本稿は Linux Japan 誌 2000 年 6 月号に掲載された記事に補筆修正したものです.

PostScript にまつわる話

PS ファイルを扱うツールの紹介も 3 回目で，そろそろ終りにしようと思います．まず，第 1 回目で述べたようにビットマップ画像の EPS 画像への変換から始めましょう.

ビットマップ画像の PS 化

Netpbm

Netpbm は非常に多くの種類の画像形式を相互に変換するためのツール群の総称です．`xxxtoyyy` という名前のコマンドは形式を xxx から yyy に変換します．PS，EPS に関しては

```
pbmtolps pbmtoepsi pnmtops psidtopgm pstopnm
```

があります．pbm(portable bitmap) はモノクロ画像形式ですが，pnm(portable aNymap) とはそれ自身が形式を表している訳ではなくて，pbm，pgm(グレイスケール)，ppm (カラー) のいずれの形式も扱うことが出来る意味です．epsi は大きさの情報と生のモノクロビットマップデータが内包された EPS ファイルです．pbmtoepsi で作成された epsi ファイルはビットマップを表示する PS コマンドが含まれないので，ファイルサイズは小さいのですが gv では枠のみで内容は表示されません．**ps2epsi** や ImageMagick の **convert** で作成された epsi ファイルは gv でも内容が確認できます.

さて，Tgif では EPS を挿入した場合にプレビュー用ビットマップがないと枠だけで内容が表示されませんので，図に細工を加えるという作業が難しくなります．したがって，epsi を使うようにしたいのですが，epsi のプレビュー用のビットマップがそのまま使われるようなので注意が必要です．つまり，はじめからビットマップだけならば拡大縮小すれば品質が劣化するのは当然なのですが，ベクター図形本体＋プレビュー用ビットマップという構成の epsi でもビットマップの部分が本体に替わって使われるので，拡大縮小すると品質が著しく劣化してしまうのです．図 1は，Tgif に *.eps と *.epsi で取り込んだ同じ図面を 4 倍に拡大して EPS で印刷した場合の比較です．epsi で取り込んだ右側の図は (Tgif で編集中に内容が表示されますが)，プレビュービットマップなしの eps で取り込んだ左側の図に比較してあきらかに品質が劣ります．まあ，作業するファイルは仮想にして，Tgif 編集中はビットマップ付きの epsi から，EPS に変換する場合にはベクター情報だけの eps からリンクをはるようにすれば両者の長所を活かせるのですが，面倒ですね.

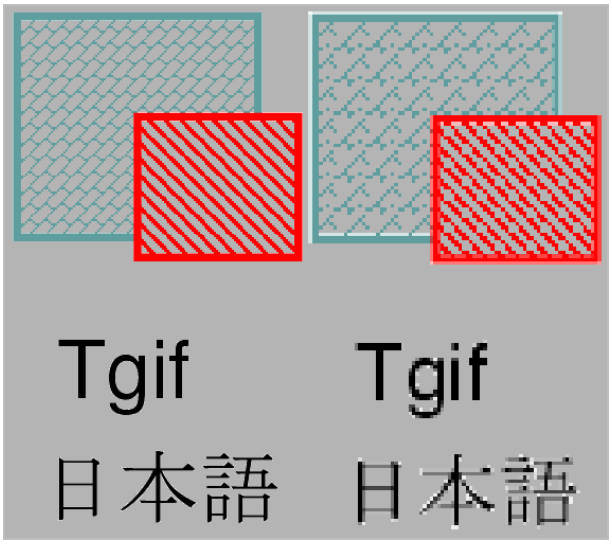

図 1 普通の EPS ファイル (拡張子 .eps) と，プレビュー用のビットマップ付きの EPS ファイル (拡張子 . epsi) を Tgif に取り込み，印刷した場合の比較

• ppmpat と ppmforge

Netpbm には画像形式の変換以外にも面白いツールが含まれています．例えば，ppmforge や ppmpat が生成する画像はスクリーンの背景として利用できます．ppmpat の生成する画像は模様ですが，特に格子模様 (`-g2`，`-g3`，`-madras`，`-tartan`) は継目が判らないものなのでタイル表示に適しています．ImageMagick の display で全画面表示させるには “`-window root`” とオプション指定します.

```
ppmpat -madras 64 64 |display [-window root] -
```

ppmforge はフラクタル図形を使った景観画像を作成します (図 2)．雲だけを作成するには “`-cloud`” を指定します．xv で全画面表示させるオプションは “`-root`” です.

```
ppmforge -tilt 30 -hour 10 |xv [-root] -
```

寄り道ついでに，この手のツールの老舗 **bggen** も使ってみましょう．段々と変わる色の帯 (PPM です) を作成するといったらいいでしょうか，傾きも付けられるし，なかなかのものです．例えば

```
bggen -g 96x96 -r 30 yellow green4
```

などは，若葉の頃にピタリと思います．なお，Tgif では，編集→画像処理→背景生成 (Generate) →RunBggen でこの bggen を呼び出してます．

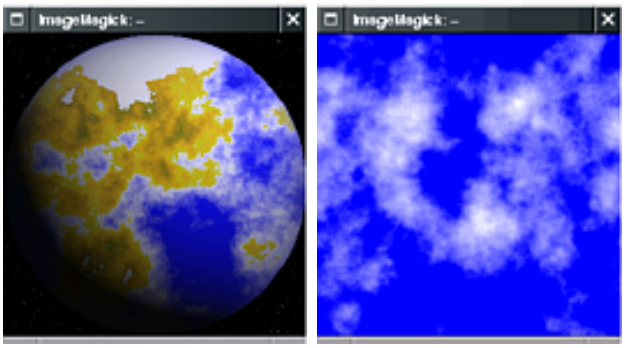

図 2　ppmforge で景観画像を作成．左：宇宙空間に惑星，右：空に雲

ImageMagick の convert

ビットマップに限定すれば，何も EPS に拘る必要はありません．実際，Tgif では画像を XPM 形式で取り込みます．あらかじめ XPM に変換しておかなくても，Tgif にファイル挿入する時点で XPM へと変換する filter を定義できます．例えば，ImageMagick の convert を用いて，汎用の filter を次のように定義してみましょう．

```
Tgif.ImportFilter0: \n\
  Any eps;ps;xwd;pbm;pgm;ppm;jpg;gif;png \n\
  convert -colors 222 %s xpm:-
```

画像形式を表す拡張子 (Tgif はこの拡張子がつくファイルを探して一覧表示します) を追加すれば，どんな形式のビットマップ画像 (convert が support しているもの) も変換して取り込めるようになります．このように，convert はオプションが豊富で汎用性の高い画像変換ツールです．筆者が重宝しているオプションに，“`-border geometry`” があります．しばしば説明のためにスクリーンダンプをするのですが，背景が白のウィンドウの一部を取り込んで誌面にそのまま載せると，何か頼りないので (図 3 左上)，枠で囲みたくなるのです．そのような場合には

```
convert -border 2x2 ***.eps ***new.eps
```

などとして，グレイの境界を付けます (図 3右上)．“`-bordercolor 境界色`” を `-border` よりも前に指定すれば，好きな境界色を使うことができます．また，画像

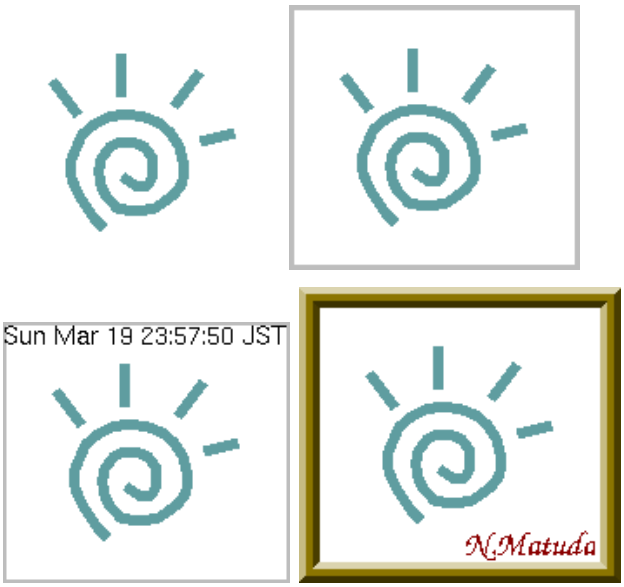

図 3　convert の活用例．左上：元図，右上：境界を付加，左下：更に日付けを書き入れ，右下：フレームで囲い，サインを書き入れ．

に細工するには普通ビューアで確認しながら行うのですが，日付けや名前を書き込むなどといった決まりきった作業は，バッチ処理的に済ませたいものです．convert はそのような定型作業の処理に向いていて，図 3右下のように，フレームで囲ってサインを書き込むという細工を，次のようにコマンドライン上で実行して，画面を開かずにオプションで指示できます．

```
convert -mattecolor gold4 -frame 12x12
  -font -gs-zapfchancery-*-*-*--20-*-*-*-*-*-*-*
  -pen red4 -draw "text 95,138 N.Matuda"
  pika.eps pikan.eps
```

“`-draw strings`” が書き込みの指示で，strings にはオブジェクト (text，rectangle, image 他) と位置を指定します．詳細はマニュアルで確認してください．この例では，Ghostscript で標準配布のスクリプト文字 `zapfchancery` を使ってサインの雰囲気を出しました．

文字列の PS 化

a2ps, psconv

less などで読める平テキスト文を印刷して保存するには，英語ならば迷わず直接 lpr に渡せばいいのですが，日本語を含んでいる場合は，日本語対応のプリンタフィルターがインストールされていなければ文字化けします．そこで，もし GS で印刷が可能になっているなら，一端 PS ファイルにしてから印刷する方が，整

形もできますし，設定に悩まずにすむので良いかもしれません．もちろん直接印刷よりもずーっと処理に時間がかかりますが，PentiumII が主流の現在では，数枚ならばイライラせずに待てる程度となってます．

Perl スクリプト a2ps は，日本語を含む平テキストを PS ファイルに変換してくれます．枠やファイル名を付けて，あるいは A4 用紙に 2 ページをまとめてくれたりと，大変重宝します．使い方は簡単で

```
a2ps [switches] [files]
```

とします．switches の詳細は -help オプションで見ることができます．良く使うオプションは用紙 1 枚に 1 ページ分しか収めない (紙浪費型？) -p(portrait の略) です．

```
a2ps テキスト |gslpr -
```

などとパイプ接続して PS ファイルの印刷コマンドに渡したりするのが普通でしょう．意味ないですが，次のように端末から直接読み込ませて，gv に表示なんてこともできます (^^;

```
a2ps |gv -
いろはにほへと....(→文字列の入力)
Ctrl-D (→入力終了)
```

psconv は **cmt，prn，cprn，kcc**(それぞれに関してはマニュアルをどうぞ) と供に JE の頃からの懐かしい印刷ユーティリティです．レイアウトの細かい指定が可能です．例えば -V2 は，80 桁 132 行 2 段 (4 ページ分に相当) という縮小印刷を可能にするオプションです．

PS のユティリティ

作成された PS ファイルに細工を施す便利なユティリティを幾つか紹介します．

psnup

複数ページを 1 枚に詰め込んでまとめるシェルスクリプトで Roy Smith 氏 が書いたものです．“-n value” には 2 の冪乗が選べますが，16 では小さすぎるでしょう．“-s” は最初のページを分割された用紙のどの部分から始めるかを指定します．“-r” は用紙内のページの並びを逆にします．

```
psnup [-n 2|4|8|16] [-r] [-s #] [file]
```

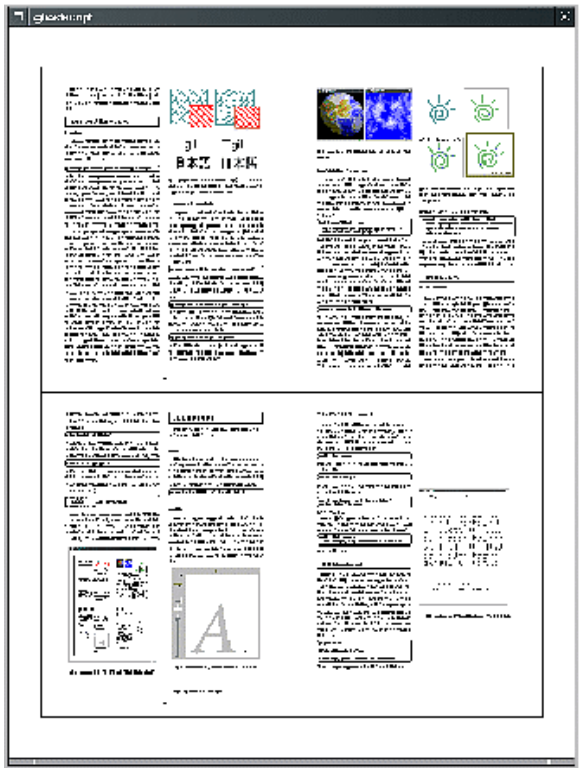

図 4　psnup で 4 ページを A4 用紙 1 枚に入れた例

PSUtils [1] W[3]

PSUtils は，Agnus Duggan 氏がそれまで公表されたものをモデルとして C 言語で書き起こしたユティリティ群の総称です．psnup があったりして (オプションが異なっている) 混乱します．指定のページを抜き出す **psselect** は大変重宝します．スクリプトファイルもが含まれていて，例えば PS で使われるフォントの字形を大写しに表示させる showchar などは面白いものだと思います (図 5).

マニュアルの PS 化： man –t

マニュアルを紙に印刷することは滅多にないでしょうが (Online の趣旨からはずれますからね)，PS ファイルで印刷できることを知っていて損はないでしょう．単に man ならば表示デバイスは Console や X11 ですが，オプション -t を指定すると

```
groff -Tps -mandoc
```

を使って，PS ファイル化し標準出力に渡すことになってます．従って，

```
man -t kon > kon.ps
```

として，kon のマニュアルの PS ファイルが出来ます．

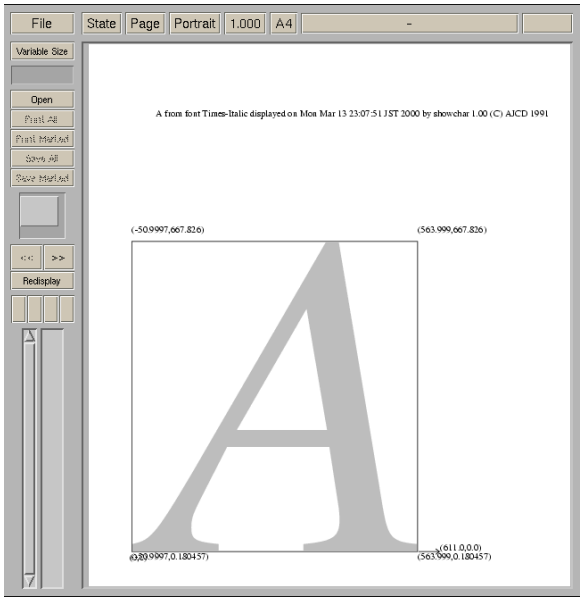

図 5　showchar で Times-Italic の A を表示

もう少し体裁を整えるなら

```
man -t kon |psnup -n 2 >kon.ps あるいは
man kon |a2ps > kon.ps
```

もいいでしょう.

man の裏では groff が動いていることが判ったついでに，TEX の DVI ファイルも作ってみましょう．groff のマニュアルを読むとデバイスを dvi にすれば良いことが判るので,

```
groff -Tdvi -mandoc
  /usr/man/ja_JP.ujis/man1/kon.1 > kon.dvi
```

のようなに実行してみましょう.

フォントをみるツール

xfontsel，xfd で字形を確認することのできるフォントは X11 で使用可能な状態 (設定済み) にあるものです．フォントを追加する場合，持ち込んだフォントを直接見るツールがあると便利です.

FreeType のデモ： ftview, ftstring

TrueType については，freetype のデモプログラムに ftview や ftstring が含まれています．フォント名を引数にして，次のように実行します.

```
ftview [-r R] [-g] ppem fontname[.ttf|.ttc]
```

“ppem” を忘れないでください．“-r R” は解像度の指定で数字が大きいと字も大きくなります．“-g” はグレイスケールのレンダリングを行うオプションで，多少滑らかな表示が得られます．日本語 TrueType では最初に英数文字部分が表示されますが，“l(小文字の L)” と “k” で 1 ずつ，あるいは “i” と “o” で 10 ずつ，表示先頭文字番号 (タイトルバーの Glyph 値) を移動させることができます．そこで，“o” を連打すると 540 でひらがな，930 で漢字が見えます (図 6)．その他にもキーバインドがあって，“u” と “j” で解像度が増減し，“q” で終了します.

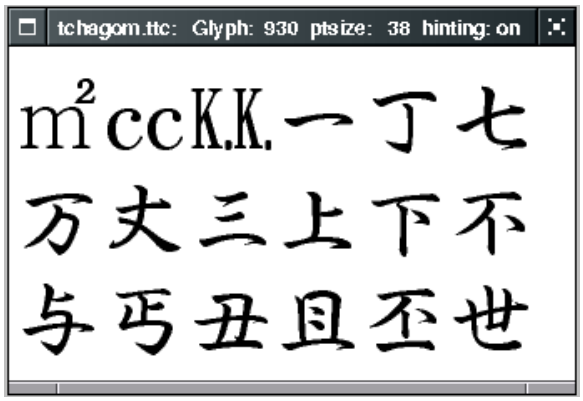

図 6　ftview で日本語 TruType フォントを表示

T1library のサンプル：xglyph

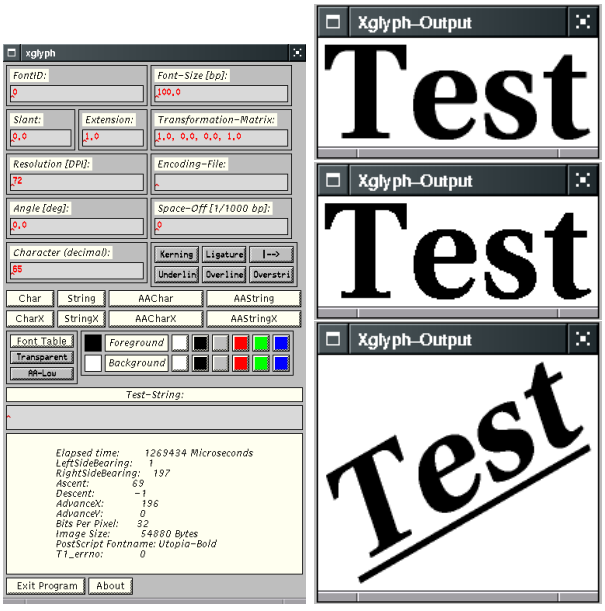

図 7　Type1 フォントビューア xglyph で表示させた Utopia-Bold フォント：アンチエイリアスあり (右上)，アンチエイリアスなし (右中)，傾斜 0.2，下線付き，回転 30 度を施した (右下).

Type1 フォントの字形を確認するには，Type1 フォントからビットマップを生成するためのライブラリ T1Library [2]W[3] に含まれる xglyph が良いでしょう．フォントの傾斜，回転，変形，線付けなどを試してみることができます．アンチエイリアス (AAChar，AAString ボタン) にも対応しています．

```
xglyph [options] Type1 フォント名
```

と起動します．オプション名は `--help` か `--Help`(詳細) で見ることができます．ftview と異なり，フォント名は拡張子まで含めた名前を入力する必要があります．中央にある `String` ボタンを押すと `Test-String:` の入力部に打ち込まれた文字列が表示されます． 打ち込みがない場合には，デフォルトで “Test” が表示されます．また，下の窓にはフォント展開に関する色々な情報が表示されます (図 7).

GS に含まれるスクリプト

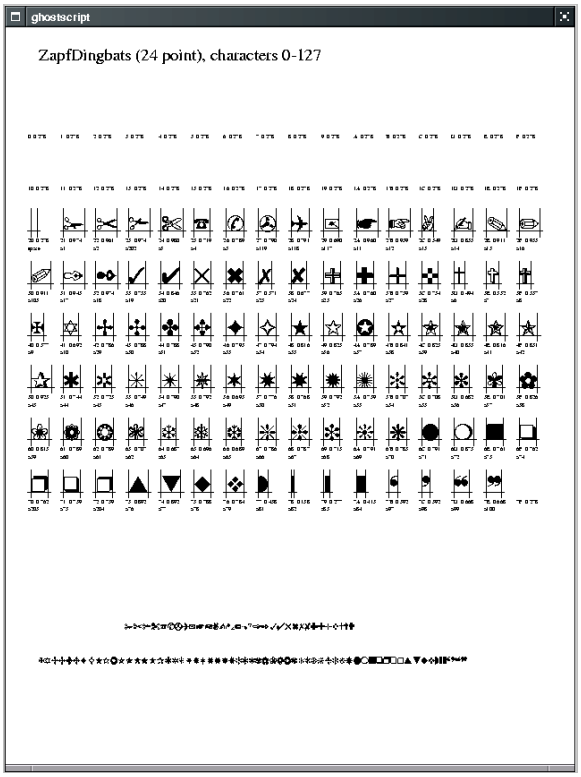

図 8　prfont.ps で ZapfDingbats フォントを表示

GS ディレクトリには沢山のファイルがあります．筆者は今まで，設定ファイルの kconfig.ps やサンプルファイルの tiger.ps，article9.ps くらいしかさわったことが無かったのですが，フォントを表示させるスクリプトがありますから試してみましょう．まず PS フォントのカタログ (番号・グリフ・外形枠) を表示する prfont.ps です．prfont.ps を読み込ませて gs を対話的に実行します．プロンプト “`GS>`” に対して，“フォント名 DoFont” と命令すると，第 1 ページを表示します．Enter キーで表示ページが順繰りに進みます．“quit” を入力すればで終了します．

```
gs prfont.ps
...
GS>/ZapfDingbats DoFont
...
>>showpage, press <return> to continue<<
```

図 8は，ZapfDingbats 記号を表示させた結果です．また，`kanji` サブディレクトリにある `allkanji.ps`，`hankaku.ps` は是非一度試してみる価値があると思います．脱線ついでに，`view***.ps` という名前の，ビットマップ画像を見るためのスクリプトもあって，遊べます．

次回は

新人が巷に溢れる季節，ネットワークというと Netscape しか思いつかない学生に対して，毎年紹介するいくつかの基本的なネットワーク関連のコマンド telnet，rlogin，ftp 等を総合的におさらいしてみようと思います．今更の感なのですが，基礎重視ということで...

参考文献

[1] PSUtils の新しいサイト．W[3]
`http://www.tardis.ed.ac.uk/~ajcd/psutils/index.html`

[2] 作者 Rainer Menzner さんの「A Type 1 Rasterizer Library for UNIX/X11」ページ．W[3]
`http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/rmz/t1lib/t1lib.html`