケージの読み込み・文書全体の設定をする

`\usepackage[a4paper]{geometry}`

→ `geometry` パッケージを、`a4paper` オプション付きで読み込む

本文を書くことはできない

逆に、プリアンブルでしか使えないコマンドもある

`\usepackage` など

パッケージとは

様々な便利機能を提供

他のプログラミング言語で言うところのライブラリ

実体は、`natcas.sty` というテキストファイル

例: ゆきだるま 🌨️ を書きたい!

→ `scsnowman` パッケージ

→ `\usepackage{scsnowman}`

→ `\scsnowman[scale=3,hat,arms,buttons]`



素敵!

パッケージの使い方

1. 用途からパッケージを探す

ググるしかない もしくは CTAN でググる¹²

2. プリアンブルで

`\usepackage[オプション]{パッケージ名}`

3. 使う

4. 使い方がわからなくなるので `texdoc パッケージ名`

¹²英語なので厳しい; ググるを誤用してるのは承知です

おすすめプリアンブル

```
1 % フォントエンコード ( 文字化けしないように )
2 \usepackage[T1]{fontenc}
3 % 図の挿入、色を扱う
4 \usepackage{graphicx,xcolor}
5 % フォントをイイカンジにしてくれる
6 \usepackage{otf}
7 % フォントを変更 ( デフォルトはサイズ指定に不具合 )
8 \usepackage{newpxtext,newpxmath} % Palatino
9 %%% \usepackage{newtxtext,newtxmath} % Times
10 %%% \usepackage{lmodern} % Latain Modern
11 % 数学するなら必要
12 \usepackage{amsmath}
13 % 用紙サイズの設定
14 \usepackage[a4paper]{geometry}
```

LaTeX を理解するまでは、これをそのまま使おう

書き方まとめ

```
1 \documentclass[12pt,dvipdfmx]{jsarticle}
2 % プリアンブル
3 \usepackage[T1]{fontenc}
4 \usepackage{graphicx,xcolor}
5 \usepackage{otf}
6 \usepackage{newpxtext,newpxmath}
7 \usepackage{amsmath}
8 \usepackage[a4paper]{geometry}
9
10 \begin{document}
11   ドキュメント本文
12 \end{document}
```

(u)p_ΛT_EX VS Lua_ΛT_EX

Listing 2: sample-lualatex.tex

```
1 \documentclass[12pt]{ltjsarticle}
2 \usepackage[no-math]{fontspec}
3 \usepackage[deluxe,haranoaji]{luatexja-preset}
4 \usepackage{graphicx,xcolor}
5 \usepackage{newpxtext,newpxmath}
6 \usepackage{amsmath}
7 \usepackage[a6paper]{geometry}
8 \begin{document}
9 吾輩は\TeX である。名前はまだない。
10 \[e^{i\pi}=-1\]
11 \end{document}
```

Lua_ΛT_EX を利用する場合

- jsclasses は p_ΛT_EX 専用 → ltjsclasses
- ドライバオブションは不要
- フォントの世話: fontenc → fontspec
- otf パッケージも p_ΛT_EX 専用 → 削除

エラーへの対処

 美文書 2 章 9 節 

⚠️ご注意ください⚠️



**エラー対処が上手かどうかで
作業効率が激変します**



エラーに遭遇する

$\text{T}_{\text{E}}\text{X}$ はプログラミング言語
書き方を間違えるとエラーが出る

`\TEX` と書いてしまうと.....

エラーに遭遇する

T_EX はプログラミング言語
書き方を間違えるとエラーが出る

\TEX と書いてしまうと.....

! Undefined control sequence.

1.3 \TEX

エラーに遭遇する

T_EX はプログラミング言語
書き方を間違えるとエラーが出る




`\TEX` と書いてしまうと.....

! Undefined control sequence.

1.3 `\TEX`

「?」と聞かれるので、   のどれかを押す

エラーが出たら

-  処理を中断して終了
-  処理を継続、ログは標準出力しない
-  処理を継続、再びエラーが出ると止まる

 を数回連打するのがおすすめ

大抵、複数のエラーが混入しているため

連続して 5 回以上エラーが出てきたら  するべし

? 以外のプロンプトの場合

Enter file name:

`\usepackage` でパッケージ名を間違えたときに出がち

X を押して ↵

*

`\end{document}` を忘れたときに出がち

1. `\stop` と打って ↵

2. `\aaa` (未定義の コントロールシークエンス 制御綴) を打って ↵

→ ? のプロンプト → **X**

3. **ctrl** + **C**

エラーメッセージの見方

! You can't use 'macro parameter character #' in horizontal mode.

1.3 O-oooooooooooo #

AAAAE-A-A-I-A-U-

?

エラーメッセージの見方

! You can't use 'macro parameter character #' in horizontal mode.

1.3 O-oooooooooooo #

AAAAE-A-A-I-A-U-

?

! エラーメッセージ

1. 行数 \TeX が読み込んだもの

まだ読み込んでいないもの

エラーが出た行に戻って治せばいいのだが.....

エラーへの対処

大体のエラーの原因

- ・ コントロールシーケンス 制御 綴 の綴りのマチガイ
- ・ 環境の閉じ忘れ
- ・ ものの不均衡 (`{ }`、`$ $13`、`\left \right` など)
- ・ コマンド 命令の用法のマチガイ

エラーが起きた行付近で上がらないか確認

コマンド
命令の用法のマチガイ → `texdoc <パッケージ名>` で確認

¹³`$ $` よりも、`\(\)` を使うほうが好ましいとされます。(「数式組版」(木枝祐介 ラムダノート株式会社 (2018)))

対処しにくいエラー

おさらい

\LaTeX は \TeX のフォーマット (マクロ体系)

→ \LaTeX レベルのエラーと、 \TeX レベルのエラーがある

起きたエラーによっては、原因が特定しにくい

例: ! Missing number, treated as zero.

対処しにくいエラー

おさらい

\LaTeX は \TeX のフォーマット（マクロ体系）

→ \LaTeX レベルのエラーと、 \TeX レベルのエラーがある

起きたエラーによっては、原因が特定しにくい

例: ! Missing number, treated as zero.

処理中に外部ファイルを読み込むこともある

→ 行番号が、どのファイルの行番号かわからなくなる

エラーを起こさないために

- ・ タイプセットを細かく行う
- ・ 開いた環境はすぐ閉じる
- ・ 全角空白「 」を使わない
段落頭の字下げは `\parindent` で設定

欧文クラスで、一番最初のパラグラフを字下げしたい場合 → `indentfirst` パッケージ

- ・ `\verb` 命令もなるべく避ける
コマンド
命令の引数にあるとエラー (`\verb` の呪い)

エラーを起こさないために

- ・ タイプセットを細かく行う
- ・ 開いた環境はすぐ閉じる
- ・ 全角空白「 」を使わない
段落頭の字下げは `\parindent` で設定

欧文クラスで、一番最初のパラグラフを字下げしたい場合 → `indentfirst` パッケージ

- ・ `\verb` 命令もなるべく避ける
コマンド
命令の引数にあるとエラー (`\verb` の呪い)

それでも意味不明なエラーが起きる

パッケージの衝突

```
1 \documentclass{jsarticle}
2 %%% 略
3 \usepackage{mathabx} % いろんな記号を使いたい
4 \usepackage{yhmath} % 大きいカッコを綺麗にしたい
5 \begin{document}
6 \[e^{i\pi}=-1\]
7 \end{document}
```



! LaTeX Error: Command `\iint` already defined.
Or name `\end...` illegal, see p.192 of the manual.
1.645 ...d{\iint}{\DOTSI\protect\MultiIntegral{2}}

パッケージの衝突

```
1 \documentclass{jsarticle}
2 %%% 略
3 \usepackage{mathabx} % いろんな記号を使いたい
4 \usepackage{ymath} % 大きいカッコを綺麗にしたい
5 \begin{document}
6 \[e^{i\pi}=-1\]
7 \end{document}
```



! LaTeX Error: Command `\iint` already defined.
Or name `\end...` illegal, see p.192 of the manual.
1.645 ...d`\iint``\DOTSI\protect\MultiIntegral{2}`

mathabx と ymath が同じ^{コマンド}命令を定義 → エラー

パッケージを読み込む順番を変えたら誤魔化せる場合も

→ 読み込む順番を変えてみる

→ どうしようもなければ諦める

パッケージが日本語対応してなくてエラーが起きる場合も

→ (u)pL^AT_EX なら plautopatch パッケージ¹⁴を試す

→ LuaL^AT_EX なら日本語非対応の問題はおこりにくい

¹⁴<https://aminophen.github.io/slide/hytexconf18.pdf>

エラーが解消できなくてどうしようもないときは

とりあえずエラーメッセージでググってみる

これで解決できたら苦労しないんだよねあ わかりにくいエラーメッセージが嫌ならば、SATySFj.....?

わからなければ詳しい人に聞く

TeX Forum¹⁵ で質問

Twitter でつぶやくのも実は有用

実はバグを踏んでいる可能性も

¹⁵<https://oku.edu.mie-u.ac.jp/tex/>

わかりにくいエラー①

[a] 真鍋 \ [b] いつき

→! Missing number, treated as zero.

\ (強制改行) 命令は、実はオプション引数をもつ

→\ [<長さ>]

\{ } のように { } で区切ると解決

[a] 真鍋 \{ } [b] いつき

→[a] 真鍋

[b] いつき

わかりにくいエラー②

```
\section{$\overrightarrow{\mbox{ぶーん}}}$}
```

→! Illegal parameter number in definition of \reserved@a.

エラーが起きる原因 → 🤔 16

`\section` や `\caption` で変なエラーが出たら、
引数に入ってるヤバそうな命令に `\protect` を前置
コマンド

```
\section{$\protect\overrightarrow{\mbox{ぶーん}}}$}
```

→ $\overrightarrow{\text{ぶーん}}$

¹⁶`\section` の引数は動くので、脆弱な`\overrightarrow` は保護しなければならない

TEXのディレクトリ構成

🐼 美文書 付録 B 3 節 🐼

TEXMF ツリー

T_EX 関連ファイルを入れるディレクトリ構成

TEXMF ← T_EX+ METAFONT¹⁷

複数の TEXMF ツリーを使い分けるのが主流
多重 TEXMF ツリー

確認方法: `kpsewhich -var-value TEXMF`

¹⁷METAFONT は Knuth が作ったフォント記述言語

多重 TEXMF ツリーの利点

ディストリビューションが用意したファイルと、自分がインストールしたファイルを分離できる

ディストリビューションを更新しても、自分のインストールしたファイルは削除されない

ディストリビューションが用意したファイル

→ `kpsewhich -var-value TEXMFDIST`

自分がインストールするファイル

→ `kpsewhich -var-value TEXMFLOCAL`

全ユーザーが使える

→ `kpsewhich -var-value TEXMFHOME`

そのユーザーが使える

kpathsea ライブラリ と ^{ファイル一覧}ls-R

kpathsea ライブラリ

TEXMF ツリーからファイルを検索する

Karl Berry 氏によって作られた path searching

例: `kpsewhich hmtrump.sty`

→/usr/local/texlive/texmf-local/tex/latex/local/hmtrump.sty

TEXMF ツリーに作られた ^{ファイル一覧}ls-R を見て検索する
TEXMF ツリーに変更 → ^{ファイル一覧}ls-R を更新する必要¹⁸

`sudo mktexlsr`

¹⁸ls-R を使わない運用方法もあるらしいですが、やったことがないのでわかりません

パッケージをインストールする

ディストリビューションに含まれないパッケージを使いたい
→ 自分で TEXMF ツリー (TEXMFLOCAL) に入れる必要
作業ディレクトリに置いてもよいけれども

正しい場所に入れなければ正常に使えない¹⁹

¹⁹拡張子に応じて、検索するディレクトリを決め打ってるため

TEXMF ツリーの構造

TEXMFLOCAL²⁰ の構造を覗いてみる²¹

```
tree -d -L 2 /usr/local/texlive/texmf-local
```

ファイルの種類ごとに分類

doc ドキュメント (説明書)

tex パッケージの本体など

font フォント関連

さらにサブディレクトリで分類

そのなかでパッケージごとに分類

```
/usr/local/texlive/texmf-local
├── doc
│   ├── fonts
│   ├── latex
│   └── luatex
├── source
│   └── latex
├── tex
│   ├── latex
│   ├── luatex
│   └── plain
├── texdoc
├── tlpkg
└── web2c
```

²⁰TeX Live on UNIX の標準では /usr/local/texlive/texmf-local

²¹<https://texwiki.texjp.org/?TeX%20のディレクトリ構成参照>

パッケージをインストールする場所

前述の通り、分類して TEXMFLOCAL に配置

まずはパッケージドキュメントを確認

ドキュメントに記載がない場合: あまり失敗しない方法
以下にディレクトリを掘ってファイルを配置

- ドキュメント → `$TEXMFLOCAL/doc/latex/<pkgname>`
- `*.dtx`, `*.doc` → `$TEXMFLOCAL/source/latex/<pkgname>`
- その他 → `$TEXMFLOCAL/tex/latex/<pkgname>`

フォント関連などはもっと複雑

とりあえず美文書は読んでください
もっと詳しく知りたい場合

- T_EX Wiki

<https://texwiki.texjp.org>

- Acetaminophen's diary

<http://acetaminophen.hatenablog.com>

以下のブログは、もっと沼にハマりたい人向け

- ラングラグー

<https://blog.wtsnjp.com>

- マクロツイーター

<https://zrbabbler.hatenablog.com>