pdfTeX for Win32 について

この文書は、 pdfLaTeX で日本語を扱うサンプルとして書かれています。

Contents

1 始めに

pdfTeX 1.40.10-2.2 (Web2c-2009, web-2009, kpathsearch-5.0.0) を Win32(x86) 用に Microsoft VC++ 5.0 で Make したものです。デフォルトで直接 PDF を出力するようにしていますが、ソースで `\pdfoutput=0` とすれば、PDF でなくて DVI を出

力します。PNG, JPEG, EPDF (左下座標が原点の /MediaBox 文のある一ページだけの PDF 画像) を埋め込むことがきます。TIFF 画像のサポートは打ちきられています。bin ディレクトリにある epstopdf.pl は EPSF を EPDF に変換する perl script です。gswin32c.exe に PATH が通っている必要があります。また、 epstopdf.exe は epstopdf.pl を、 perl を通して呼ぶための実行ファイルです。なお，LaTeX 2_ε の graphics package の pdftex オプションでは PNG(.png), JPEG(.jpg), EPDF(.pdf), MetaPost 出力の EPSF(.mps) の 4 種類の画像を埋め込むことができます。ただし以前の pdfTeX にくらべてプリミティブが大きく変化して改善されていますので、graphics package は最新のものを使用して下さい。 ltxpkgs.tar.gz をインストールした場合は、新しい pdfTeX に対応した graphics package がインストールされます。 また hyperref package も最新のものが必要です。これも ltxpkgs.tar.gz を、 TeX のインストールディレクトリ(デフォルトは c:/usr/local) で展開すると、1.40.10-2.2 対応のものがインストールされます。

プリミティブが変更された例を一つあげます。初期の pdfTeX では、プリミティブレベルでは

```
\pdfimage width 12cm height 8cm depth 3pt {image.pdf}
```

とすれば，イメージを挿入できましたが，1.40.10 では

```
\pdfximage width 12cm height 8cm depth 3pt {image.pdf}
```

でまず図を定義します。まだこれでは実際に挿入されません。

```
\pdfrefximage\pdflastximage
```

とすると挿入されます。定義した直後、カウンタ \pdflastximage に番号が入っていて、これをセーブしておけば、同じ図を複数回コールできるようになります。 例えば

```
\newcount\figa \figa=\pdflastximage
```

としておくと、任意の場所で

```
\pdfrefximage\figa
```

によって、同じ図を view できるようになります。 しかし pdfLaTeX では、プリミティブを使うよりも、 graphics パッケージを使って

```
... ...
\usepackage[pdftex]{graphicx}
```

```
... ...
\begin{document}
... ...
\includegraphics[SomeOptions]{somefigure.png}
... ...
\end{document}
```

などとするのが普通であり、お勧めです。 pdfTEX パッケージに入っている、サンプル JPEG 画像をここに含めてみます:

その他,
`\pdfannotlink ---> \pdfstartlink`,
`\pdfform ---> \pdfxform`,
`\pdfrefform ---> \pdfrefxform`

などの変更があります。詳細は texmf/doc/pdftex/base/syntax.txt をご覧下さい。
pdfTeX-1.40.10-2.2 では、 pdfcrypt 機能とTIFF サポート機能が削除されています。pdfcrypt に関しては、Paulo Soares さんによる、java によるプログラムを入れ、これを簡単に使用するための pdfcrypt.exe を作成して入れています。texmf/pdfcrypt にある文書を読んで下さい。
e-TeX 拡張なしの pdfTeX は無くなりました。W32TeX の文書では、pdfTeX と pdfeTeX を同じ意味で使うことにします。基本実行ファイルはpdftex.exe とpdflatex.exe です。pdfetex.exe, pdfelatex.exe も入れてありますが、それぞれpdftex.exe, pdflatex.exe と全くおなじものです。pdftex.exe は pdftex.fmt を使い、pdflatex.exe は pdflatex.fmt を使います。全く同様にpdfetex.exe は pdftex.fmt を使い、pdfelatex.exe は pdflatex.fmt を使います。pdfetex.fmt, pdfelatex.fmtは無いことに注意して下さい。全ての場合に e-TeX の拡張プリミティブが使用できます。e-TeX 2.2 の拡張プリミティブなどについては、e-TeX をインストールするとコピーされる texmf/doc/etex 以下の文書を参照して下さい。

2 pdfTeX での TrueType フォント使用法

pdfTeX では Type 1 fonts に加えて，TrueType fonts も使用することができます。以下にその方法を説明します。

1. texmf.cnf の TTFONTS に TrueType フォントの場所を設定しておきます:

   ```
   TTFONTS = .;c:/windows/fonts;$TEXMF/fonts/truetype//
   ```

2. 付属のドライバプログラム ttftotfm.exe を次のように使って tfm と vf ファイルを作ります:

   ```
   ttftotfm  arial.ttf  8r.enc  arial8t  arial8r
   ```

   (ttftotfm の 使用法は後で説明します。)
   上のコマンドを実行する場合、 ttf フォントファイルや Encoding ファイルは Kpathsearch 機構で探すようにしているので、カレントディレクトリにコピー

しておく必要はありません。こうしてできた tfm ファイル (例では arial8t.tfm と arial8r.tfm) を TEXMF tree の適当な場所 (texmf/fonts/tfm/windows/arial など) に移動しておきます。また vf ファイル (今の例では arial8t.vf) を texmf/fonts/vf/windows/arial に移動しておきます。
ttftotfm.exe は次の三つのコマンドを内部から実行しているにすぎません:

```
ttf2afm -e 8r.enc -o arial8t.afm arial.ttf
afm2tfm arial8t.afm -t 8r.enc -p 8r.enc \
        -v arial8t.vpl arial8r.tfm
vptovf arial8t.vpl arial8t.vf arial8t.tfm
```

afm2tfm の場合は、一行におさまらないので、二行に渡って記述していますが、実際には一行のコマンド行です。

3. texmf/fonts/map/pdftex/updmap/pdftex-base.map を編集して次のようなエントリを追加します:

```
arial8r ArialMT <8r.enc <arial.ttf
```

この記述をするためのヒントは、 2. で tfm ファイルを作るときに表示されます。この変更を有効にするために updmap コマンドを実行します。

以上で pdfTEX ソースで (上の例では arial8t なる名前で) TrueType フォントが使用できるようになります。勿論日本語 TrueType フォントは使用不可です。

```
\font\exam=arial8t at 48pt
{\exam This is a test of TrueType fonts in pdf\TeX.}
\bye
```

さて、上の 2. では必ず encoding file が必要ですが、適当な encoding file が無いときには

```
ttf2afm -c encname -e texnansi.enc ttfname.ttf >nul
```

とすると encname.eMN (MN は数字) なる名前の encoding file が得られます。このディレクトリ (texmf/doc/pdftex/base) に入れてある wingding.enc は

```
ttf2afm -c wingding -e texnansi.enc wingding.ttf >nul
```

としてできた wingding.e10 の名前を wingding.enc に変更し、

```
/WingdingsEncoding [
```

という行だけ手で編集して作成したものです。ファイル wingding.enc を texmf/fonts/enc/pdftex/base なるディレクトリに入れておけば

```
ttftotfm wingding.ttf wingding.enc wingding
```

とすると、 wingding.tfm, rwingding.tfm, wingding.vf ができるので、これらを texmf/fonts/tfm/windows/misc と texmf/fonts/vf/windows/misc に移動し、texmf/fonts/map/pdftex/updmap/pdftex-base.map に

```
rwingding Wingdings-Regular <wingding.enc <wingding.ttf
```

と記述してから updmap コマンドを実行することによって、 wingding.ttf を pdfTEX で使用できるようになります。どのコードが何に対応するかは wingding.enc から大体わかると思います。 (`\char255` は Windows Logo です)。詳しくは wingding.pdf をご覧下さい。
最後に ttftotfm コマンドの使用法を記します。

```
(a) ttftotfm TTFfile.ttf ENCfile TeXfontname MapFontname
(b) ttftotfm TTFfile.ttf ENCfile TeXfontname
(c) ttftotfm TTFfile.ttf ENCfile
```

の三つの使用法があります。(a) が一番一般的な使用法です。TTFfile.ttf は True-Type font ファイル名です。 ENCfile は encoding ファイル名です。TeXfontname は TEX で使用するときのフォント名です。MapFontname は pdftex.map に記述するときのフォント名です。
(b) のように使用すると、 MapFontname は TeXfontname の先頭に r をくっつけた

ものとなります。
(c) のように使用すると、 TeXfontname は TTFfile.ttf から拡張子をとり除いた TTFfile となり、 MapFontname は先頭に r をくっつけた rTTFfile となります。ttf2afm コマンドは，引数なしで実行すると使用法を表示します。

3 xpdf 付属のアプリケーション

xpdf 付属の Derek B. Noonburg さんによるプログラムを 5 個入れておきます。ヘルプファイルはこのディレクトリにあります。なお、これらは xpdf 2.03 付属のユーティリティです。

1. pdftotext.exe
   PDF → Plain Text の変換をします。 `pdftotext -help` で使用法を表示します。 第 4 節で述べる準備をすれば、日本語にも対応するようになり、オプション`-enc Shift-JIS` で Shift-JIS の日本語テキストを出力します。

2. pdftops.exe
   PDF → PS の変換をします。 `pdftops -help` で使用法を表示します。

3. pdfinfo.exe
   PDF ファイルの情報を表示します。 `pdfinfo -help` で使用法を表示します。

4. pdfimages.exe
   PDF に埋め込まれたイメージを抽出するためのものです。 `pdfimages -help` で使用法を表示します。

5. pdffonts.exe
   PDF 内のフォント情報を表示します。

4　pdftotext を日本語対応に

xpdf 付属ユーティリティを日本語対応にする方法を述べます。まずGhostscript で CID フォントを扱っていたり、 gs-7.07 をインストールして見た場合には、既に Resource ディレクトリがあると思います。 Resource ディレクトリが無い場合には新たに作れば良いです。 ここではこれが D:/Resource だとして説明します。違う場合には D:/Resource の部分を読みかえて下さい。
まず最初に、 このディレクトリに pdftotext-supp.zip というファイルがあることを確認して下さい。これを Resource ディレクトリをカレントディレクトリとして展開します:

```
copy pdftotext-supp.zip D:\Resource
D:
cd \Resource
unzip -o pdftotext-supp.zip
del pdftotext-supp.zip
```

次に、TEX のバイナリディレクトリ (例えば c:/usr/local/bin) にテキストファイル xpdfrc が入っているはずですから、 これを編集します。 Resource ディレクトリが実際に D:/Resource である場合には、編集しなくて良いようにしています。たとえば Resource ディレクトリがC:/gs/Resource である場合には、 xpdfrc で D:/Resource とある部分をC:/gs/Resource に変更して下さい。
以上で、pdftotext が日本語対応となります。

5　日本語対応 pdfLATEX

pdfLATEX または pdfeLATEX で日本語を扱う方法を述べます。 Windows 上に限りますが、pdfLATEX と pdfeLATEX で日本語を含むソースを pdf に変換できるようにしてみました。

5.1 条件

omegaj-w32.tar.gz をインストールし、フォント msmin??.tfm, msgoth??.tfm (?? は二桁の16進文字です) をインストール済みであること。 また、稲垣淳さんによる ums パッケージ (ums.tar.gz) をインストールしてあること。 また、TrueType collection (.ttc) でない、適当な日本語 TrueType フォントファイルがあること。(OS にインストールしてある必要はありません。) 入手可能な ipam.ttf, ipag.ttf が適当でしょう。以前は、msmin??.pfb などの Type1 フォントを準備しておく必要がありましたが、最新版として提供している pdfTEX, pdfeTEX はサブフォント機構をサポートしているので、ソースのはじめのほうに、例えば

```
\pdfmapline{=msmin@Unicode@ <ipam.ttf}
\pdfmapline{=msgoth@Unicode@ <ipag.ttf}
```

のように書いておくと、 TrueType フォントを直接埋め込むことができます。但しこの場合、出来上がった pdf から日本語文字を探したりすることはできません。一方 omegaj-w32.tar.gz で提供しているバッチファイルを実行して、msmin??.pfb, msgoth??.pfb を作成してある場合には、ソースに

```
\input jpdftextounicode
\pdfgentounicode=1
```

と書いておくと、出来上がった pdf は本物の日本語 pdf となって、日本語文字列の検索などもできます。もちろんこの場合には、ソースに

```
\pdfmapline{=msmin@Unicode@ <ipam.ttf}
\pdfmapline{=msgoth@Unicode@ <ipag.ttf}
```

と書いたらいけません。また、msmin??.pfb, msgoth??.pfb は、“現在の” omegaj-w32.tar.gz で提供しているバッチファイルで作成したものでなくてはいけません。以前提供していたバッチファイルで作成したものや、 TeXTrace で作成したものは使用できません。 jpdftextounicode.tex はグリフ名と Unicopde との対応表であり、グリフ名をこの表と一致させるためには、現在提供しているバッチファイルで Type1 フォントを作成する必要があるのです。

5.2 実現法

編集した、オリジナルの日本語を含む LaTeX 文書 (e.g., org.tex) を、プログラム topdftex.exe によって、 pdfLaTeX, pdfeLaTeX が理解できるもの (e.g., pdfsrc.tex) に変換します。 org.tex では必ず`\usepackage{ums}` をプリアンブルに記述しておきます。 topdftex.exe は pdftex-w32.tar.gz に同梱しています。 ums.sty は、稲垣淳さんによるものを少し拡張して、日本語 pdfLaTeX でも使用できるようにしたものです。

```
topdftex org.tex pdfsrc.tex
pdflatex pdfsrc
```

により、 pdfsrc.pdf が作成されます。例えば、

```
\documentclass[12pt]{article}
\usepackage{ums}
\begin{document}
\pdfmapline{=msmin@Unicode@ <ipam.ttf}
\pdfmapline{=msgoth@Unicode@ <ipag.ttf}
これは、
pdf\TeX
で日本語を
扱ってみた
ものです。
\end{document}
```

を topdftex.exe で変換すると

```
\documentclass[12pt]{article}
\usepackage{ums}
\begin{document}
\pdfmapline{=msmin@Unicode@ <ipam.ttf}
\pdfmapline{=msgoth@Unicode@ <ipag.ttf}
\UMS{3053}\UMS{308C}\UMS{306F}\PREUMS{3001}%
```

```
pdf\TeX
\UMS{3067}\UMS{65E5}\UMS{672C}\UMS{8A9E}\UMS{3092}%
\UMS{6271}\UMS{3063}\UMS{3066}\UMS{307F}\UMS{305F}%
\UMS{3082}\UMS{306E}\UMS{3067}\UMS{3059}\PREUMS{3002}%
\end{document}
```

のように、日本語文字は全てコマンドで置き換えられます。これらのコマンドは ums.sty で定義されています。

5.3 日本語しおり作成法

hyperref で bookmarks を付ける場合にも最新の ums パッケージと、最新の out2uni.exe を使うと極めて簡単に日本語しおりを作成することができます。topdftex.exe で変換済みの pdfsrc.tex を、クロスリファレンスの解決が終了するまで必要回数 pdfLaTeX にかけます。次に一回だけ

```
out2uni  pdfsrc
```

とします。out2uni.exe は dvipdfm-w32.tar.gz に入っています。これにより、pdf 文字列が Unicode に変換されます。そうして最後にやはり一回だけ `pdflatex pdfsrc` とすると、日本語しおりを含む pdf が出力されます。このディレクトリに入っている jpdfsample.pdf はこのようにして作成した、日本語しおり付きの簡単な pdf です。jpdforg.tex がオリジナルのソースで、 jpdfsample.tex は

```
topdftex jpdforg.tex jpdfsample.tex
```

によって変換されたソースです。なお、使用するクラスファイルや、パッケージは、欧文用のものを使うことに注意して下さい。 pLaTeX や jLaTeX 特有のものは使用できません。topdftex での変換作業が必要なので、面倒なようですが、Makefile などを使用すると、訂正を頻繁にする場合でもそれほど面倒ではありません。例えば、この文書自身は

```
#
# Makefile for jpdfsample.pdf
```

```
# (uses Korn shell sh.exe)
#
all: jpdfsample.pdf
jpdfsample.pdf: jpdfsample.tex
        pdflatex jpdfsample
        (while egrep '^LaTeX Warning: Label' jpdfsample.log; \
         do pdflatex jpdfsample; done)
        out2uni jpdfsample
        pdflatex jpdfsample
jpdfsample.tex: jpdforg.tex
        topdftex jpdforg.tex jpdfsample.tex
test: jpdfsample.pdf
        pdfview jpdfsample
clean:
        rm -f jpdfsample.* *~
```

のような Makefile を作っておくと、 jpdforg.tex を訂正し終わったとき、 make とするだけで、全ての作業は自動的になされます。 make が成功した場合、仕上がり具合を見るには make test とします。 pdfview というのは Acrobat Reader を立ち上げるものと仮定します。なお、上の Makefile は、 texmf/doc/ksh に入れている sh.exe に加えて、cp.exe, rm.exe, egrep.exe の存在を仮定して書いてあります。 sh.exe は、 Windows NT/2000/XP で使用することができます。 Windows 9x ではうまく機能しないでしょう。シェルとして、cmd.exe や command.com を使う通常の Windows 上の、標準的な Makefile は、下のようにしておけばよいです:

```
#
# Makefile for jpdfsample.pdf
# (standard Windows) (cmd.exe or command.com)
#
all: jpdfsample.pdf
jpdfsample.pdf: jpdfsample.tex
```

```
        pdflatex jpdfsample
        pdflatex jpdfsample
        out2uni jpdfsample
        pdflatex jpdfsample
jpdfsample.tex: jpdforg.tex
        topdftex jpdforg.tex jpdfsample.tex
test: jpdfsample.pdf
        pdfview jpdfsample
clean:
        del jpdfsample.*
        del *~
```

A. Kakuto <kakuto@fuk.kindai.ac.jp>