ε-pTEX　取扱説明書

北川 弘典（H7K）*

2009 年 3 月 9 日，build 090309

1　はじめに

ε-pTEX は，東京大学理学部数学科 3 年生対象の 2007 年度の授業「計算数学 II」*1 において北川が作成したプログラムであり，pTEX 3.1.10 を基盤として，ε-TEX 2.2 相当の機能と，10 進 21 桁の浮動小数点演算とを追加したものである．

製作の動機や作業過程などについては，詳しくは [1] を参照して欲しいけれども，大雑把に言うと，動機は以下のように要約できる．

- pTEX は，TEX が持っている「レジスタ 1 種類につき 256 個まで」という制限をひきずっており，現状でも非常に多数のパッケージを読み込ませたりすると制限にぶち当たってしまう．
- 一方，ε-TEX 拡張ではこれが「レジスタ 1 種類につき 32768 個まで」と緩和されており，欧文で標準となっている pdfTEX やその後継の LuaTEX でも ε-TEX の機能が取り込まれている．
- そうすると，pTEX だけが制限をレジスタ制限を引きずっているのは世界から取り残されることになるのではないか．

インストールについては，ε-pTEX のアーカイブ内にある `INSTALL.txt` を参照してください．

- ptetex3-20080616，及びそれをベースとした upTEX-0.26
- ptexlive-20080715（**自由選択**で upTEX-0.26 もまとめてコンパイル可能）
- TEX Live 2008（upTEX-0.26 も**自動的**にまとめてコンパイルされる）

の環境でコンパイルが通るようにしています*2．upTEX がある場合は，ε-pTEX と upTEX を無理やりマージした「ε-upTEX」といえそうなものもコンパイルされます．

* `http://sourceforge.jp/projects/eptex/wiki/`, e-mail: `h_kitagawa2001@yahoo.co.jp`

*1 `http://ks.ms.u-tokyo.ac.jp/`

*2 たぶん upTEX のバージョンが上がってもそれほど変更しないでコンパイルできると思います．ptexlive, ptetex3 については分かりません．

2　ε-TEX 拡張について

前に述べたように，ε-TEX は TEX の拡張の一つである．ε-TEX のマニュアル [4] には，開発目的が以下のように述べられている．

> The $\mathcal{NTS}$ project intends to develop an 'New Typesetting System' ($\mathcal{NTS}$) that will eventually replace today's TEX3. The $\mathcal{NTS}$ program will include many features missing in TEX, but there will also exist a mode of operation that is 100% compatible with TEX3. It will, necessarily, require quite some time to develop $\mathcal{NTS}$ to maturity and make it widely available.
>
> Meanwhile ε-TEX intends to fill the gap between TEX3 and the future $\mathcal{NTS}$. It consists of a series of features extending the capabilities of TEX3.

$\mathcal{NTS}$ がどうなったのか僕は知らない．しかし，少なくとも ε-TEX 拡張自体は実用的な物であり，そのせいか ℵ (Aleph), pdfTEX, XETEX などの他の拡張にもマージされており，かなりの人が ε-TEX 拡張を使うことができるようになっている．

ε-TEX 拡張で追加される機能について，詳しくは [4] を参照して欲しい．しかしそうやって丸投げするのはよろしくなさそうな気もするので，ひとまず [1] 中の 4.2 節「ε-TEX の機能」を引用することにする（一部改変）：

> ε-TEX には Compatibility mode と Extended mode の 2 つが存在し，前者では ε-TEX 特有の拡張は無効になるのでつまらない．後者がおもしろい．
>
> 拡張機能を使うにはファイル名を渡すときに `*` をつけるかコマンドラインオプションとして `-etex` スイッチをつければいいが，ε-TEX 拡張に関わる追加マクロは当然ながらそれだけでは駄目である．「plain マクロ for ε-TEX」（`etex.fmt` というのが一番マシかな）では自動的に追加マクロである `etex.src` が呼ばれる．LATEX 下ではちょうど `etex.src` に対応した `etex` パッケージを読み込む必要がある．

■レジスタの増加

最初に述べたように，TEX では 6 種類のレジスタが各 256 個ずつ利用できる．それぞれのレジスタには`\dimen75` などのように 0 ～ 255 の番号で指定できる他，予め別名の定義をしておけばそれによって指定することもできる．これらのいくつかは特殊な用途に用いられる（例えば`\count0` はページ番号などのように）ことになっているので，さらに user が使えるレジスタは減少する．

ε-TEX では，追加のレジスタとして番号で言うと 256 ～ 32767 が使用できるようになった．上の pdf によると最初の 0 ～ 255 と違って若干の制限はあるようだが，それは些細な話である．追加された（各種類あたり）$32768 - 256 = 32512$ 個のレジスタは，メモリの効率を重視するため sparse register として，つまり，必要な時に始めてツリー構造の中で確保されるようになっている．

■式が使用可能に

TEX における数量の計算は充実しているとは言い難い．例えば，

$$\texttt{\textbackslash dimen123} \leftarrow (\texttt{\textbackslash dimen42} + \texttt{\textbackslash @tempdima})/2$$

という計算を元々の TEX で書こうとすると，

```
\dimen123=\dimen42\advance\dimen123by\@tempdima\dimen123=0.5\@tempdima
```

のように書かないといけない．代入，加算代入，乗算代入，除算代入ぐらいしか演算が用意されていない状態になっている（上のコードのように，d_2 += $0.8d_1$ というような定数倍を冠することは平気）．

ε-TEX では，そのレジスタの演算に，他のプログラミング言語で使われているような数式の表現が使えるようになった．上の PDF では実例として

```
\ifdim \dimexpr (2pt-5pt)*\numexpr 3-3*13/5\relax + 34pt/2<\wd20
```

が書かれている．これは，

$$32\,\mathrm{pt} = (2\,\mathrm{pt} - 5\,\mathrm{pt})\,(3 - \mathrm{div}(3 \cdot 13, 5)) + \frac{34\,\mathrm{pt}}{2} < \texttt{\textbackslash box20}\ \text{の幅}$$

が真か偽かを判定していることになる．

■\middle primitive

TEX に\left, \right という primitive があり，それを使えば括弧の大きさが自動調整されるのはよく知られている．ε-TEX では，さらに\middle primitive が追加された．

具体例を述べる．

$$\left\{ n + \frac{1}{2} \,\middle|\, n \in \omega \right\} \left\{ n + \frac{1}{2} \,\middle|\, n \in \omega \right\}$$

これは以下の source で出力したものである：

```
\def\set#1#2{\setbox0=\hbox{$\displaystyle #1,#2$}%
\left\{\, \vphantom{\copy0}#1 \,\right|\!\left.\, %
    \vphantom{\copy0}#2 \,\right\}}
\def\eset#1#2{\left\{\, #1 \,\middle|\, #2 \,\right\}}
\[ \set{n+\frac12}{n\in \omega} \eset{n+\frac12}{n\in \omega} \]
```

両方とも集合の表記を行うコマンドである．TEX 流の\set では 2 つの\left, \right の組で実現させなければならず，そのために | の左側と右側に入る式の最大寸法を測定するという面倒な方法を使っている．その上，この定義では\textstyle 以下の数式（文章中の数式とか）ではそのまま使えず，それにも対応させようとすると面倒になる．一方，ε-TEX 流の\eset では，何も考えずに\left, \middle, \right だけで実現できる．

■ TEX--XETEX (TeX--XeT)

left-to-right と right-to-left を混植できるという機能であるらしい．ヘブライ語あたりの組版に使えるらしいが，よく知らない．ここでの RtoL は LtoR に組んだものを逆順にしているだけのような気がする．

とりあえず一目につきそうな拡張機能といったらこれぐらいだろうか．他にも tracing 機能や条件判断文の強化などあるが，そこら辺はパッとしないのでここで紹介するのは省略することにしよう．

ε-pTEX ではここに述べた代表的な機能を含め，ほとんどすべての機能を実装しているつもりである．ただ，TEX--XETEX を和文で使うと約物の位置がずれたり空白がおかしかったりするけれども，そこの修正は大変に思えるので放置してある．

`\lastnodetype` と `\currentiflevel` の挙動については，[1] にもあるが，pTEX や浮動小数点演算実装のため，（それらで追加された部分に関しては）ε-TEX 本来の挙動では起こり得ない値をとることがある．

まず `\lastnodetype` については，以下のような値をとる．

−1:	none (empty list)	6:	adjust node	13:	penalty node
0:	char node	7:	ligature node	14:	unset node
1:	hlist node	8:	disc node	15:	math mode nodes
2:	vlist node	9:	whatsit node	**16:**	direction node
3:	rule node	10:	math node	**17:**	displacement node
4:	ins node	11:	glue node		
5:	mark node	12:	kern node		

次に，`\currentiflevel` における条件判断文とそれを表す数字との対応は，以下のようになっている．

1:	`\if`	8:	`\ifmmode`	15:	`\iftrue`	**22:**	`\ifydir`
2:	`\ifcat`	9:	`\ifinner`	16:	`\iffalse`	**23:**	`\ifmdir`
3:	`\ifnum`	10:	`\ifvoid`	17:	`\ifcase`	**24:**	`\iftbox`
4:	`\ifdim`	11:	`\ifhbox`	18:	`\ifdefined`	**25:**	`\ifybox`
5:	`\ifodd`	12:	`\ifvbox`	19:	`\ifcsname`	**26:**	`\iffp`
6:	`\ifvmode`	13:	`\ifx`	20:	`\iffontchar`		
7:	`\ifhmode`	14:	`\ifeof`	**21:**	`\iftdir`		

3 浮動小数点演算

この機能は時間が余ったためにネタで入れたようなものである．非常に原始的なコマンドしか提供しておらず，僕自身ほとんど使わないためにもはや改善しようという意欲もあまりない（バグが見つかれば，もちろん直す気ですが）．

ここでいう**浮動小数点数**とは，BASIC や FORTRAN で見られるような，`-235.673578432E-534` のように，符号と 10 進小数からなる仮数部に，必要に応じて `E` or `e` で始まる指数部が続いたものである．符号は複数あってもよく，そのときは全部掛け算したものが最終的な符号となる．小数点は日米などで使われるピリオドも，欧州大陸で使われるコンマも許容される．

浮動小数点数に纏わるエラーは，オーバーフロー (`Floating arithmetic overflow`) と例外 (`Floating arithmetic exception`) の 2 種類のみである．エラーメッセージを出して一旦停止するが，無限大値や NaN 値をそれぞれ代入して続けることができる．

3.1 初期化

浮動小数点演算を行うときには，ε-TEX でいうところの extended mode で処理する必要があり，さらに計算に使用する一時領域や定数 π などを確保する必要がある．この一時領域確保はは自動では行われない．

- `\fpinit` 一時領域その他を確保する．
- `\fpdest` 一時領域その他を解放する．

初期化を忘れて演算を行おうとしても，特に TEX 側でエラーチェックは行わない．そのため，セグメンテーション違反のようなエラーを引き起こす危険性がある．解放忘れや二重確保は単に（TEX 内の）メモリを無駄にするだけで，それ以外に問題はない．

3.2 代入，型変換，入出力

- `\real`⟨*real*⟩ 浮動小数点数を表現する glue（以下，⟨*f-glue*⟩ と称する）を返す．
- `\fpfrac`⟨*f-glue*⟩ 引数の仮数部を返す．
- `\fpexpr`⟨*f-glue*⟩ 引数の指数部を返す．
- `\fptoint`⟨*f-glue*⟩ 引数を整数に変換したものを返す．範囲内に収まらないときは，`Number too big` エラーを返す．なお，引数が整数でないときは，0 に近い方に丸められる．
- `\fptodim`⟨*f-glue*⟩ 引数を dimension に，1.0 がちょうど 1 pt になるように変換したものを返す．範囲内に収まらないときは，`Dimension too large` エラーを返す．なお，引数が 1/65536 (1 sp に対応) でないときは，同じように 0 に近い方に丸められる．

3.3 四則演算等

- `\fpadd`⟨*f-reg*⟩⟨*real*⟩ ⟨*f-glue*⟩ が格納された skip レジスタ（以下，⟨*f-reg*⟩ と称する）と浮動小数点数を引数にとり，2 つの浮動小数点数の和を計算して，⟨*f-glue*⟩ に上書きする．
- `\fpsub`⟨*f-reg*⟩⟨*real*⟩ 同様に差を計算して，⟨*f-glue*⟩ に上書きする．
- `\fpmul`⟨*f-reg*⟩⟨*real*⟩ 同様に積を計算して，⟨*f-glue*⟩ に上書きする．
- `\fpdiv`⟨*f-reg*⟩⟨*real*⟩ 同様に商を計算して，⟨*f-glue*⟩ に上書きする．
- `\fppow`⟨*f-reg*⟩⟨*real*⟩ 同様にべき乗を計算して，⟨*f-glue*⟩ に上書きする．このコマンドは $x^y = \exp(y \log x)$ で計算するため，第 1 引数が負では当然ながらエラーが生じる．

- \fppowi⟨*f-reg*⟩⟨*num*⟩　\fppow と同様に累乗を計算するが，第 2 引数，つまり指数部分は整数に限られる．その代わり，第 1 引数は負でもかまわない．

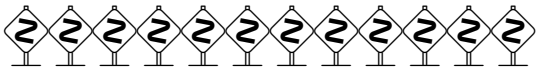
3.4 単項演算

以下は単項演算で，⟨*f-reg*⟩ を 1 つとり，演算をし，結果をその ⟨*f-reg*⟩ に上書きする．よって，以下はコマンドの引数も省略し，演算内容しか書かない．引数の内容は便宜的に x で表す．

- \fpneg　-1 倍を計算する．
- \fpsqr　平方根 $\sqrt{x}$ を計算する．
- \fpexp　指数関数 $\exp x$ を計算する．
- \fplog　対数関数 $\log x$ を計算する．底は $e = 2.71828\cdots$ である．
- \fpabs　絶対値を計算する．
- \fpceil　天井関数 $\lceil x \rceil$，つまり x を越えない最小の整数を計算する．
- \fpfloor　床関数 $\lfloor x \rfloor$，つまり x 以下の最大の整数を計算する．
- \fpsin, ..., \fptan　それぞれ三角関数 $\sin x$, $\cos x$, $\tan x$ を計算する．
- \fpsinh, ..., \fptanh　それぞれ双曲線関数 $\sinh x$, $\cosh x$, $\tanh x$ を計算する．
- \fpasin, ..., \fpatan　それぞれ逆三角関数 $\arcsin x$, $\arccos x$, $\arctan x$ を計算する．
- \fpasinh, ..., \fpatanh　それぞれ逆双曲線関数 $\operatorname{arcsinh} x$, $\operatorname{arccosh} x$, $\operatorname{arctanh} x$ を計算する．

逆三角関数では主値をとっている．すなわち，計算結果は $\arcsin x$, $\arctan x \in [-\pi/2, \pi/2]$, $\arccos x \in [0, \pi]$ となる．また, 逆双曲線関数の計算結果は $\operatorname{arcsinh} x$, $\operatorname{arctanh} x \in \mathbf{R}$, $\operatorname{arccosh} x \in [0, \infty]$ の範囲に収まる．

これらの初等関数の計算は，関係式を用いて引数 x を適度な大きさに変換し．そして Taylor 展開を用いて行っている．



4 「FAM256」パッチについて

FAM256 パッチは，掲示板 TEX Q & A の山本氏の書き込み [2] に刺激されて作ったものであり，以下に説明する Ω の一部機能を使えるようにするパッチである（この名称は便宜的なもの）.

4.1 パッチの使い方

FAM256 パッチは標準では無効にしてある.

有効にするには，INSTALL.txt にあるように，コンパイルに使用するスクリプト[*3]にある

```
FAM256=0
```

という行を

```
FAM256=1
```

に変えて，通常のコンパイルの作業を行えば良い.

また，FAM256 パッチが有効なバイナリにおいては，

```
h7k ~ $ euptex
This is e-upTeXk, Version 3.1415926-p3.1.10-u0.26-090309 FAM256-PATCHED
(utf8.uptex) (Web2C 7.5.7)
 %&-line parsing enabled.
**
```

のように，起動時のバナーの 1 行めに「FAM256-PATCHED」が追加される．-version オプションをつけて起動したときも，

```
BSD h7k ~ $ eptex -version
e-pTeX 3.141592-p3.1.10-090309_FAM256-PATCHED (utf8.euc) (Web2C 7.5.6)
kpathsea version 3.5.6
ptexenc version 0.999-u00
（後略）
```

のように，「FAM256-PATCHED」が追加される.

ε-TEX 拡張とは違い，FAM256 パッチを有効にしてバイナリを作った場合，追加機能は extended mode でなくても有効になっている．ただし，後に述べるように，本パッチではレジスタ

[*3] option（TEX Live 2008 or ptexlive 使用時）か script/build（ptetex3 + upTEX 使用時）.

が 65536 個まで使えるようにしているが，それについてだけは extended mode の時に限り有効になる．

4.2 機能解説

本ドキュメントの最後のページ[*4]にちょっとしたサンプルを載せてある．

■数式フォント制限の緩和

Ω の大きな特徴としては，TEX 内部のデータ構造を倍の領域を用いるように改変し[*5]，TEX に従来から存在していた「256 個制限」を 2^{16} 個にまで緩和したことが挙げられる．同様に，Ω では（[2] にもあるように）数式フォントを同時に 256 個まで用いることができ，各フォントも 65536 文字まで許されるようになっている．

FAM256 パッチでは，中途半端だが，数式フォント 1 つあたりの使用可能文字数は 256 個のままで，同時に数式フォントを 256 個まで使えるようにしている．基本的には Ω と同様の方法を用いているが，内部でのデータ構造に違いがある（数字はすべて bit 幅）：

	category	family	char	math code	delimiter code
TEX	3	4	8	$3+4+8=15$	$3+2\cdot(4+8)=27$
Ω	3	8	16	$3+8+16=27$	$(3+8+16,\ 8+16)=(27,24)$
FAM256	3	8	8	$3+8+8=21$	$(3+8+8,\ 8+8)=(19,16)$

TEX に既存のコマンド類は互換性維持のために同じ動作とする必要があるので，16 番から 255 番のフォントを利用する際には別のコマンドが必要となる．（実装自体に Ω の流儀を使っているから）FAM256 では，Ω のコマンド類を流用することにした．すなわち，FAM256 では，

- `\omathcode` ⟨*8-bit number*⟩ = ⟨*27-bit number*⟩
- `\omathcode` ⟨*8-bit number*⟩
- `\omathchar` ⟨*27-bit number*⟩
- `\omathaccent` ⟨*27-bit number*⟩
- `\omathchardef` ⟨*control-sequence*⟩ = ⟨*27-bit number*⟩
- `\odelcode` ⟨*8-bit number*⟩ = ⟨*27-bit number*⟩⟨*24-bit number*⟩
- `\odelimiter` ⟨*27-bit number*⟩⟨*24-bit number*⟩
- `\oradical` ⟨*27-bit number*⟩⟨*24-bit number*⟩

が追加されている[*6]．27 bit とか 24 bit の自然数の意味については，上の表の Ω の行を参照して欲しい．上に書いた FAM256 での実装から，character code の指定に使われる 16 bit の数値で，実際に使われるのは下位 8 bit であり，上位 8 bit は無視される．

[*4] ただし，ソースファイルで言えば `fam256d.tex`（本文）と `fam256p.tex`（preamble 部）に対応する．

[*5] 詳しい話は `texk/web2c/texmfmem.h` 中の共用体 `memoryword` の定義を参照してください．大雑把に言うと，1 つの「メモリ要素」に 2 つの 32 bit 整数を同時に格納できるようになっています．

[*6] Ω では ⟨*8-bit number*⟩ のところが ⟨*16-bit number*⟩ になっている．

なお，\odelcode⟨*8-bit number*⟩ として delimiter code を取得しようとしても，現時点のパッチでは，うまく動作しない[*7]．

■無限のレベル

TEX では，glue の伸縮量に 3 つの無限大のレベルが存在した：fil, fill, filll であり，l が多いほど無限大のオーダーが高い．Ω では，「inter-letter spacing のために」fi という，有限と fil の中間にあたる無限大のレベルが付け加えられ，\hfi, \vfi という 2 つの primitive も追加された．FAM256 パッチでは，この無限大レベル fi も採用することにした．

実装方法は，大まかには Ω で fi の実装を行っている change file omfi.ch の通りであるのだが，これに pTEX や ε-TEX に伴う少々の修正を行っている．

- コマンド\pagefistretch を新たに定義している．
- \gluestretchorder, \glueshrinkorder の動作を ε-TEX のそれと合わせた．具体的には，ある適当な glue \someglue の stretch 幅を ⟨*stretch*⟩ とおくとき，

$$\texttt{\textbackslash gluestretchorder\textbackslash someglue} = \begin{cases} 0 & \langle \textit{stretch} \rangle \text{ が高々 fi レベルの量} \\ 1 & \langle \textit{stretch} \rangle \text{ がちょうど fil レベルの無限量} \\ 2 & \langle \textit{stretch} \rangle \text{ がちょうど fill レベルの無限量} \\ 3 & \langle \textit{stretch} \rangle \text{ がちょうど filll レベルの無限量} \end{cases}$$

となっている．内部では fi レベルが 1，fil レベルが 2，……として処理している．

■レジスタについて

Ω では（前にも書いたが）データ構造の変更が行われ，それによってレジスタが各種類あたり 65536 個使えるようになっている．

一方，ε-TEX では，256 番以降のレジスタを専用の sparse tree に格納することにより，32767 番までのレジスタの使用を可能にしていた．この tree 構造を分析してみると，65536 個までレジスタを拡張するのはさほど難しくないことのように思われた．具体的には，tree の階層を 1 つ増やしてみた（だから，おそらく各種類あたり $16 \cdot 32768 = 524288$ 個まで使えるとは思うが，これはきりが悪い）．そこで，FAM256 パッチでは ε-TEX 流の方法を用いながらも，レジスタをさらに 65536 個まで増やしている．

参考文献

[1] 北川 弘典,「計算数学 II　作業記録」, 2008.
https://sourceforge.jp/projects/eptex/document/resume/ja/1/resume.pdf 他

[2] 山本 和義,「数式 fam の制限と luatex」, 掲示板「TEX Q & A」52744 番書き込み, 2009.2.12,
http://oku.edu.mie-u.ac.jp/~okumura/texfaq/qa/52744.html

[*7] 51 bit 自然数を返さないといけないですからねえ．やる気があれば検討してみます．

[3] 山本 和義,「Re: 数式 fam の制限と luatex」, 掲示板「TEX Q & A」52767 番書き込み, 2009.2.16, http://oku.edu.mie-u.ac.jp/~okumura/texfaq/qa/52767.html

[4] The NTS Team. *The ε-TEX manual*. http://www.tug.org/texlive/Contents/live/texmf-dist/doc/etex/base/etex_man.pdf

[5] J. Plaice, Y. Haralambous. *Draft documentation for the Ω system*, 1999. http://www.tug.org/texlive/Contents/live/texmf-dist/doc/omega/base/doc_1.8.tex

Test source for FAM256 patch

本ソースは山本和義氏による「数式 fam の制限と luatex」（qa:52744）中のコードをベースにしたものである.

■ **More than 16 math font families.**

A**BCDEFGHIJKL fam = 21**

AaAbAcAdAeAfAgAhAiAjAkAlAmAnAoAp fam37

AaAbAcAdAeAfAgAhAiAjAkAlAmAnAoAp fam53

AaAbAcAdAeAfAgAhAiAjAkAlAmAnAo**Ap fam69**

Aa fam70 Ab fam71 Ac fam72 Ad fam73 roman

■**\omathchar etc.**

\mathchar"7F25 : %, \omathchar"7420125 : %

meaning of \langle: macro:->\delimiter "426830A ,

meaning of \lx: macro:->\odelimiter "4450068"030001

\lx : **h**, \bigl\lx :), \Bigl\lx :)

\the\mathcode‘\f: 7166, \the\omathcode‘\f: 7010066（どちらも 16 進に変換した）

t ⟩⟩⟩))) e e

√, p $\sqrt{a}$, p■ $\sqrt{\int_V f\,d\mu}$, $\sqrt{\int_V f\,d\mu}$

\odelcode primitive による **delimiter code の取得**はうまく動かない：
\the\delcode‘\|: 618, \the\odelcode‘\|: −1（どちらも 10 進）

■ **Infinite level “fi”**

■(fi)■(fil)■(fill)■	(filll)			■
■(fi)■(fil)■	(fill)			■
■(fi)■	(fil)			■
■ (fi)	■	(fi)	(fi)	■

■ **65536 registers**

fuga! a 漢字仮名sdf TEX ほげほげTEX

589