upLaTeX 2ε について

Ken Nakano & Japanese TeX Development Community & TTK

作成日：2016/05/08

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が、内容はソースコード部分を除いて元の pLaTeX 2ε の文書と全く同一です。

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、pLaTeX 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。pLaTeX 2$_\varepsilon$ の機能についての説明は、[7] を参照してください。日本語 TeX については [6] を参照してください。

pLaTeX 2$_\varepsilon$ では [2] で説明されている、いくつかの拡張コマンドの動作を修正しています。その詳細については、`plext.dtx` を参照してください。

LaTeX の機能については、[4] や [3] などを参照してください。新しい機能については `usrguide.tex` 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と、DOCSTRIP のためのオプションについて述べています。

第 2 節 pLaTeX 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。

第 3 節 旧バージョンの pLaTeX との互換性について述べています。

付録 A pLaTeX 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説明をしています。

付録 B 付録 A 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。

1.1 DOCSTRIP プログラムのためのオプション

この文書を 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。

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	pLaTeX 2$_\varepsilon$ 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

1.1.1 ファイルの取り出し方

たとえば、この文書の “plcore” の部分を “platex.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. platex 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（platex）を入力する。
6. platex.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、platex.ltx が作られます。

あるいは、次のような内容のファイル batch.ins を作成し、platex fmt.ins することでも platex.ltx を作ることができます。

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

DOCSTRIP プログラムの詳細は、docstrip.dtx を参照してください。

2 pLaTeX 2ε の機能について

pLaTeX 2ε 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、pLaTeX 2ε 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。

クラスファイルとパッケージファイルは、従来、スタイルファイルと呼ばれていたものです。LaTeX 2$_\varepsilon$ ではそれらを、レイアウトに関するものをクラスファイルと呼び、マクロの拡張をするものをパッケージファイルと呼んで区別するようになりました。

TeX 文書が使用するクラスは、文書のプリアンブルで`\documentclass` コマンドを用いて指定します。`\documentclass` ではなく、旧版の`\documentstyle` を用いると、自動的に *2.09* **互換モード**に入ります。互換モードは旧版の文書を組版するためだけに作られていますので、新しく文書を作成する場合は、`\documentclass` コマンドを用いてください。互換モードでは LaTeX の新機能も使えなくなります。

旧版では、`\documentstyle` のオプションでマクロファイルを読み込んでいましたが、LaTeX では、`\usepackage` コマンドを用いて読み込みます。

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 “platex.ltx” を`iniptex` プログラムで処理します。

次のリストが、その内容です。ただし、このバージョンでは、LaTeX から pLaTeX 2$_\varepsilon$ への拡張を `plcore.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 `platex.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、`plcore.ltx` には pLaTeX 2$_\varepsilon$ 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
\makeatother
\the\everyjob
\let\dump\orgdump
\dump
⟨plcore⟩\endinput
⟨/plcore⟩
```

実際に pLaTeX 2$_\varepsilon$ への拡張を行なっている `plcore.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。

- plvers.dtx は、pLaTeX 2$_\varepsilon$ のフォーマットバージョンを定義しています。
- plfonts.dtx は、NFSS2 を拡張しています。
- plcore.dtx 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、pldefs.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、pLaTeX 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、pldefs.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。pldefs.cfg は pldefs.ltx の代わりに読み込まれます。

2.1.1 バージョン

pLaTeX 2$_\varepsilon$ のバージョンやフォーマットファイル名は、plvers.dtx で定義しています。

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。

pLaTeX 2$_\varepsilon$ の NFSS2 は、フォントを切替えるコマンドを指定するときに、それが欧文書体か和文書体のいずれかを対象とするものかを、できるだけ意識しないようにする方向で拡張しています。いいかえれば、コマンドが（可能な限りの）判断をします。したがって数多くある英語版のクラスファイルやパッケージファイルなどで書体の変更を行っている箇所を修正する必要はあまりありません。

plfonts.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。

NFSS2 についての詳細は、LaTeX 2$_\varepsilon$ に付属の fntguide.tex を参照してください。

2.1.3 出力ルーチンとフロート

plcore.dtx は、次の項目に関するコマンドを日本語処理用に修正や拡張をしています。

- プリアンブルコマンド
- 改ページ

- 改行
- オブジェクトの出力順序
- トンボ
- 脚注マクロ
- 相互参照
- 疑似タイプ入力

2.2 クラスファイルとパッケージファイル

クラスファイルとパッケージファイルは、従来、スタイルファイルと呼ばれていたものです。LaTeX ではそれらを、レイアウトに関するものをクラスファイルと呼び、マクロの拡張をするものをパッケージファイルと呼んで区別するようになりました。

pLaTeX 2_ε 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、オリジナルのファイルを修正しています。修正箇所には “`platex`” 条件が付けられています。

pLaTeX 2_ε に付属のクラスファイルは、次のとおりです。

- jbook.cls, jarticle.cls, jreport.cls

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

- tbook.cls, tarticle.cls, treport.cls

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

- jltxdoc.cls

  `.dtx` ファイルを組版するためのクラスファイル。`jltxdoc.dtx` から作成される。

また、pLaTeX 2_ε に付属のパッケージファイルは、次のとおりです。

- oldpfont.sty

  pLaTeX 2.09 のフォントコマンドを提供するパッケージ。`pl209.dtx` から作成される。

- ptrace.sty

  `tracefnt.sty` で再定義された NFSS2 コマンドを pLaTeX 2_ε 用に再々定義するためのパッケージ。

- ascmac.sty, tascmac.sty

  旧バージョンの pLaTeX で配布されていたファイル。

- plext.sty

  縦組用の拡張コマンドなどが定義されているファイル。

- nidanfloat.sty

  二段組時に段抜きのフロートをページ下部にも配置可能にするパッケージ。

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 2.09 との互換性

pLaTeX 2_ε は、LaTeX の上位互換という形を取っていますが、いくつかのパラメータなども変更しています。したがって英文書など、LaTeX でも処理できるファイルを pLaTeX 2_ε で処理しても、完全に同じ結果になるとは限りません。これは、英語版の LaTeX でも同じです。詳細は、LaTeX 2_ε に付属の `usrguide.tex` を参照してください。

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、それらが pLaTeX 2_ε で拡張しているコマンドと同じ名前のコマンドを再定義している場合は、コマンドの拡張の仕方によってはエラーになることもあります。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 パッケージを用意しました。platexrelease パッケージを用いると、過去の pLaTeX 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 pLaTeX 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　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 150 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。pldoc.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。plext.dtx の中でサンプルを組み立てていますので、plext パッケージが必要です。

```
\documentclass{jltxdoc}
\usepackage{plext}
\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 from uplvers.dtx
\makeatletter
\def\patchdate{0}%% Modified (May 8, 2016)
\begingroup
```

```
   \def\ProvidesFile#1\pfmtversion#2{\date{#2}\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\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}   % pLaTeX version

\DocInclude{uplfonts}  % NFSS2 commands

%\DocInclude{plcore}   % kernel commands

%\DocInclude{plext}    % external commands

%\DocInclude{pl209}    % 2.09 compatibility mode commands

\DocInclude{ukinsoku}  % kinsoku parameter

\DocInclude{ujclasses} % Standard class

%\DocInclude{jltxdoc}  % dtx documents class

%\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 B.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⟩

B　おまけプログラム

B.1　シェルスクリプト mkpldoc.sh

ここでは、pLaTeX 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[4]について説明をしています。また、このシェルスクリプトを取り出すための、DOCSTRIP バッチファイルについても説明をしています。

このシェルスクリプト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。

B.1.1 mkpldoc.sh の内容

まず、以前に pldoc.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。

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

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

そして、pldoc.tex を処理します。

```
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。

-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。

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

ltxdoc.cfg の内容を\includeonly{}にし、pldoc.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。

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

最後に、再び ltxdoc.cfg を空にして、pldoc.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⟩
```

B.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
```

B.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。

183 ⟨*plprog⟩
184 ##
185 ## DOCSTRIP 文書内の環境や条件の入れ子を調べる perl スクリプト
186 ##

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。

187 push(@dst,"DUMMY"); push(@dst,"000");
188 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。

189 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。

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

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。

```
193   } elsif (/^%<\/([^>]+)>/) {
194     $linenum = pop(@dst);
195     $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。

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

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。

```
207   if (/^% *\\begin\{verbatim\}/) { # check environments
208     while(<>) {
209         last if (/^% *\\end\{verbatim\}/);
210     }
211   } elsif (/^% *\\begin\{([^{}]+)\}\{(.*)\}/) {
212     push(@env,$1);
213     push(@env,$.);
214   } elsif (/^% *\\begin\{([^{}]+)\}/) {
215     push(@env,$1);
216     push(@env,$.);
217   } elsif (/^% *\\end\{([^{}]+)\}/) {
218     $linenum = pop(@env);
219     $environment = pop(@env);
220     if ($1 ne $environment) {
221       if ($environment eq "DUMMY") {
222         print "$ARGV: `\\end{$1}' (l.$.) is not started.\n";
223         push(@env,"DUMMY");
224         push(@env,"000");
225       } else {
226         print "$ARGV: \\begin{$environement} (l.$linenum) is ended ";
227         print "by \\end{$1} (l.$.)\n";
228       }
229     }
```

```
230   }
```
ここまでが、最初の`while`ループです。
```
231 }
```
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。
```
232 $linenum = pop(@dst);
233 $conditions = pop(@dst);
234 while ($conditions ne "DUMMY") {
235     print "$ARGV: `<*$conditions>' (l.$linenum) is not ended.\n";
236     $linenum = pop(@dst);
237     $conditions = pop(@dst);
238 }
```
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。
```
239 $linenum = pop(@env);
240 $environment = pop(@env);
241 while ($environment ne "DUMMY") {
242     print "$ARGV: `\\begin{$environment}' (l.$linenum) is not ended.\n";
243     $linenum = pop(@env);
244     $environment = pop(@env);
245 }
246 exit;
247 ⟨/plprog⟩
```

B.3 DOCSTRIP バッチファイル

ここでは、付録 B.1 と付録 B.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。
```
248 ⟨*Xins⟩
249 \input docstrip
250 \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。
```
251 {\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⟩
```

References

[1] Donald E. Knuth. "*The TEXbook*". Addison-Wesley, 1984. (邦訳：斎藤信男監修, 鷺谷好輝訳, TEX ブック 改訂新版, アスキー出版局, 1989)

[2] インプレス・ラボ監修, アスキー書籍編集部編『縦組対応 パーソナル日本語 TEX』アスキー出版局, 1994

[3] Michel Goossens, Frank Mittelbach, Alexander Samarin. "*The LATEX Companion*". Addison-Wesley, 1994.

[4] Laslie Lamport. "*LATEX: A Document Preparation System*". Addison-Wesley, second edition, 1994.

[5] Laslie Lamport. "*LATEX: A Document Preparation System*". Addison-Wesley, 1986. (邦訳：倉沢良一監修, 大野俊治・小暮博通・藤浦はる美訳, 文書処理システム LATEX, アスキー, 1990)

[6] アスキー出版技術部責任編集『日本語 TEX テクニカルブック I』アスキー, 1990.

[7] 中野 賢『日本語 LATEX 2_ε ブック』アスキー, 1996.

[8] 河野真治著『入門 perl』アスキー出版局, 1994