npTEX の開発（方針）について

北川 弘典

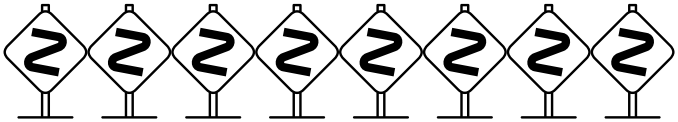
2023–11–11

TEXConf 2023

目次

注意

このスライドのように，
タイトルが白抜きになっているものは
発表では原則飛ばします．



スライドはどこかにアップロードの**予定**

はじめに：最近の LaTeX 2ε

LaTeX 2ε 2020-02-02 以降，expl3 が
フォーマット内に読み込まれるように

- LaTeX 2ε が徐々に expl3 の機能を用いて書き直される

はじめに：最近の LaTeX 2ε

LaTeX 2ε 2020-02-02 以降，expl3 が
フォーマット内に読み込まれるように

- LaTeX 2ε が徐々に expl3 の機能を用いて書き直される

  例　フック機能の統一化 (2020-10-01)
  everyshi, filehook, atbegshi, ... が「ほぼ不要」になった

はじめに：最近の LaTeX 2ε

LaTeX 2ε 2020-02-02 以降，expl3 がフォーマット内に読み込まれるように

- LaTeX 2ε が徐々に expl3 の機能を用いて書き直される

  例　フック機能の統一化 (2020-10-01)
  everyshi, filehook, atbegshi, ... が「ほぼ不要」になった

- Tagged PDF project

Frank Mittelbach and Chris Rowley. “LaTeX Tagged PDF—A blueprint for a large project”, TUGboat **41**:3, 2020.

ところが……


```
\RequirePackage{pdfmanagement-testphase}
\DocumentMetadata{backend = dvipdfmx}
\documentclass{article}
\usepackage{hyperref,pxjahyper}
\hypersetup{pdftitle={日本語}}
```

ところが……


```
\RequirePackage{pdfmanagement-testphase}
\DocumentMetadata{backend = dvipdfmx}
\documentclass{article}
\usepackage{hyperref,pxjahyper}
\hypersetup{pdftitle={日本語}}
```

結果 ((u)pLaTeX <2023-02-14u04>+1)

```
! LaTeX Error: Invalid UTF-8 string: missing continuation
   byte (x3).
```

ところが……


```
\RequirePackage{pdfmanagement-testphase}
\DocumentMetadata{backend = dvipdfmx}
\documentclass{article}
\usepackage{hyperref,pxjahyper}
\hypersetup{pdftitle={日本語}}
```

結果 ((u)pLaTeX <2023-02-14u04>+1)

```
! LaTeX Error: Invalid UTF-8 string: missing continuation
   byte (x3).
```

“Japanese characters in metadata on (u)pLaTeX”, aminophen, latex3/pdfresources/#18, 2021-06-14

ところが……


```
\RequirePackage{pdfmanagement-testphase}
\DocumentMetadata{backend = dvipdfmx}
\documentclass{article}
\usepackage{hyperref,pxjahyper}
\hypersetup{pdftitle={日本語}}
```

結果 ((u)pLaTeX <2023-02-14u04>+1)

```
! LaTeX Error: Invalid UTF-8 string: missing continuation
   byte (x3).
```

2 年以上前
今も未解決

“Japanese characters in metadata on (u)pLaTeX”, amin-phen, latex3/pdfresources/#18, 2021-06-14

(u)pTEX の特殊性：和文文字トークン

(u)pTEX は**欧文部分は 8 bit エンジン**だが
和文文字トークンという「異質なもの」がある：

(u)pTEX の特殊性：和文文字トークン

(u)pTEX は**欧文部分は 8 bit エンジン**だが
和文文字トークンという「異質なもの」がある：

- 0–15 の範囲の \catcode を**持たない**
- 文字コードが **256 以上になりうる**
- 「\lowercase トリック」等が不可能

(u)pTEX の特殊性：和文文字トークン

(u)pTEX は**欧文部分は 8 bit エンジン**だが
和文文字トークンという「異質なもの」がある：

- 0–15 の範囲の \catcode を**持たない**
- 文字コードが **256 以上になりうる**
- 「\lowercase トリック」等が不可能

→**和文文字トークンを考慮しないパッケージ**は，
(u)pLATEX で動かないことがある

とくに expl3 の一部

(u)pTEX の特殊性：和文文字トークン

前スライドの通り，比重がますます増しているexpl3では「和文文字トークンを満足にサポート」とは言えない．

(u)pTEX の特殊性：和文文字トークン

前スライドの通り，比重がますます増している expl3 では「和文文字トークンを満足にサポート」とは言えない．

しかし，今の日本語 TEX 開発コミュニティには，
和文文字トークンについて

- パッチを自前で書く
- LATEX チーム他に事情説明する

だけの余力はない．

(u)pTEX の特殊性：和文文字トークン

話は聞かせてもらったぞ！

(u)pLATEX は滅亡する！

和文文字トークンの「厄介な」挙動

- l3regex において，和文文字トークン
  「±」(U+00B1) と UTF-8 列中のバイト 𝔅1 が区別されない

  「[l3regex] [upTeX] 和文文字と欧文バイトがマッチしてしまう」，h20y6m，h20y6m/plexpl3/#2, 2022-09-02

- TEXConf 2019 に話す予定だったネタも
  無関係ではなかった

  pTEX 4.0.0 (TL2022) で解決

  Hironori Kitagawa, “Distinguishing 8-bit characters and Japanese characters in (u)pTEX”, TUGboat **41**:3, 2020.

次を目標とした新エンジンの開発を計画中：

「[npTeX] 欧文トークンに似せた CJK トークン」，t-tk，
tex-jp-build/#150, 2022-10-16.

- LaTeX 本体の変更に伴う保守が「楽に」行える
- 従来の upLaTeX 用クラス・パッケージが
  （ほぼ）そのまま処理できる

次を目標とした新エンジンの開発を計画中：

「[npTeX] 欧文トークンに似せた CJK トークン」，t-tk，
tex-jp-build/#150, 2022-10-16.

- LaTeX 本体の変更に伴う保守が「楽に」行える
- 従来の upLaTeX 用クラス・パッケージが
  （ほぼ）そのまま処理できる

名称だけは **npTEX** と決まっている．

→ ttk さんの案．new pTEX, next pTEX, novel pTEX,
notable pTEX, nippon (p)TEX, nihongo pTEX

名称は決まったが……

npTEX については，名称以外は

- 内部コードは Unicode　　入力は EUC/Shift_JIS も……
- 1 文字について 1 つの文字トークンを生成
- **文字トークン段階**では，欧文・和文の区別は必須でない　　LuaTEX-ja はそれでやっている

以外はほとんど決まっていない．

名称は決まったが……

npTEX については，名称以外は

- 内部コードは Unicode　　入力は EUC/Shift_JIS も……
- 1 文字について 1 つの文字トークンを生成
- **文字トークン段階**では，欧文・和文の区別は必須でない　　LuaTEX-ja はそれでやっている

以外はほとんど決まっていない．

以下**「何を悩んでいるのか」**を説明．

トークン以外でどう欧文・和文を区別？

文字ノードを１つずつ作っていく際に，
その文字コードの \cjkxcode（仮称）の値を見て判定

従来の \kcatcode で一度に制御された属性を
\cjkxcode ではビットで分割してもよい：

- 欧文文字ノードか和文文字ノードか
- \jchrwidowpenalty の挿入対象か
- 直後の改行は半角空白を発行するか

LuaTEX-ja では
attribute
kcatcode 最下位
欧文/和文に連動

大きな開発方針は 2 案

i. ε-upTEX から文字トークンの扱いを変える

2022.10 に途中まで試して放置

ii. XETEX に pTEX の和文組版機能を載せる

2023.8 に試作品→そこから放置

- XETEX の「pdf 出力部」は `xdvipdfmx`
- 「fontspec, unicode-math は欧文のみ」
  「和文フォントは JFM のみ」までが現実的

どこまで「過去を見る」か？

前提　既存の TEX ソースから組版を変えたくないのなら

TEX 環境 (TEX Live xxxx) ごと保存すべき.

どこまで「過去を見る」か？

前提　既存の TEX ソースから組版を変えたくないのなら
TEX 環境 (TEX Live xxxx) ごと保存すべき.

今新しく文書を作成する際に，
（数式以外で）8 bit フォントを用いる必要性はどの程度？

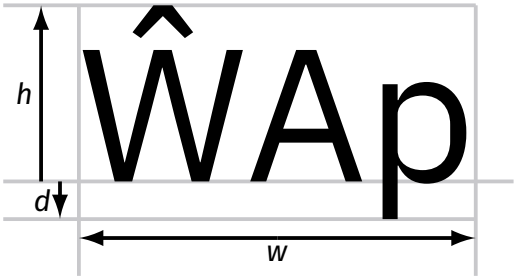
どこまで「過去を見る」か？

前提　既存の TeX ソースから組版を変えたくないのなら
TeX 環境 (TeX Live xxxx) ごと保存すべき.

今新しく文書を作成する際に，
（数式以外で）8 bit フォントを用いる必要性はどの程度？

- Type 1 フォントのサポート終了記事：Adobe, Microsoft
- dvips による PostScript 出力の必要性？
- フォントによる組版の差異をどこまで許容？

フォントによる組版の差異の例



	enc	*h*	*d*	*w*
cmr10	OT1	9.47221	1.94444	23.33339
ecrm1000	T1	8.9978	1.94397	23.32764
lmroman10-regular	TU	8.69	1.94	22.23

LuaTEX-ja への移行の 3 ハードル

pTEX 系列から Unicode TEX への移行

LuaTEX-ja への移行の 3 ハードル

1. fontspec など，Unicode LATEX の流儀

pTEX 系列から Unicode TEX への移行

LuaTEX-ja への移行の 3 ハードル

1. fontspec など，Unicode LATEX の流儀
2. マクロパッケージであることによる制限
   - \ltjsetparameter{xkanjiskip=...} など
   - 単位 Q, H, zw, zh なし

pTEX 系列から Unicode TEX への移行

LuaTEX-ja への移行の 3 ハードル

1. fontspec など，Unicode LATEX の流儀
2. マクロパッケージであることによる制限
   - \ltjsetparameter{xkanjiskip=...} など
   - 単位 Q, H, zw, zh なし
3. メモリ食い・遅い

pTeX 系列から Unicode TeX への移行

LuaTeX-ja への移行の 3 ハードル

1. fontspec など，Unicode LaTeX の流儀
2. マクロパッケージであることによる制限
   - `\ltjsetparameter{xkanjiskip=...}` など
   - 単位 `Q`, `H`, `zw`, `zh` なし
3. メモリ食い・遅い

「XeTeX ベースの npTeX」では 2, 3 はない

→ Unicode TeX への移行を促せるか？

LaTeX における，Unicode 直接入力

	a	b	c
トークン	バイト単位	Unicode 文字	Unicode 文字
フォント	8 bit	8 bit	Unicode
直接入力実現法	アクティブ文字 (inputenc)		—

- a 8 bit エンジン上 (pdfLaTeX, (u)pLaTeX)
- b Unicode LaTeX + T1
- c XeLaTeX, LuaLaTeX (Unicode LaTeX) 標準

LaTeX における，Unicode 直接入力

	a	b	c
トークン	バイト単位	Unicode 文字	Unicode 文字
フォント	8 bit	8 bit	Unicode
直接入力実現法	アクティブ文字 (inputenc)		—

- a 8 bit エンジン上 (pdfLaTeX, (u)pLaTeX)
- b Unicode LaTeX + T1, **ε-upTeX ベースの npTeX**
- c XeLaTeX, LuaLaTeX (Unicode LaTeX) 標準

LaTeX における，Unicode 直接入力

	a	b	c
トークン	バイト単位	Unicode 文字	Unicode 文字
フォント	8 bit	8 bit	Unicode
直接入力実現法	アクティブ文字 (inputenc)	**未考慮**	–

a 8 bit エンジン上 (pdfLaTeX, (u)pLaTeX)

b Unicode LaTeX + T1, **ε-upTeX ベースの npTeX**

c XeLaTeX, LuaLaTeX (Unicode LaTeX) 標準

8 bit エンジンでの Unicode 直接入力

PQ でバイト 0xPQ を，hoge で名称が hoge の制御綴を表す

1. UTF-8 上位バイト (80–FF) をアクティブ化 (latex.ltx)
2. 各上位バイトに「UTF-8 列を読み取る命令」を割り当て (utf8.def)

C2 9F ⟶ \UTFviii@two@octets C2 9F ⟶ ⋯ ⟶ u8:C2 9F

3. Unicode 文字の実際の出力命令を定義 (\DeclareUnicodeCharacter, utf8enc.dfu)

u8:C2 9F ⟶ \IeC{\ss} ⟶ \ss

トークンは Unicode，フォントは 8 bit

\DeclareUnicodeCharacter などを再定義し，
「各 Unicode 文字を個別にアクティブ化，出力命令割当」は？

ß ⟶ ⋯ ⟶ \IeC{\ss} ⟶ \ss

トークンは Unicode，フォントは 8 bit

\DeclareUnicodeCharacter などを再定義し，
「各 Unicode 文字を個別にアクティブ化，出力命令割当」は？

ß ⟶ ⋯ ⟶ \IeC{\ss} ⟶ \ss

本家 LaTeX 2ε で対応されるのが望ましい

トークンは Unicode，フォントは 8 bit

\DeclareUnicodeCharacter などを再定義し，
「各 Unicode 文字を個別にアクティブ化，出力命令割当」は？

ß ⟶ ⋯ ⟶ \IeC{\ss} ⟶ \ss

本家 LaTeX 2ε で対応されるのが望ましい

懸案事項　「\Gödel」「\straße」など

アクティブ化以外の実装？

\nptexnoderecipe *c* = ⟨***token list***⟩

⟨*token list*⟩ が空でない場合，水平モード中の
「文字コード *c*，\catcode 11 の文字トークン」は
「文字コード *c* の文字ノード」を作るのではなく，
⟨*token list*⟩ のように振る舞う．

- 「\nptexnoderecipe *c*」が非空かどうか見分ける手段は？
- \accent との兼ね合い？

パッケージ開発者への説明

LaTeX や多くのパッケージでは，現状次のような認識：

\sys_if_engine_luatex:p ∨ \sys_if_engine_xetex:p

⟺ （\Umathcode 定義済）

⟺ （Unicode 1 文字でいつでも 1 トークン）

⟺ （本文の欧文フォントは Unicode）

パッケージ開発者への説明

LaTeX や多くのパッケージでは，現状次のような認識：

\sys_if_engine_luatex:p ∨ \sys_if_engine_xetex:p

⟺ （\Umathcode 定義済）

⟺ （Unicode 1 文字でいつでも 1 トークン）

⟺ （本文の欧文フォントは Unicode）

「ε-upTeX ベースの npTeX」や，「XeTeX ベースの npTeX，本文 8 bit フォント」では上記の**認識は誤りとなる**.

XeTeX ベースにする場合の懸案事項

XeTeX 互換 (\sys_if_engine_xetex:p) を謳うなら
LaTeX やパッケージの改変は少なくて済むが，

- XeTeX 拡張と pTeX 拡張の干渉への対応
  - 入力の文字コード（次スライド）
  - \XeTeXinterchartoken

XeTeX ベースにする場合の懸案事項

XeTeX 互換 (\sys_if_engine_xetex:p) を謳うなら
LaTeX やパッケージの改変は少なくて済むが，

- XeTeX 拡張と pTeX 拡張の干渉への対応
  - 入力の文字コード（次スライド）
  - \XeTeXinterchartoken
- dvips を想定するなら，「**dvi 互換モード**」必要
  → 「ε-upTeX ベースの npTeX」実装と実質的に同じ作業

入力の文字コード

- **pTEX 系列 (ptexenc)**
  文字コード　EUC, Shift_JIS, ISO-2022-JP, UTF-8
  　　ただし，ISO-2022-JP は常に認識
  Unicode 正規化　濁点・半濁点付き仮名を合成
- **XETEX**
  文字コード　UTF-8，UTF-16，バイト単位，ICU
  Unicode 正規化　無変換，NFC，NFD から選択

両者のマージは大変そう→ \{epTeX / XeTeX\}inputencoding の値で二者択一？

余談：“native font” にしか効かない

「和文フォントは従来通り JFM のみ」とすれば，
これらは pTEX 由来の和文組版処理とは干渉せず

- \XeTeXdashbreakstate, \XeTeXlinebreak{locale / skip / penalty}
- Unicode の合字，異体字シーケンスなど

← upTEX に実装計画あり．npTEX では？

「upTeX+dvipdfmx] 異体字セレクタ、Unicode 合成文字」，
t-tk，tex-jp-build/#46, 2018-01-28.

目次

- はじめに
- 過去を見るか．未来を見るか
- Unicode の直接入力
- 開発者への説明
- XeTeX ベースにする場合の懸案事項

typeset by npTeX 3.141592653-2.6-0.999995-0.0000
TeX ε-TeX XeTeX npTeX