

pL^AT_EX 2_ε について

Ken Nakano & Japanese T_EX Development Community

作成日：2016/05/08

注意：

これは、株式会社アスキー（現アスキー・メディアワークス¹）が配布している **pL^AT_EX 2_ε** の付属文書ではありません。アスキーのオリジナル版から **fork** したコミュニティ版 **pL^AT_EX 2_ε** の付属文書です。

2010 年以降、アスキー pT_EX は、国際的に広く使われている T_EX Live というディストリビューションに取り込まれ、そこで独自の改良や仕様変更が加えられてきました。最近の T_EX Live や W32T_EX では、pL^AT_EX も元々の pT_EX ではなく、その拡張版 ϵ -pT_EX をエンジンとして用いるようになっています。また、pL^AT_EX のベースである L^AT_EX も更新が進められています。

こうした流れにあわせた新しい pL^AT_EX として、アスキー版から fork して日本語 T_EX 開発コミュニティ (Japanese T_EX Development Community) が配布しているものが、コミュニティ版 pL^AT_EX です。開発中の版は GitHub のリポジトリ²で管理しています。コミュニティ版 pL^AT_EX はアスキー版とは異なりますので、バグレポートはアスキー宛てではなく、日本語 T_EX 開発コミュニティに報告してください。T_EX Forum や GitHub の Issue システムが利用できます。

この文書 (platex.pdf) はコミュニティ版 pL^AT_EX の概要を説明したもの³で、実際の pL^AT_EX のソースコードは pldoc.pdf で説明しています。

¹アスキー日本語 T_EX (pT_EX)、<http://ascii.asciimw.jp/pb/ptex/>

²<https://github.com/texjporg/platex>

³ただし、アスキーの中野さんによって書かれた古い文書を一部アップデートしたものですので、最近の状況に即していない部分が残っているかもしれません。

1 概要

この文書は、 $\text{p}\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ 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。 $\text{p}\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ 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。

$\text{p}\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ 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、`plex.tdx` を参照してください。

$\text{L}\text{A}\text{T}\text{E}\text{X}$ 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 $\text{p}\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ 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 $\text{p}\text{L}\text{A}\text{T}\text{E}\text{X}$ との互換性について述べています。

付録 A $\text{p}\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ の `tdx` 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
<code>plcore</code>	フォーマットファイルを作るためのファイルを生成
<code>pldoc</code>	$\text{p}\text{L}\text{A}\text{T}\text{E}\text{X} 2_{\epsilon}$ 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
<code>shprog</code>	上記のファイルを作成するための <code>sh</code> スクリプトを生成
<code>plprog</code>	入れ子構造を調べる簡単な <code>perl</code> スクリプトを生成
<code>Xins</code>	上記の <code>sh</code> スクリプトや <code>perl</code> 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 pL^AT_EX 2_ε の機能について

pL^AT_EX 2_ε の機能は、いくつか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、pL^AT_EX 2_ε 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 T_EX 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。L^AT_EX 2_ε 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。

T_EX 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 L^AT_EX 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、L^AT_EX 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 T_EX 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、L^AT_EX から pL^AT_EX 2_ε 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 L^AT_EX のコマンドが、`plcore.ltx` には pL^AT_EX 2_ε で拡張したコマンドが定義されています。

```
1 <plcore>
2 \let\orgdump\dump
3 \let\dump\relax
4 \input latex.ltx
5 \edef\platexBANNER{\the\everyjob}% save LaTeX banner
6 \typeout{*****^J%
7         *^J%
8         * making pLaTeX format^J%
9         *^J%
10        *****}
11 \makeatletter
12 \input plcore.ltx
13 \makeatother
14 \the\everyjob
15 \let\dump\orgdump
16 \dump
17 <plcore>\endinput
18 </plcore>
```

実際に pL^AT_EX 2_ε 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、 $\mathrm{p}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\ 2_{\epsilon}$ 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、 $\mathrm{p}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\ 2_{\epsilon}$ 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

$\mathrm{p}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\ 2_{\epsilon}$ 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

$\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$ では、フォント選択機構として NFSS2 を用いています。 $\mathrm{p}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\ 2_{\epsilon}$ 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。

$\mathrm{p}\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\ 2_{\epsilon}$ 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、 $\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}\ 2_{\epsilon}$ に付属の `fntguide.tex` を参照してください。

2.1.3 出カ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。L^AT_EX 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。pL^AT_EX 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。

pL^AT_EX 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、pL^AT_EX 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
pL^AT_EX 2.09 のフォントコマンドを提供するパッケージ。p1209.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 pL^AT_EX 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 2ε` 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 2ε` 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 2ε` 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 2ε` 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。

```
19 \*pldoc)
20 \begin{filecontents}{pldoc.dic}
21 西暦      せいいき
22 和暦      われき
23 \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。

```
24 \documentclass{jltxdoc}
25 \usepackage{plext}
26 \listfiles
27
```

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。

```
28 \DoNotIndex{\def,\long,\edef,\xdef,\gdef,\let,\global}
29 \DoNotIndex{\if,\ifnum,\ifdim,\ifcat,\ifmmode,\ifvmode,\ifhmode,%
30             \iftrue,\iffalse,\ifvoid,\ifx,\ifeof,\ifcase,\else,\or,\fi}
31 \DoNotIndex{\box,\copy,\setbox,\unvbox,\unhbox,\hbox,%
32             \vbox,\vtop,\vcenter}
33 \DoNotIndex{\@empty,\immediate,\write}
34 \DoNotIndex{\egroup,\bgroup,\expandafter,\begingroup,\endgroup}
35 \DoNotIndex{\divide,\advance,\multiply,\count,\dimen}
36 \DoNotIndex{\relax,\space,\string}
37 \DoNotIndex{\csname,\endcsname,\@spaces,\openin,\openout,%
38             \closein,\closeout}
39 \DoNotIndex{\catcode,\endinput}
40 \DoNotIndex{\jobname,\message,\read,\the,\m@ne,\noexpand}
41 \DoNotIndex{\hsize,\vsize,\hskip,\vskip,\kern,\hfil,\hfill,\hss,\vss,\unskip}
42 \DoNotIndex{\m@ne,\z@,\z@skip,\@ne,\tw@,\p@,\@minus,\@plus}
43 \DoNotIndex{\dp,\wd,\ht,\setlength,\addtolength}
44 \DoNotIndex{\newcommand,\renewcommand}
45
```

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。

```
46 \IndexPrologue{\part*{索引}}%
47             \markboth{索引}{索引}%
48             \addcontentsline{toc}{part}{索引}%
49 イタリアック体の数字は、その項目が説明されているページを示しています。
```



```

50 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。
51 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。}
52 %
53 \GlossaryPrologue{\part*{変更履歴}}%
54     \markboth{変更履歴}{変更履歴}%
55     \addcontentsline{toc}{part}{変更履歴}}
56

```

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。

```

57 \makeatletter
58 \def\changes@#1#2#3{%
59     \let\protect\@unexpandable@protect
60     \edef\@tempa{\noexpand\glossary{#2\space\currentfile\space#1\levelchar
61         \ifx\saved@macroname\@empty
62             \space\actualchar\generalname
63         \else
64             \expandafter\@gobble
65             \saved@macroname\actualchar
66             \string\verb\quotechar*%
67             \verbatimchar\saved@macroname
68             \verbatimchar
69         \fi
70         :\levelchar #3}}%
71     \@tempa\endgroup\@esphack}
72 \makeatother
73 \RecordChanges
74 \CodelineIndex
75 \EnableCrossrefs
76 \setcounter{IndexColumns}{2}
77 \settowidth\MacroIndent{\ttfamily\scriptsize 000\ }

```

ここからが本文ページとなります。

```

78 \begin{document}
79 \title{The p\LaTeXe\ Sources}
80 \author{Ken Nakano \& Japanese \TeX\ Development Community}
81
82 % This command will be used to input the patch file
83 % if that file exists.
84 \newcommand{\includepatch}{%
85     \def\currentfile{plpatch.ltx}
86     \part{plpatch}
87     {\let\ttfamily\relax
88         \xdef\filekey{\filekey, \thepart={\ttfamily\currentfile}}}%
89     Things we did wrong\ldots
90     \IndexInput{plpatch.ltx}}
91
92 % Get the date from plvers.dtx
93 \makeatletter
94 \def\patchdate{0}%% Modified (May 8, 2016)
95 \begin{group}

```

```

96   \def\ProvidesFile#1\pfmtversion#2{\date{#2}\endinput}
97   \input{plvers.dtx}
98   \global\let\X@date=\@date
99
100  % Add the patch version if available.
101   \long\def\Xdef#1#2#3\def#4#5{%
102     \xdef\X@date{#2}%
103     \xdef\patchdate{#5}%
104     \endinput}%
105   \InputIfFileExists{plpatch.ltx}
106     {\let\def\Xdef{\global\let\includeltpatch\relax}
107   \endgroup
108
109   \ifx\@date\X@date
110     \def\Xpatch{0}
111     \ifx\patchdate\Xpatch\else
112       \edef\@date{\@date\space Patch level\patchdate}
113     \fi
114   \else
115     \@warning{plpatch.ltx does not match plvers.dtx!}
116     \let\includeltpatch\relax
117   \fi
118   \makeatother
119
120   \pagenumbering{roman}
121   \maketitle
122   \renewcommand\maketitle{}
123   \tableofcontents
124   \clearpage
125   \pagenumbering{arabic}
126
127   \DocInclude{plvers}    % pLaTeX version
128
129   \DocInclude{plfonts}  % NFSS2 commands
130
131   \DocInclude{plcore}   % kernel commands
132
133   \DocInclude{plext}    % external commands
134
135   \DocInclude{pl209}    % 2.09 compatibility mode commands
136
137   \DocInclude{kinsoku}  % kinsoku parameter
138
139   \DocInclude{jclasses} % Standard class
140
141   \DocInclude{jltxdoc}  % dtx documents class
142
143   %\includeltpatch      % patch file (comment out May 8, 2016)
144

```

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

```
145 \StopEventually{\end{document}}
146
```

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。

```
147 \clearpage
148 \pagestyle{headings}
149 % Make TeX shut up.
150 \hbadness=10000
151 \newcount\hbadness
152 \hfuzz=\maxdimen
153 %
154 \PrintChanges
155 \clearpage
156 %
157 \begingroup
158   \def\endash{--}
159   \catcode'\-\active
160   \def-{\futurelet\temp\indexdash}
161   \def\indexdash{\ifx\temp-\endash\fi}
162
163   \PrintIndex
164 \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。

```
165 \let\PrintChanges\relax
166 \let\PrintIndex\relax
167 \end{document}
168 \end{pdoc}
```

B おまけプログラム

B.1 シェルスクリプト mkpdoc.sh

ここでは、pL^AT_EX 2_ε のマクロ定義ファイルをまとめて組版するとき便利な、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 mkpdoc.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。

```
169 <*shprog>
170 for f in pldoc.toc pldoc.idx pldoc.glo ; do
171 if [ -e $f ]; then rm $f; fi
172 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。

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

そして、pldoc.tex を処理します。

```
174 platex pldoc.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、L^AT_EX 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。

```
175 mendex -s gind.ist -d pldoc.dic -o pldoc.ind pldoc.idx
176 mendex -f -s gglo.ist -o pldoc.gls pldoc.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。

```
177 echo "\includeonly{}" > ltxdoc.cfg
178 platex pldoc.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。

```
179 echo "" > ltxdoc.cfg
180 platex pldoc.tex
181 # EOT
182 </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うがちです。 \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是次のとおりです。

```
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 "$ARGV: \\begin{$environment} (l.$linenum) is ended ";
227             print "by \\end{$1} (l.$.)\n";
228         }
229     }
230 }
```

```
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(%%) で始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 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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、プリアンプ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。

```
252 \declarepreamble\thispre
253 \endpreamble
254 \usepreamble\thispre
```

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

```
255 \declarepostamble\thispost
256 \endpostamble
257 \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。

```
258 \generate{
259   \file{dstcheck.pl}{\from{platex.dtx}{plprog}}
260   \file{mkpldoc.sh}{\from{platex.dtx}{shprog}}
261 }
262 \endbatchfile
263 </Xins>
```


References

- [1] Donald E. Knuth. “*The T_EXbook*”. Addison-Wesley, 1984. (邦訳：斎藤信男監修, 鷺谷好輝訳, T_EX 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 T_EX』アスキー出版局, 1994
- [3] Michel Goossens, Frank Mittelbach, Alexander Samarin. “*The L^AT_EX Companion*”. Addison-Wesley, 1994.
- [4] Laslie Lamport. “*L^AT_EX: A Document Preparation System*”. Addison-Wesley, second edition, 1994.
- [5] Laslie Lamport. “*L^AT_EX: 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 L^AT_EX, アスキー, 1990)
- [6] アスキー出版技術部責任編集『日本語 T_EX テクニカルブック I』アスキー, 1990.
- [7] 中野 賢『日本語 L^AT_EX 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