

upL^AT_EX 2_ε について

Ken Nakano & Japanese T_EX Development Community & TTK

作成日：2017/09/24

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 pL^AT_EX 2_ε の付属文書を upL^AT_EX 2_ε 用書き換えたものです。

アスキー pT_EX 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) pT_EX 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、pT_EX の内部コードを Unicode 化した拡張版が upT_EX です。また、upT_EX 上で用いる Unicode 版 pL^AT_EX が upL^AT_EX です¹。現在の upL^AT_EX は、日本語 T_EX 開発コミュニティが配布しているコミュニティ版 pL^AT_EX² をベースにしています。開発中の版は pL^AT_EX 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 upT_EX や upL^AT_EX 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、 $\text{up}\text{\LaTeX} 2_{\epsilon}$ 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。元となっている $\text{p}\text{\LaTeX} 2_{\epsilon}$ や $\text{\LaTeX} 2_{\epsilon}$ については、それぞれ $\text{p}\text{\LaTeX} 2_{\epsilon}$ と $\text{\LaTeX} 2_{\epsilon}$ 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 $\text{up}\text{\LaTeX} 2_{\epsilon}$ 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。

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 $\text{up}\text{\LaTeX} 2_{\epsilon}$ 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 $\text{up}\text{\LaTeX} 2_{\epsilon}$ の機能について

$\text{up}\text{\LaTeX} 2_{\epsilon}$ 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、 $\text{up}\text{\LaTeX} 2_{\epsilon}$ 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 $\text{p}\text{\LaTeX} 2.09$ ユーザへの注意：

$\text{up}\text{\LaTeX}$ 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。 $\text{\LaTeX} 2_{\epsilon}$ 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 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ Live や $\mathrm{W}32\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ではこの処理を簡単にする `fmtutil-sys` あるいは `fmtutil` 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-sys --byfmt uplatex
```

次のリストが、“uplatex.ltx” の内容です。ただし、このバージョンでは、 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ から $\mathrm{upL}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\epsilon}$ 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 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のコマンドが、`uplcore.ltx` には $\mathrm{upL}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\epsilon}$ で拡張したコマンドが定義されています。

```
1 <{*plcore}
```

`latex.ltx` の末尾で使われている `\dump` をいったん無効化します。

```
2 \let\orgdump\dump
```

```
3 \let\dump\relax
```

`latex.ltx` を読み込み、起動時のバナーを保存します。 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ Live の標準的インストールでは、この中で Babel 由来のハイフネーション・パターン `hyphen.cfg` が読み込まれ、そのバージョンも含めて保存されるはずです。

```
4 \input latex.ltx
```

```
5 \edef\platexBANNER{\the\everyjob\noexpand\typeout{}\relax}% save LaTeX banner
```

`uplcore.ltx` を読み込み、この $\mathrm{upL}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のバージョンを表示します。

```
6 \typeout{*****^J%
```

```
7      *^J%
```

```
8      * making upLaTeX format^J%
```

```
9      *^J%
```

```
10     *****}
```

```
11 \makeatletter
```

```
12 \input uplcore.ltx
```

```
13 \the\everyjob
```

起動時に `uplatex.cfg` がある場合、それを読み込むようにします。バージョン 2016/07/01 ではコードを `uplcore.ltx` に入れていましたが、`uplatex.ltx` へ移動しました。

```
14 \everyjob\expandafter{%
```

```
15   \the\everyjob
```

```
16   \IfFileExists{uplatex.cfg}{%
```

```
17     \typeout{*****^J%
```

```

18          * Loading uplatex.cfg.^~J%
19          *****}%
20      \input{uplatex.cfg}}{}%
21 }

フォーマットファイルにダンプします。

22 \let\dump\orgdump
23 \let\orgdump\@undefined
24 \makeatother
25 \dump
26 %\endinput
27 </plcore>

```

実際に $\text{upL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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、 $\text{upL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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、 $\text{pL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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、 $\text{upL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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、`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

$\text{upL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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、 $\text{pL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ では、フォント選択機構として NFSS2 を用いています。 $\text{pL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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 を拡張していますので、 $\text{upL}^{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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

upL^AT_ΕX 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、pL^AT_ΕX 2_ε に含まれるファイルを修正しています。

upL^AT_ΕX 2_ε に付属のクラスファイルは、次のとおりです。

- ujbook.cls, ujournal.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、upL^AT_ΕX 2_ε に付属のパッケージファイルは、次のとおりです。

- uptrace.sty

L^AT_ΕX 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、upL^AT_ΕX 2_ε 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 pL^AT_ΕX との互換性

upL^AT_ΕX 2_ε は、pL^AT_ΕX 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 pL^AT_ΕX 文書を簡単に upL^AT_ΕX 文書に変更することができます。ただし、upL^AT_ΕX では pL^AT_ΕX 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、upL^AT_ΕX 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。L^AT_ΕX 2_ε の仕様に従ってドキュメントを作成してください。

pL^AT_ΕX 向けあるいは L^AT_ΕX 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、pL^AT_ΕX 標準の漢字エンコーディン

グ (JY1, JT1) を前提としたものが $\text{up}\text{\LaTeX}$ 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 $\text{p}\text{\LaTeX}$ では platexrelease パッケージが用意されました。本来は $\text{up}\text{\LaTeX}$ 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 $\text{p}\text{\LaTeX}$ から $\text{up}\text{\LaTeX}$ 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、 $\text{up}\text{\LaTeX}$ 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 $\text{up}\text{\LaTeX}$ 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 $\text{up}\text{\LaTeX}$ 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	$\text{up}\text{\LaTeX}$ 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てませんが、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。

```
28 <*pldoc>
29 \begin{filecontents}{upldoc.dic}
30 西暦      せいでき
31 和暦      われき
32 \end{filecontents}
```

文書クラスには、`jltxdoc` クラスを用います。

```
33 \documentclass{jltxdoc}
34 %\usepackage{plext} %% comment out for upLaTeX
35 \listfiles
36
```

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。

```
37 \DoNotIndex{\def,\long,\edef,\xdef,\gdef,\let,\global}
38 \DoNotIndex{\if,\ifnum,\ifdim,\ifcat,\ifmmode,\ifvmode,\ifhmode,%
39           \iftrue,\iffalse,\ifvoid,\ifx,\ifeof,\ifcase,\else,\or,\fi}
40 \DoNotIndex{\box,\copy,\setbox,\unvbox,\unhbox,\hbox,%
41           \vbox,\vtop,\vcenter}
42 \DoNotIndex{\@empty,\immediate,\write}
43 \DoNotIndex{\egroup,\bgroup,\expandafter,\begingroup,\endgroup}
44 \DoNotIndex{\divide,\advance,\multiply,\count,\dimen}
45 \DoNotIndex{\relax,\space,\string}
46 \DoNotIndex{\csname,\endcsname,\@spaces,\openin,\openout,%
47           \closein,\closeout}
48 \DoNotIndex{\catcode,\endinput}
49 \DoNotIndex{\jobname,\message,\read,\the,\m@ne,\noexpand}
50 \DoNotIndex{\hsize,\vsize,\hskip,\vskip,\kern,\hfil,\hfill,\hss,\vss,\unskip}
51 \DoNotIndex{\m@ne,\z@,\z@skip,\@ne,\tw@,\p@,\@minus,\@plus}
52 \DoNotIndex{\dp,\wd,\ht,\setlength,\addtolength}
53 \DoNotIndex{\newcommand,\renewcommand}
54
```

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。

```
55 \IndexPrologue{\part*{索引}}%
56           \markboth{索引}{索引}%
57           \addcontentsline{toc}{part}{索引}%
58 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。
59 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。
60 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。}
61 %
62 \GlossaryPrologue{\part*{変更履歴}}%
63           \markboth{変更履歴}{変更履歴}%
64           \addcontentsline{toc}{part}{変更履歴}}
65
```

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。

```
66 \makeatletter
67 \def\changes@#1#2#3{%
68   \let\protect\@unexpandable@protect
69   \edef\@tempa{\noexpand\glossary{#2\space\currentfile\space#1\levelchar
70           \ifx\saved@macroname\@empty
71             \space\actualchar\generalname
72           \else
73             \expandafter\@gobble
74             \saved@macroname\actualchar
75             \string\verb\quotechar*%
76             \verbatimchar\saved@macroname
77             \verbatimchar
78           \fi
79           :\levelchar #3}}%
80   \@tempa\endgroup\@esphack}
81 \makeatother
```



```

82 \RecordChanges
83 \CodelineIndex
84 \EnableCrossrefs
85 \setcounter{IndexColumns}{2}
86 \settowidth\MacroIndent{\ttfamily\scriptsize 000\ }

ここからが本文ページとなります。

87 \begin{document}
88 \title{The up\LaTeXe\ Sources}
89 \author{Ken Nakano \& Japanese \TeX\ Development Community \& TTK}
90
91 % This command will be used to input the patch file
92 % if that file exists.
93 \newcommand{\includelpatch}{%
94   \def\currentfile{uplpatch.ltx}
95   \part{uplpatch}
96   {\let\ttfamily\relax
97     \xdef\filekey{\filekey, \thepart={\ttfamily\currentfile}}}%
98   Things we did wrong\ldots
99   \IndexInput{uplpatch.ltx}}
100
101 % Get the date and patch level from uplvers.dtx
102 \makeatletter
103 \let\patchdate=\@empty
104 \begingroup
105   \def\ProvidesFile#1\pfmtversion#2#3\ppatch@level#4{%
106     \date{#2}\xdef\patchdate{#4}\endinput}
107   \input{uplvers.dtx}
108 \global\let\X@date=\@date
109
110 % Add the patch version if available.
111 \long\def\Xdef#1#2#3\def#4#5{%
112   \xdef\X@date{#2}%
113   \xdef\patchdate{#5}%
114   \endinput}%
115 \InputIfFileExists{uplpatch.ltx}
116 {\let\def\Xdef}{\global\let\includelpatch\relax}
117 \endgroup
118
119 \ifx\@date\X@date
120   \def\Xpatch{0}
121   \ifx\patchdate\Xpatch\else
122     % number is assumed
123     \ifx\patchdate>0
124       \edef\@date{\@date\space Patch level\space\patchdate}
125     \else
126       \edef\@date{\@date\space Pre-Release\patchdate}
127     \fi\fi
128 \else
129   \@warning{uplpatch.ltx does not match uplvers.dtx!}

```

```

130 \let\includeltpatch\relax
131 \fi
132 \makeatother
133
134 \pagenumbering{roman}
135 \maketitle
136 \renewcommand\maketitle{}
137 \tableofcontents
138 \clearpage
139 \pagenumbering{arabic}
140
141 \DocInclude{uplvers} % upLaTeX version
142
143 \DocInclude{uplfonts} % NFSS2 commands
144
145 %\DocInclude{plcore} % kernel commands (comment out for upLaTeX)
146
147 %\DocInclude{plext} % external commands (comment out for upLaTeX)
148
149 %\DocInclude{pl209} % 2.09 compatibility mode commands (comment out for upLaTeX)
150
151 \DocInclude{ukinsoku} % kinsoku parameter
152
153 \DocInclude{ujclasses} % Standard class
154
155 %\DocInclude{jltxdoc} % dtx documents class (comment out for upLaTeX)
156
157 %\includeltpatch % patch file (comment out May 8, 2016)
158

ltxdoc.cfg に \AtEndOfClass{\OnlyDescription}が指定されている場合は、こ
こで終了します。
159 \StopEventually{\end{document}}
160

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。

161 \clearpage
162 \pagestyle{headings}
163 % Make TeX shut up.
164 \hbadness=10000
165 \newcount\hbadness
166 \hfuzz=\maxdimen
167 %
168 \PrintChanges
169 \clearpage
170 %
171 \begingroup
172 \def\endash{--}

```

```

173 \catcode'\- \active
174 \def-{\futurelet\temp\indexdash}
175 \def\indexdash{\ifx\temp-\endash\fi}
176
177 \PrintIndex
178 \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。

```

179 \let\PrintChanges\relax
180 \let\PrintIndex\relax
181 \end{document}
182 \pdoc)

```

C おまけプログラム

C.1 シェルスクリプト mkpdoc.sh

upL^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の使用方法是次のとおりです。

```
sh mkpdoc.sh
```

C.1.1 mkpdoc.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。

```

183 (*shprog)
184 for f in upldoc.toc upldoc.idx upldoc.glo ; do
185 if [ -e $f ]; then rm $f; fi
186 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。

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

そして、upldoc.tex を処理します。

```
188 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。

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。

up \LaTeX 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。

```
189 mendex -U -s gind.ist -d upldoc.dic -o upldoc.ind upldoc.idx
190 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。

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

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。

```
193 echo "" > ltxdoc.cfg
194 uplatex upldoc.tex
195 # EOT
196 </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
```

⁵uplatex 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。

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。

```
197 <*plprog>
198 ##
199 ## DOCSTRIP 文書内の環境や条件の入れ子を調べる perl スクリプト
200 ##
```

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。

```
201 push(@dst,"DUMMY"); push(@dst,"000");
202 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。

```
203 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。

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

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。

```
207 } elsif (/^%</([>]+)>/) {
208     $linenum = pop(@dst);
209     $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。

```
210 if ($1 ne $conditions) {
211     if ($conditions eq "DUMMY") {
212         print "$ARGV: '</$1>' (1.$.) is not started.\n";
213         push(@dst,"DUMMY");
```

```

214     push(@dst,"000");
215   } else {
216     print "$ARGV: '<*$conditions>' (l.$linenum) is ended ";
217     print "by '<*$1>' (l.$.)\n";
218   }
219 }
220 }

```

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。

```

221 if (/^% *\\begin\\{verbatim\\}/) { # check environments
222   while(<>) {
223     last if (/^% *\\end\\{verbatim\\}/);
224   }
225 } elsif (/^% *\\begin\\{([^{}]+)\\}\\{(.*)\\}/) {
226   push(@env,$1);
227   push(@env,$.);
228 } elsif (/^% *\\begin\\{([^{}]+)\\}/) {
229   push(@env,$1);
230   push(@env,$.);
231 } elsif (/^% *\\end\\{([^{}]+)\\}/) {
232   $linenum = pop(@env);
233   $environment = pop(@env);
234   if ($1 ne $environment) {
235     if ($environment eq "DUMMY") {
236       print "$ARGV: '\\end{$1}' (l.$.) is not started.\n";
237       push(@env,"DUMMY");
238       push(@env,"000");
239     } else {
240       print "$ARGV: \\begin{$environment} (l.$linenum) is ended ";
241       print "by \\end{$1} (l.$.)\n";
242     }
243   }
244 }

```

ここまでの、最初の while ループです。

```
245 }
```

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。

```

246 $linenum = pop(@dst);
247 $conditions = pop(@dst);
248 while ($conditions ne "DUMMY") {
249   print "$ARGV: '<*$conditions>' (l.$linenum) is not ended.\n";
250   $linenum = pop(@dst);

```

```

251     $conditions = pop(@dst);
252 }

```

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。

```

253 $linenum = pop(@env);
254 $environment = pop(@env);
255 while ($environment ne "DUMMY") {
256     print "$ARGV: '\\begin{$environment}' (1.$linenum) is not ended.\n";
257     $linenum = pop(@env);
258     $environment = pop(@env);
259 }
260 exit;
261 </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。

```

262 <*Xins>
263 \input docstrip
264 \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。

```

265 {\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。

```

266 \declarepreamble\thispre
267 \endpreamble
268 \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。

```

269 \declarepostamble\thispost
270 \endpostamble
271 \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。

```

272 \generate{
273     \file{dstcheck.pl}{\from{uplax.tex}{plprog}}

```

```
274 \file{mkpldoc.sh}{\from{uplatex.dtx}{shprog}}
275 }
276 \endbatchfile
277 </Xins>
```


変更履歴

2011/05/07 v1.0c-u00	的に改訂	1
・ p \LaTeX 用から up \LaTeX 用に修正。	2016/06/19 v1.0l-u01	
2016/05/07 v1.0g	・ パッチレベルを <code>uplvers.dtx</code> から	
・ フォーマット作成時に \LaTeX のバ	取得	9
ナーを一旦保存	2016/08/26 v1.0m-u01	
3	・ <code>uplatex.cfg</code> の読み込みを	
2016/05/08 v1.0h-u00	<code>uplcore.ltx</code> から <code>uplatex.ltx</code>	
・ ドキュメントから <code>uplpatch.ltx</code>	へ移動	3
を除外		
9	2016/09/14 v1.0n	
2016/05/12 v1.0i	・ \LaTeX のバナーの保存しかたを改良	3
・ 一時コマンド <code>\orgdump</code> を最終的	2017/09/24 v1.0o	
に未定義へ	・ パッチレベルが負の数の場合を	
4	pre-release 扱いへ	9
2016/06/06 v1.0k-u01		
・ up \LaTeX 用にドキュメントを全体		