upLaTeX 2ε について

Ken Nakano & Japanese TeX Development Community & TTK

作成日：2016/06/19

注意：

これは、アスキーのオリジナル版から fork したコミュニティ版 pLaTeX 2ε の付属文書を upLaTeX 2ε 用に書き換えたものです。

アスキー pTeX には、高品質の日本語組版ソフトウェアとしてデファクトスタンダードの地位にあるといえます。しかし、(1) 直接使える文字集合が原則的に JIS X 0208（JIS 第 1,2 水準）の範囲に限定されていること、(2) 8bit の非英語欧文との親和性が高いとは言えないこと、(3) pTeX の利用が日本語に限られ、中国語・韓国語との混植への利用が進んでいないこと、といった弱点がありました。

これらの弱点を克服するため、pTeX の内部コードを Unicode 化した拡張版が upTeX です。また、upTeX 上で用いる Unicode 版 pLaTeX が upLaTeX です[1]。現在の upLaTeX は、日本語 TeX 開発コミュニティが配布しているコミュニティ版 pLaTeX[2] をベースにしています。開発中の版は pLaTeX と同様に、GitHub のリポジトリ[3]で管理しています。

より詳細な upTeX や upLaTeX の情報は、それぞれ `README_uplatex.txt` や `README_uptex.txt` などのテキストファイルを参照してください。

[1] `http://www.t-lab.opal.ne.jp/tex/uptex.html`
[2] `https://github.com/texjporg/platex`
[3] `https://github.com/texjporg/uplatex`

1 概要

この文書は、upLaTeX 2$_\varepsilon$ の概要を示していますが、使い方のガイドではありません。元となっている pLaTeX 2$_\varepsilon$ や LaTeX 2$_\varepsilon$ については、それぞれ pLaTeX 2$_\varepsilon$ と LaTeX 2$_\varepsilon$ の付属文書を参照してください。

この文書の構成は次のようになっています。

第 1 節 この節です。この文書についての概要を述べています。

第 2 節 upLaTeX 2$_\varepsilon$ で拡張した機能についての概要です。付属のクラスファイルやパッケージファイルについても簡単に説明しています。

付録 A この文書ソースの DOCSTRIP のためのオプションについて述べています。

付録 B upLaTeX 2$_\varepsilon$ の dtx ファイルをまとめて一つの DVI ファイルにするための文書ファイルの説明をしています。

付録 C 付録 B で説明をした文書ファイルを処理する sh スクリプト（手順）、DOCSTRIP 文書ファイル内の入れ子の対応を調べる perl スクリプトなどについて説明しています。

2 upLaTeX 2$_\varepsilon$ の機能について

upLaTeX 2$_\varepsilon$ の機能は、いくつものファイルに分割されて実装されています。これらのファイルはつぎの 3 種類に分類することができます。

- フォーマットファイル
- クラスファイル
- パッケージファイル

フォーマットファイルには、基本的な機能が定義されており、upLaTeX 2$_\varepsilon$ の核となるファイルです。このファイルに定義されているマクロは、実行時の速度を高めるために、あらかじめ TeX の内部形式の形で保存されています。

クラスファイルは文書のレイアウトを設定するファイル、パッケージファイルはマクロの拡張を定義するファイルです。前者は`\documentclass` コマンドを用いて読み込み、後者は`\usepackage` コマンドを用いて読み込みます。

古い *pLaTeX 2.09* ユーザへの注意：

upLaTeX は新しいマクロパッケージですので、2.09 互換モードはサポートしていません。LaTeX 2$_\varepsilon$ の仕様に従ってドキュメントを作成してください。

2.1 フォーマットファイル

フォーマットファイルには、基本的な機能が定義されていますが、これらは TEX の内部形式に変換された形式となっています。フォーマットファイルを作成するには、ソースファイル “uplatex.ltx” を`iniuptex` プログラムで処理します。ただし、TEX Live や W32TEX ではこの処理を簡単にする`fmtutil` あるいは`fmtutil-sys` というプログラムが用意されています。以下を実行すれば、フォーマットファイル `uplatex.fmt` が作成されます。

```
fmtutil --byfmt uplatex
```

次のリストが、“uplatex.ltx” のの内容です。ただし、このバージョンでは、LATEX から upLATEX 2ε への拡張を `uplcore.ltx` をロードすることで行ない、`latex.ltx` には直接、手を加えないようにしています。したがって `uplatex.ltx` はとても短いものとなっています。`latex.ltx` には LATEX のコマンドが、`uplcore.ltx` には upLATEX 2ε で拡張したコマンドが定義されています。

```
⟨*plcore⟩
\let\orgdump\dump
\let\dump\relax
\input latex.ltx
\edef\platexBANNER{\the\everyjob}% save LaTeX banner
\typeout{**************************^^J%
         *^^J%
         * making upLaTeX format^^J%
         *^^J%
         **************************}
\makeatletter
\input uplcore.ltx
\the\everyjob
\let\dump\orgdump
\let\orgdump\@undefined
\makeatother
\dump
⟨/plcore⟩\endinput
⟨/plcore⟩
```

実際に upLATEX 2ε への拡張を行なっている `uplcore.ltx` は、DOCSTRIP プログラムによって、次のファイルの断片が連結されたものです。

- `uplvers.dtx` は、upLATEX 2ε のフォーマットバージョンを定義しています。
- `uplfonts.dtx` は、NFSS2 を拡張しています。
- このほか、pLATEX 2ε に含まれる `plcore.dtx` をそのまま利用しています。これは、上記以外のコマンドでフォーマットファイルに格納されるコマンドを定義しています。

プリロードフォントや組版パラメータなどの設定は、`upldefs.ltx` をロードすることで行なっています。このファイルに記述されている設定を変更すれば、upLaTeX 2$_\varepsilon$ をカスタマイズすることができます。カスタマイズする場合は、このファイルを直接、修正するのではなく、`upldefs.cfg` という名前でコピーをして、そのファイルを編集します。`upldefs.cfg` は `upldefs.ltx` の代わりに読み込まれます。

2.1.1 バージョン

upLaTeX 2$_\varepsilon$ のバージョンやフォーマットファイル名は、`uplvers.dtx` で定義しています。これは、pLaTeX 2$_\varepsilon$ のバージョンやフォーマットファイル名が `plvers.dtx` で定義されているのと同じです。

2.1.2 NFSS2 コマンド

LaTeX では、フォント選択機構として NFSS2 を用いています。pLaTeX 2$_\varepsilon$ では、オリジナルの NFSS2 と同様のインターフェイスで、和文フォントを選択できるように、`plfonts.dtx` で NFSS2 を拡張していますので、upLaTeX 2$_\varepsilon$ も `uplfonts.dtx` で同じ方式を採用しています。

`uplfonts.dtx` ファイルでは、NFSS2 コマンドの定義のほか、プリロードフォントの設定、和文エンコードの定義、組版パラメータなどの設定、フォント定義ファイルなどの記述も含まれています。

2.2 クラスファイルとパッケージファイル

upLaTeX 2$_\varepsilon$ が提供をする、クラスファイルやパッケージファイルのいくつかは、pLaTeX 2$_\varepsilon$ に含まれるファイルを修正しています。

upLaTeX 2$_\varepsilon$ に付属のクラスファイルは、次のとおりです。

- ujbook.cls, ujarticle.cls, ujreport.cls

  横組用の標準クラスファイル。`ujclasses.dtx` から作成される。

- utbook.cls, utarticle.cls, utreport.cls

  縦組用の標準クラスファイル。`ujclasses.dtx` から作成される。

また、upLaTeX 2$_\varepsilon$ に付属のパッケージファイルは、次のとおりです。

- uptrace.sty

  LaTeX でフォント選択コマンドのトレースに使う `tracefnt.sty` が再定義してしまう NFSS2 コマンドを、upLaTeX 2$_\varepsilon$ 用に再々定義するためのパッケージ。`uplfonts.dtx` から作成される。

3　旧バージョンとの互換性

ここでは、このバージョンと以前のバージョンとの互換性や拡張部分について説明をしています。

3.1　pLaTeX との互換性

upLaTeX 2$_\varepsilon$ は、pLaTeX の上位互換という形を取っていますので、クラスファイルやいくつかのコマンドを置き換えるだけで、たいていの pLaTeX 文書を簡単に upLaTeX 文書に変更することができます。ただし、upLaTeX では pLaTeX で問題が指摘されていたフォントメトリックの不都合などいくつかのパラメータを変更していますので、レイアウトが変化することがあります。

また、upLaTeX は新しいマクロパッケージですので、2.09 互換モードをサポートしていません。LaTeX 2$_\varepsilon$ の仕様に従ってドキュメントを作成してください。

pLaTeX 向けあるいは LaTeX 向けに作られた多くのクラスファイルやパッケージファイルはそのまま使えると思います。ただし、pLaTeX 標準の漢字エンコーディング (JY1, JT1) を前提としたものが upLaTeX で採用した漢字エンコーディング (JY2, JT2) と合致しないといったエラーが発生することもあります。用いようとしているクラスファイルやパッケージファイルがうまく動くかどうかを、完全に確かめる方法は残念ながらありません。一番簡単なのは、動かしてみることです。不幸にもうまく動かない場合は、ログファイルや付属の文書ファイルを参考に原因を調べてください。

3.2　latexrelease パッケージへの対応

LaTeX <2015/01/01>で導入された latexrelease パッケージをもとに、新しい pLaTeX では platexrelease パッケージが用意されました。本来は upLaTeX でも同様のパッケージを用意するのがよいのですが、現在は pLaTeX から upLaTeX への変更点が含まれていませんので、幸い platexrelease パッケージをそのまま用いることができます。このため、upLaTeX で独自のパッケージを用意することはしていません。platexrelease パッケージを用いると、過去の upLaTeX をエミュレートしたり、フォーマットを作り直すことなく新しい upLaTeX を試したりすることができます。詳細は platexrelease のドキュメントを参照してください。

A　DOCSTRIP プログラムのためのオプション

この文書のソース (uplatex.dtx) を DOCSTRIP プログラムによって処理することによって、いくつかの異なるファイルを生成することができます。DOCSTRIP プログ

ラムの詳細は、docstrip.dtx を参照してください。
　この文書の DOCSTRIP プログラムのためのオプションは、次のとおりです。

オプション	意味
plcore	フォーマットファイルを作るためのファイルを生成
pldoc	upLaTeX 2_ε のソースファイルをまとめて組版するための文書ファイルを生成
shprog	上記のファイルを作成するための sh スクリプトを生成
plprog	入れ子構造を調べる簡単な perl スクリプトを生成
Xins	上記の sh スクリプトや perl スクリプトを取り出すための DOCSTRIP バッチファイルを生成

A.1　ファイルの取り出し方

たとえば、この文書の “plcore” の部分を “uplatex.ltx” というファイルにするときの手順はつぎのようになります。

1. uplatex docstrip
2. 入力ファイルの拡張子（dtx）を入力する。
3. 出力ファイルの拡張子（ltx）を入力する。
4. DOCSTRIP オプション（plcore）を入力する。
5. 入力ファイル名（uplatex）を入力する。
6. uplatex.ltx が存在する場合は、確認を求めてくるので、“y” を入力する。
7. 別の処理を行なうかを問われるので、“n” を入力する。

これで、uplatex.ltx が作られます。
　あるいは、次のような内容のファイル fmt.ins を作成し、uplatex fmt.ins することでも uplatex.ltx を作ることができます。

```
\def\batchfile{fmt.ins}
\input docstrip.tex
\generateFile{uplatex.ltx}{t}{\from{uplatex.dtx}{plcore}}
```

B　文書ファイル

ここでは、このパッケージに含まれている dtx ファイルをまとめて組版をするための文書ファイルについて説明をしています。個別に処理した場合と異なり、変更履歴や索引も付きます。全体で、およそ 120 ページ程度になります。

filecontents 環境は、引数に指定されたファイルが存在するときは何もしませんが、存在しないときは、環境内の内容でファイルを作成します。upldoc.dic ファイルは、mendex プログラムで索引を処理するときに \ 西暦, \ 和暦に対する「読み」を付けるために必要です。

```
⟨*pldoc⟩
\begin{filecontents}{upldoc.dic}
西暦　　せいれき
和暦　　われき
\end{filecontents}
```

文書クラスには、jltxdoc クラスを用います。

```
\documentclass{jltxdoc}
%\usepackage{plext} %% comment out for upLaTeX
\listfiles

```

いくつかの TEX プリミティブとコマンドを索引に出力しないようにします。

```
\DoNotIndex{\def,\long,\edef,\xdef,\gdef,\let,\global}
\DoNotIndex{\if,\ifnum,\ifdim,\ifcat,\ifmmode,\ifvmode,\ifhmode,%
            \iftrue,\iffalse,\ifvoid,\ifx,\ifeof,\ifcase,\else,\or,\fi}
\DoNotIndex{\box,\copy,\setbox,\unvbox,\unhbox,\hbox,%
            \vbox,\vtop,\vcenter}
\DoNotIndex{\@empty,\immediate,\write}
\DoNotIndex{\egroup,\bgroup,\expandafter,\begingroup,\endgroup}
\DoNotIndex{\divide,\advance,\multiply,\count,\dimen}
\DoNotIndex{\relax,\space,\string}
\DoNotIndex{\csname,\endcsname,\@spaces,\openin,\openout,%
            \closein,\closeout}
\DoNotIndex{\catcode,\endinput}
\DoNotIndex{\jobname,\message,\read,\the,\m@ne,\noexpand}
\DoNotIndex{\hsize,\vsize,\hskip,\vskip,\kern,\hfil,\hfill,\hss,\vss,\unskip}
\DoNotIndex{\m@ne,\z@,\z@skip,\@ne,\tw@,\p@,\@minus,\@plus}
\DoNotIndex{\dp,\wd,\ht,\setlength,\addtolength}
\DoNotIndex{\newcommand, \renewcommand}

```

索引と変更履歴の見出しに \part を用いるように設定をします。

```
\IndexPrologue{\part*{索 引}%
                  \markboth{索 引}{索 引}%
                  \addcontentsline{toc}{part}{索 引}%
イタリック体の数字は、その項目が説明されているページを示しています。
下線の引かれた数字は、定義されているページを示しています。
```

```
その他の数字は、その項目が使われているページを示しています。}
%
\GlossaryPrologue{\part*{変更履歴}%
                  \markboth{変更履歴}{変更履歴}%
                  \addcontentsline{toc}{part}{変更履歴}}

```

標準の\changes コマンドを、複数ファイルの文書に合うように修正しています。

```
\makeatletter
\def\changes@#1#2#3{%
  \let\protect\@unexpandable@protect
  \edef\@tempa{\noexpand\glossary{#2\space\currentfile\space#1\levelchar
               \ifx\saved@macroname\@empty
                  \space\actualchar\generalname
               \else
                  \expandafter\@gobble
                  \saved@macroname\actualchar
                  \string\verb\quotechar*%
                  \verbatimchar\saved@macroname
                  \verbatimchar
               \fi
               :\levelchar #3}}%
  \@tempa\endgroup\@esphack}
\makeatother
\RecordChanges
\CodelineIndex
\EnableCrossrefs
\setcounter{IndexColumns}{2}
\settowidth\MacroIndent{\ttfamily\scriptsize 000\ }
```

ここからが本文ページとなります。

```
\begin{document}
 \title{The up\LaTeXe\ Sources}
 \author{Ken Nakano \& Japanese \TeX\ Development Community \& TTK}

% This command will be used to input the patch file
% if that file exists.
\newcommand{\includeltpatch}{%
  \def\currentfile{uplpatch.ltx}
  \part{uplpatch}
  {\let\ttfamily\relax
    \xdef\filekey{\filekey, \thepart={\ttfamily\currentfile}}}%
  Things we did wrong\ldots
  \IndexInput{uplpatch.ltx}}

% Get the date and patch level from uplvers.dtx
\makeatletter
\let\patchdate=\@empty
\begingroup
   \def\ProvidesFile#1\pfmtversion#2#3\ppatch@level#4{%
```

```
      \date{#2}\xdef\patchdate{#4}\endinput}
   \input{uplvers.dtx}
\global\let\X@date=\@date

% Add the patch version if available.
   \long\def\Xdef#1#2#3\def#4#5{%
    \xdef\X@date{#2}%
    \xdef\patchdate{#5}%
    \endinput}%
   \InputIfFileExists{uplpatch.ltx}
    {\let\def\Xdef}{\global\let\includeltpatch\relax}
\endgroup

\ifx\@date\X@date
  \def\Xpatch{0}
  \ifx\patchdate\Xpatch\else
    \edef\@date{\@date\space Patch level\space\patchdate}
  \fi
\else
   \@warning{uplpatch.ltx does not match uplvers.dtx!}
   \let\includeltpatch\relax
\fi
\makeatother

\pagenumbering{roman}
\maketitle
\renewcommand\maketitle{}
\tableofcontents
\clearpage
\pagenumbering{arabic}

\DocInclude{uplvers}   % upLaTeX version

\DocInclude{uplfonts}  % NFSS2 commands

%\DocInclude{plcore}   % kernel commands (comment out for upLaTeX)

%\DocInclude{plext}    % external commands (comment out for upLaTeX)

%\DocInclude{pl209}    % 2.09 compatibility mode commands (comment out for upLaTeX)

\DocInclude{ukinsoku}  % kinsoku parameter

\DocInclude{ujclasses} % Standard class

%\DocInclude{jltxdoc}  % dtx documents class (comment out for upLaTeX)

%\includeltpatch       % patch file (comment out May 8, 2016)

```

ltxdoc.cfg に\AtEndOfClass{\OnlyDescription}が指定されている場合は、ここで終了します。

```
\StopEventually{\end{document}}
```

変更履歴と索引を組版します。変更履歴ファイルと索引の作り方の詳細については、おまけ C.1 を参照してください。

```
\clearpage
\pagestyle{headings}
% Make TeX shut up.
\hbadness=10000
\newcount\hbadness
\hfuzz=\maxdimen
%
\PrintChanges
\clearpage
%
\begingroup
  \def\endash{--}
  \catcode`\-\active
  \def-{\futurelet\temp\indexdash}
  \def\indexdash{\ifx\temp-\endash\fi}

  \PrintIndex
\endgroup
```

ltxdoc.cfg に２度目の\PrintIndex が指定されているかもしれません。そこで、最後に、変更履歴や索引が２度組版されないように\PrintChanges および\PrintIndex コマンドを何も実行しないようにします。

```
\let\PrintChanges\relax
\let\PrintIndex\relax
\end{document}
```

⟨/pldoc⟩

C　おまけプログラム

C.1　シェルスクリプト mkpldoc.sh

upLaTeX 2ε のマクロ定義ファイルをまとめて組版するときに便利なシェルスクリプトです。このシェルスクリプト[4]の使用方法は次のとおりです。

```
sh mkpldoc.sh
```

[4] このシェルスクリプトは UNIX 用です。しかし rm コマンドを delete コマンドにするなどすれば、簡単に DOS などのバッチファイルに修正することができます。

C.1.1 mkpldoc.sh の内容

まず、以前に upldoc.tex を処理したときに作成された、目次ファイルや索引ファイルなどを削除します。

```
171 ⟨*shprog⟩
172 for f in upldoc.toc upldoc.idx upldoc.glo ; do
173 if [ -e $f ]; then rm $f; fi
174 done
```

そして、ltxdoc.cfg を空にします。このファイルは、jltxdoc.cls の定義を変更するものですが、ここでは、変更されたくありません。

```
175 echo "" > ltxdoc.cfg
```

そして、upldoc.tex を処理します。

```
176 uplatex upldoc.tex
```

索引と変更履歴を作成します。このスクリプトでは、変更履歴や索引を生成するのに mendex プログラムを用いています。mendex は makeindex の上位互換のファイル整形コマンドで、索引語の読みを自動的に付けるなどの機能があります。

upLaTeX の場合は mendex コマンドを UTF-8 モードで実行する必要がありますので、-U というオプションを付けます[5]。makeindex コマンドには、このオプションがありません。

-s オプションは、索引ファイルを整形するためのスタイルオプションです。索引用の gind.ist と変更履歴用の gglo.ist は、LaTeX のディストリビューションに付属しています。

-o は、出力するファイル名を指定するオプションです。

-f は、項目に “読み” がなくてもエラーとしないオプションです。makeindex コマンドには、このオプションがありません。

```
177 mendex -U -s gind.ist -d upldoc.dic -o upldoc.ind upldoc.idx
178 mendex -U -f -s gglo.ist -o upldoc.gls upldoc.glo
```

ltxdoc.cfg の内容を\includeonly{}にし、upldoc.tex を処理します。このコマンドは、引数に指定されたファイルだけを “\include” するためのコマンドですが、ここでは何も\include したくないので、引数には何も指定をしません。しかし、\input で指定されているファイルは読み込まれます。したがって、目次や索引や変更履歴のファイルが処理されます。この処理は、主に、これらでエラーが出るかどうかの確認です。

```
179 echo "\includeonly{}" > ltxdoc.cfg
180 uplatex upldoc.tex
```

[5] uplatex コマンドも実際には UTF-8 モードで実行する必要がありますが、デフォルトの内部漢字コードが UTF-8 に設定されているはずですので、-kanji=utf8 を付けなくても処理できると思います。

最後に、再び ltxdoc.cfg を空にして、upldoc.tex を処理をします。本文を 1 ページから開始していますので、この後、もう一度処理をする必要はありません。

```
echo "" > ltxdoc.cfg
uplatex upldoc.tex
# EOT
```
⟨/shprog⟩

C.2 perl スクリプト dstcheck.pl

DOCSTRIP 文書ファイルは、LaTeX のソースとその文書を同時に管理する方法として、とてもすぐれていると思います。しかし、たとえば jclasses.dtx のように、条件が多くなると、入れ子構造がわからなくなってしまいがちです。LaTeX で処理すれば、エラーによってわかりますが、文書ファイルが大きくなると面倒です。

ここでは、DOCSTRIP 文書ファイルの入れ子構造を調べるのに便利な、perl スクリプトについて説明をしています。

この perl スクリプトの使用方法は次のとおりです。

```
perl dstcheck.pl file-name
```

C.2.1 dstcheck.pl の内容

最初に、この perl スクリプトが何をするのかを簡単に記述したコメントを付けます。

⟨*plprog⟩
```
##
## DOCSTRIP 文書内の環境や条件の入れ子を調べる perl スクリプト
##
```

このスクリプトは、入れ子の対応を調べるために、次のスタックを用います。⟨ 条件 ⟩ あるいは ⟨ 環境 ⟩ を開始するコードが現れたときに、それらはスタックにプッシュされ、終了するコードでポップされます。したがって、現在の ⟨ 条件 ⟩ あるいは ⟨ 環境 ⟩ と、スタックから取り出した ⟨ 条件 ⟩ あるいは ⟨ 環境 ⟩ と一致すれば、対応が取れているといえます。そうでなければエラーです。

@dst スタックには、⟨ 条件 ⟩ が入ります。条件の開始は、“%<*⟨ 条件 ⟩>” です。条件の終了は、“%</⟨ 条件 ⟩>” です。⟨ 条件 ⟩ には、>文字が含まれません。@env スタックには、⟨ 環境 ⟩ が入ります。

先頭を明示的に示すために、ダミーの値を初期値として用います。スタックは、⟨ 条件 ⟩ あるいは ⟨ 環境 ⟩ の名前と、その行番号をペアにして操作をします。

```
push(@dst,"DUMMY"); push(@dst,"000");
push(@env,"DUMMY"); push(@env,"000");
```

この while ループの中のスクリプトは、文書ファイルの 1 行ごとに実行をします。

```
while (<>) {
```

入力行が条件を開始する行なのかを調べます。条件の開始行ならば、@dst スタックに ⟨ 条件 ⟩ と行番号をプッシュします。

```
192  if (/^%<\*([^>]+)>/) { # check conditions
193    push(@dst,$1);
194    push(@dst,$.);
```

そうでなければ、条件の終了行なのかを調べます。現在行が条件の終了を示している場合は、@dst スタックをポップします。

```
195  } elsif (/^%<\/([^>]+)>/) {
196    $linenum = pop(@dst);
197    $conditions = pop(@dst);
```

現在行の ⟨ 条件 ⟩ と、スタックから取り出した ⟨ 条件 ⟩ が一致しない場合、その旨のメッセージを出力します。

なお、DUMMY と一致した場合は、一番外側のループが合っていないということを示しています。このとき、これらのダミー値をスタックに戻します。いつでもスタックの先頭をダミー値にするためです。

```
198    if ($1 ne $conditions) {
199      if ($conditions eq "DUMMY") {
200        print "$ARGV: `</$1>' (l.$.) is not started.\n";
201        push(@dst,"DUMMY");
202        push(@dst,"000");
203      } else {
204        print "$ARGV: `<*$conditions>' (l.$linenum) is ended ";
205        print "by `<*$1>' (l.$.)\n";
206      }
207    }
208  }
```

環境の入れ子も条件と同じように調べます。

verbatim 環境のときに、その内側をスキップしていることに注意をしてください。

```
209  if (/^% *\\begin\{verbatim\}/) { # check environments
210    while(<>) {
211        last if (/^% *\\end\{verbatim\}/);
212    }
213  } elsif (/^% *\\begin\{([^{}]+)\}\{(.*)\}/) {
214    push(@env,$1);
215    push(@env,$.);
216  } elsif (/^% *\\begin\{([^{}]+)\}/) {
217    push(@env,$1);
218    push(@env,$.);
219  } elsif (/^% *\\end\{([^{}]+)\}/) {
220    $linenum = pop(@env);
221    $environment = pop(@env);
222    if ($1 ne $environment) {
223      if ($environment eq "DUMMY") {
224        print "$ARGV: `\\end{$1}' (l.$.) is not started.\n";
```

```
225         push(@env,"DUMMY");
226         push(@env,"000");
227       } else {
228         print "$ARGV: \\begin{$environement} (l.$linenum) is ended ";
229         print "by \\end{$1} (l.$.)\n";
230       }
231     }
232   }
```

ここまでが、最初の`while`ループです。

```
233 }
```

文書ファイルを読み込んだ後、終了していない条件があるかどうかを確認します。すべての条件の対応がとれていれば、この時点での`@dst`スタックにはダミー値しか入っていません。したがって、対応が取れている場合は、最初の２つのポップによって、ダミー値が設定されます。ダミー値でなければ、ダミー値になるまで、取り出した値を出力します。

```
234 $linenum = pop(@dst);
235 $conditions = pop(@dst);
236 while ($conditions ne "DUMMY") {
237     print "$ARGV: `<*$conditions>' (l.$linenum) is not ended.\n";
238     $linenum = pop(@dst);
239     $conditions = pop(@dst);
240 }
```

環境の入れ子についても、条件の入れ子と同様に確認をします。

```
241 $linenum = pop(@env);
242 $environment = pop(@env);
243 while ($environment ne "DUMMY") {
244     print "$ARGV: `\\begin{$environment}' (l.$linenum) is not ended.\n";
245     $linenum = pop(@env);
246     $environment = pop(@env);
247 }
248 exit;
249 ⟨/plprog⟩
```

C.3 DOCSTRIP バッチファイル

ここでは、付録 C.1 と付録 C.2 で説明をした二つのスクリプトを、このファイルから取り出すための DOCSTRIP バッチファイルについて説明をしています。

まず、DOCSTRIP パッケージをロードします。また、実行経過のメッセージを出力しないようにしています。

```
250 ⟨*Xins⟩
251 \input docstrip
252 \keepsilent
```

DOCSTRIP プログラムは、連続する二つのパーセント記号 (%%) ではじまる行をメタコメントとみなし、条件によらず出力をします。しかし、“%” は TEX ではコメントであっても、sh や perl にとってはコメントではありません。そこで、メタコメントとして出力する文字を “##” と変更します。

```
{\catcode`#=12 \gdef\MetaPrefix{## }}
```

そして、プリアンブルに出力されるメッセージを宣言します。ここでは、とくに何も指定していませんが、宣言をしないとデフォルトの記述が ‘%%’ 付きで出力されてしまうため、それを抑制する目的で使用しています。

```
\declarepreamble\thispre
\endpreamble
\usepreamble\thispre
```

ポストアンブルも同様に、宣言をしないと ‘\endinput’ が出力されます。

```
\declarepostamble\thispost
\endpostamble
\usepostamble\thispost
```

\generate コマンドで、どのファイルに、どのファイルのどの部分を出力するのかを指定します。

```
\generate{
   \file{dstcheck.pl}{\from{uplatex.dtx}{plprog}}
   \file{mkpldoc.sh}{\from{uplatex.dtx}{shprog}}
}
\endbatchfile
```

⟨/Xins⟩

変更履歴