fonts and scripting

Installing PostScript Fonts Under Unix/Linux

Roland Kwee
email:
rolandkwee@acm.org

abstract
This article discusses practical issues for installing Adobe PostScript fonts for TEX running on Unix/Linux. It also presents a script for automating the installation in order to install a package with over 200 fonts. As a useful side effect, the script creates a font data base and a font catalog to help the buyer/user of the font collection.

keywords
fonts, PostScript, linux, Adobe, dvips, ghostscript.

Introduction

At the recent NTG meeting in Leuven, Belgium, I got enthousiastic about fonts, and even bought my first book on typography. After years of using TEX and Latex, I had become familiar with many text processing and formatting issues, and most of you will agree that this is a big field. But after reading the typography book I realized that text formatting is only one part of the job, and that using fonts in an effective manner is the other half. This article addresses people who are convinced that using fonts is a very important issue when creating documents. If you are not convinced of this, I recommend to study one of the introductory books on typography before going into the remainder of this article, because using fonts in the "wrong" way would probably not improve your documents.

In order to use fonts, one first needs to obtain some. The computer modern fonts and the 35 standard PostScript fonts are just not enough for ambitious use of fonts. At the NTG meeting I thought differently, but after reading on typography I know better. I happened to get the Adobe Type–On–Call CDROM from a colleague who once purchased some Adobe fonts. It seems that font vendors distribute CDROMs at no or little charge, because they charge you later for the key codes to unlock the fonts that you want to buy, usually about f 60 per type face (a font usually contains several type faces, e.g., roman, italic, bold). However, upon registering the Adobe Type–On–Call CDROM, you get to choose two free fonts, and in my case, Adobe threw in another free font. The latter was package 950–00, containing over two hundred fonts.

In order to use the obtained fonts, one needs to install those. The vendor usually provides you with an install program for DOS/Windows, or Mac, but not for Linux. Font installation, whether PK or PostScript format, is never simple. Installing over two hundred fonts needs to be automated. This article presents an overview of the issues around PostScript font installation, and a way to automate this.

In order to use the installed fonts, one really needs a font catalog, showing samples of each installed font, and linking those to the font codes used in TEX documents. The scripts presented in this article therefore also create a font catalog as part of the installation procedure. The catalog is also a test of the installed fonts.

This article assumes some familiarity with dvips and ghostscript/ghostview, because those are the programs I use to print TEX documents. The reader is also assumed to understand common shell commands, or to be able to look these up in the man pages.

Font basics

First I will give you an overview of the installation procedure of Adobe fonts. In the following section I'll describe ways to automate the installation.

The description is for Type 1 Postscript fonts. To be honest, I haven't taken the trouble yet to find out about Type 2 fonts.

Font installation under windows The installation of a purchased font must begin under windows, because only then is it possible to use the key code faxed to you by Adobe to unlock a font.

From the CDROM, the following files are supplied for a font:

- `font____.pfb` (postscript font binary) contains the characters, this file must be unlocked with a key to be ordered from Adobe.
- `font____.afm` (adobe font metrics) contains metrics. It is normally not installed under Windows. To get this file, check the appropriate box in the Adobe Installer. It can also be copied from the CDROM, but note that the *hidden* attribute is set.
- `font____.pfm` (printer font metrics) contains metrics for windows, this file is extracted from the afm file during installation.
- `font____.inf` is for windows, this file is also extracted from the afm file during installation.

The Adobe file names are eight plus three characters, and underscores are added for font names shorter than eight characters.

After you installed the fonts under windows, you can immediately start using those, e.g., with notepad. By browsing the list of fonts from the notepad menu, you'll not be able to see whether a name relates to a Windows standard font, or a font purchased from Adobe. Because a font is really one typeface, some fonts must be selected with the bold and/or italic buttons. Under windows, you'll soon miss a home made font data base and font catalog.

If you install more than one font package, you should create a directory for each package and specify those in the Adobe Installer. After all, if you obtained the fonts legally, you want to be able to keep track of the origin of each and every font file in an easy way.

NFSS font name scheme Postscript fonts are sold by various suppliers: Adobe, Bitstream, Linotype, etc. Some fonts have more than one supplier, and those may be identical fonts or variants. This gave rise to all kinds of file name conflicts and other chaos. Now we use the NFSS: New Font Selection Scheme. In this scheme, many fonts of most suppliers have gotten new file names, see the file adobe.map by Karl Berry, available on CTAN. For fonts you may have obtained not listed in that file, you should choose file names that have not yet been assigned.

The NFSS font names include codes for the supplier, encoding, weight, etc. For details, refer to *The Latex Graphics Companion* or other excellent books.

All this helps when you want to see the correct fonts used for your TEX files when processed at other sites.

NFSS encoding The pfb font file defines a set of characters (glyphs), identified by names, such as "A", "A-acute" (Á), "A-ring" (Å), etc. The pfb font file also defines a mapping of glyph names to numbers. In a PostScript document all characters are referenced by these numbers.

In practice, this mapping often does not include all available characters, and sometimes the ordering of the mapping is not convenient. Therefore NFSS has defined some new

encodings, suitable for TEX usage. The program *afm2tfm* has a feature to rearrange the encoding.

Font installation for TeX on Linux To install a postscript font for TEX, several steps are needed. It is worthwhile to try to install a single font file *by hand* just to get familiar with the issues involved.

- Install the font under windows, as described.
- Determine the adobe file name (with the underscores), the official font name (see the FontName header line in the afm file), the desired TEX font name (e.g.: the NFSS FontName naming scheme), and the best encoding: 8r for text fonts, 7a for symbol fonts.
- TeX needs a tfm file (tex font metrics), which can be obtained from the afm file with the program afm2tfm which is commonly installed with TEXThe typical location for the file is texmf/fonts/tfm/adobe/..../font.tfm. Don't forget to run MakeTeXls–R.
- dvips needs a new entry in its psfonts.map file to know that the font is a postscript font, and to translate between the font name in the TEX system and its postscript name. Typical location is texmf/dvips/psfonts.map.
- The printer, in my case the ghostscript interpreter, needs the pfb file. Ghostscript needs an entry in the file Fontmap to associate the pfb file with the postscript font name that dvips writes in the postscript output. The typical directory for file Fontmap is `/usr/lib/ghostscript-4.03/fonts/`.
- In your TEX document, use the commands `\font\f=pabc at 10pt \f` to typeset text in the new font, assuming that you installed the new font under the name pabc.

Note that ghostscript does not complain if it cannot find a font. It just substitutes another font without warnings. If you are not sure which file is used for a particular font, just use ghostscript without the user–friendly shell: `$ gs -sDEVICE=x11 file.ps` This will show some information. Leave the gs shell with `exit` .

Font installation procedure

To install a large number of files, a script can be written that performs all necessary steps on all font files. However, it turned out that a single script will be complex. Many fonts have something special, e.g., about the file name or encoding, and the handling of special cases makes a script extra complicated. A complicated script takes more effort to debug.

A better approach seemed to split the installation in several steps. A first script builts a kind of data base text file with a line for each font and columns for each attribute. Then, after verifying that all attributes are correct, a second script installs the files. Both scripts can be quite simple and straightforward, with a high degree of trust that if one font is installed correctly, all fonts will be installed correctly.

The font data base text file is not only used as an intermediate step for installation, but also after installation as an aid in using the fonts. A third script produces a TEX file that prints samples of each font, i.e., a font catalog.

Creating the font data base

To keep things organized, I asked the Adobe font install program to install package 950–00 in directory `C:\PSFONTS\950-00\` . This is where the pfb files are installed. The afm files

are installed in directory `C:\PSFONTS\950-00\AFM\` . On my computer, these directories are accessible under linux:

```
$ ls /dosc/psfonts/950-00/*pfb | wc
```

$\Longrightarrow$ 227 font files

To obtain the Fontname font names from the file adobe.map with grep, this file must be sorted on the second field. Otherwise, the first line of a `grep` on Garamond-BoldCondensed will give the entry for Garamond-BoldCondensedItalic instead. To sort the file, remove the comments, run this shell command:

```
$ sort -b -k 2 adobe.map > adobe.map.sorted
```

and reinstall the comments.

The following script reads all file names, processes those to extract the base file name etc, and to look up the standard font name and encoding. I present the file with all comment lines, in the hope that these help in understanding the script. I haven't looked very hard for a web version for shell scripts, but if I had it available, I'd written this entire article in the script.

```
#!/bin/sh
# fontdb1.rc     Roland Kwee    30 Dec 1998
# Purpose: build a font data base text file for installing PS fonts.
# Needed files: adobe.map must be in the current dir.
# Argument(s): $1 = package number, e.g., 950-00
# Usage: chmod +x fontdb1.rc;
#        rm -f fontdb1.txt; ./fontdb1.rc 950-00 > fontdb1.txt

package=$1 # Adobe package code, e.g., 950-00

# This function will be run on each pfb file name.
# Argument(s): $1 = pfb file name with or without path.
func(){
  # Make several versions of the file name.
  fname=`basename $1` # remove path name
  fontname=`expr $fname : "\([^_]+\)_*\.pfb"` # e.g., ovbk
  fontn___=`expr $fname : "\([^\.]+\).pfb"`   # e.g., ovbk____
  # Find official font name from the afm file.
  offname1=`grep FontName $sourceafm/$fontn___.afm | tr -d '\015'`
  offname=`expr "$offname1" : "FontName\ \(.+\)$"`
  # Find name in FontName scheme.
  fnname1=`grep $offname adobe.map`
  fnname=`expr "$fnname1" : "\(^.+\)\ .*$"`
  # Not all fonts will be in adobe.map, so mark fonts not found.
  # But if it is found, extract the encoding.
  if [ -z "$fnname" ]
  then
    fnname="!!!!!!!!"
    encoding=""
  else
    # Note that 0 and 2 do appear in font abbreviations,
    # but not in encodings.
    encoding=`expr "$fnname" : "[a-z0-3]+\([4-9][a-z]\).*$"`
  fi
  # Some fonts require no reencoding by afm2tfm.
  if [ -z "$encoding" ]
  then
```

```
    encoding="xx"
  fi
  # Output all attributes thus found in a data base text file.
  # Each line starts with a function name to enable further processing.
  # Use tabs to separate variable-width columns in a nice way.
  echo -e \
   "func  $package  $fontn___  $fontname\t$fnname\t$encoding $offname"
}

# e.g., Officina Book font of package 950-00 is in files:
#   $sourcedos/950-00/ovbk____.pfb, $sourcedos/950-00/afm/ovbk____.afm
sourcedos=/dosc/psfonts
sourceafm=$sourcedos/$package/afm

# Run the above function on each file in the package directory.
for f in $sourcedos/$package/*.pfb
do
  func $f
done
```

After running this script, and saving the output in the font data base text file fontdb1.txt, you have to inspect the data base file to resolve fonts that could not be found in adobe.map. I did that by extending adobe.map with a new file adobe-ext.map as shown below, and running the above script again.

Find all missing names:

```
$ grep \!  fontdb1.txt > adobe-ext.map
```

Next, type new FontName names in file adobe-ext.map. For compatibility, do not re-use abbreviations that are already defined for other fonts, e.g., ov for Octavian. In other cases, the Adobe file name is a good starting point for abbreviations. All names begin with p (supplier code for Adobe). All text file names end with 8r (TEXbase1 encoding), all other fonts (i.e., symbol fonts) with 7a (alternate characters only). For Adobe package 950-00, my file `adobe-ext.map` got entries on the basis of the following new 2-character Fontname abbreviations (Adobe file names in parentheses):

- ir Isadora (ir)
- of OfficinaSerif (ov)
- ow OfficinaSans (ow)
- th Tiepolo (tp)
- rf Rosewood-Fill (uxfi)
- rr Rosewood-Regular (uxrg)
- wm Myriad-Sketch (wmske)
- wq Quake (wq)
- wy Mythos (wy)

After rerunning the script, you must again check very carefully the resulting data base. Especially the encodings are prone to errors, e.g., if the font name has digits. Correct those errors and store the file as fontdb2.txt.

ZapfDingbats appears without encoding, which is now indicated with xx. The program afm2tfm should then not use any encoding.

Check for uniqueness of the font names (5th column) as found in adobe.map, by running the following shell command:

```
$ sort -b +4 fontdb1.txt | uniq -d -4 -w 9
```

which will print only duplicate entries.

Note that it is much more fun to correct entries in the font data base text file than to correct fonts that are installed incorrectly. Programmers and engineers would say that it is cheaper to correct errors earlier in the development process.

Now we add the following shell commands to install the various files. I really had one file containing the script code for all stages of the installation, but here I show only the code relevant for the installation of the files according to the font data base. The preamble and postamble stuff is something that is not really used in this stage, but is meant for producing the font catalog later on.

```
#!/bin/sh
# install.rc            Roland Kwee     30 Dec 1998
# Shell commands to install font files in TeX
# and to create font maps for dvips and ghostscript.
# Usage: create a directory for the new package in the ghostscript dir
#        and in the tfm dir tree, then run this script.

# Stop shell script upon an error.
set -e

func_install(){
  package=$1
  adobename=$2
  fontname=$3
  fnname=$4 # name according to Fontname scheme
  encoding=$5
  offname=$6 # official postscript name
  echo Installing: $offname
  # Create and install tfm file and add entry to map file.
  afmdir=/dosc/psfonts/$package/afm
  tfmdir=/bigdisk1/usr/local/share/texmf/fonts/tfm/adobe/$package
  if [ ! -d $tfmdir ]
  then
    echo Need to do: mkdir $tfmdir
    exit 1
  fi
  dvipsdir=/bigdisk1/usr/local/share/texmf/dvips/base
  # Note: there is no encoding file 7a.enc in dvips, so ...
  if [ "$encoding" = "8r" ]
  then # with encoding
    (cd $dvipsdir; afm2tfm $afmdir/$adobename.afm -T $encoding.enc \
                   $tfmdir/$fnname.tfm >> $package.map)
  else # without encoding
    (cd $dvipsdir; afm2tfm $afmdir/$adobename.afm \
                   $tfmdir/$fnname.tfm >> $package.map)
  fi
  # Copy afm and pfb files to the ghostscript directory.
  # Create a map file for ghostscript.
  pfbdir=/dosc/psfonts/$package
  gsdir=/usr/lib/ghostscript-4.03/fonts
```

```
  if [ ! -d $gsdir/$package ]
  then
    echo Need to do: mkdir $gsdir/$package
    exit 1
  fi
  cp $pfbdir/$adobename.pfb $gsdir/$package/$fontname.pfb
  cp $afmdir/$adobename.afm $gsdir/$package/$fontname.afm
  echo -e "/$offname \t($package/$fontname.pfb) \t;" \
    >> $gsdir/$package/Fontmap
}
# Make it easy to switch between scripts to run on the font data base.
func_preamble(){
}
func(){
  func_install $*
  #func_catalog $*
}
func_postamble(){
}
func_preamble
# Now follow the font data base entries, like:
#func  950-00  agd_____  agd     pagd8r      8r AvantGarde-Demi
#func  950-00  agdo____  agdo    pagdo8r     8r AvantGarde-DemiOblique
# etc.
func_postamble
```

Save the new file with the above script and the font data base entries from fontdb.txt under the name install.rc, and run it as superuser, but don't forget to create the package directories for tfm and pfb files.

If, for some reason, you need to rerun the script, make sure to first remove the font maps generated with the old run.

As a final step, paste the new font map files for dvips and ghostscript to the original ones. Now the files should be installed and ready to use.

Common installation problems

A problem can arise from the default dvips font map file psfonts.map. Many more than the 35 standard postscript fonts are listed. Some may overlap with the added fonts, but specify a font file that is not available. This may lead to most new fonts to become unavailable, perhaps by triggering bugs in dvips. A rigorous cure is to delete all superfluous font names from psfonts.map, before adding the new font names. It was a nice idea to include many fashionable fonts in the psfonts.map file, but in my opinion, the extra lines are just garbage. It is just as easy to update psfonts.map each time you obtain and install new fonts.

A similar problem may arise with the ghostscript Fontmap file. E.g., ZapfDingbats is a standard font and is standard installed. When installing Adobe package 950-00 in a separate font directory, a second ZapfDingbats file is installed. These should be identical, but one of the fonts may never be used. If both font files must be available, it may be necessary to do some renaming in Fontmap.

Font Catalog

The following script generates TeX code for a font catalog. It includes the calls to include a file with TEX macros, and to end a TEX session.

```
#!/bin/sh

# Function to create an entry in a TeX font catalog.
func_catalog(){
  package=$1
  adobename=$2
  fontname=$3
  fnname=$4 # name according to Fontname scheme
  encoding=$5
  offname=$6 # official postscript name
  echo \bs entry{$offname}{$fnname}{$package}{$fontname}
}
# Make it easy to switch between scripts to run on the font data base.
func_preamble(){
  echo %this file is generated by script fontdb.rc.
  echo \bs input catalog.mac
}
func(){
  #func_install $*
  func_catalog $*
}
func_postamble(){
  echo \bs bye
}
func_preamble
# Now follow the font data base entries, like:
#func  950-00  agd_____  agd     pagd8r      8r AvantGarde-Demi
#func  950-00  agdo____  agdo    pagdo8r     8r AvantGarde-DemiOblique
# etc.
func_postamble
```

The following file contains TeX code to typeset the font catalog. Of course, as a dedicated TEX user, you will want to improve this to suit yourself.

```
% catalog.mac     Roland Kwee    1 Jan 1999
% TeX macros for fontdb.rc.

\font\c=ptmr at 10 pt \c

\def\entry#1#2#3#4{%#1=official ps name, #2=tex font name,
                   %#3=adobe package code, #4=adobe file name
  {\font\f=#2 at 20pt \f #1} #2, Adobe #3 #4\par\medskip
}

\def\entrynontext#1#2#3#4{%#1=official ps name, #2=tex font name,
                          %#3=adobe package code, #4=adobe file name
  {\font\f=ptmrb at 10pt \f #1} (non-text) #2, Adobe #3 #4\par\medskip
}
```

Create the tex file, after setting func in fontdb.rc:

```
$ ./fontdb.rc > catalog.tex
```

Typeset the font catalog, it cannot be easier:

```
$ tex catalog.tex
```

Ghostscript must load the font file for each font used in the document, and may choke on a postscript file with two hundred fonts. A remedy is to split the postscript file in a number of smaller files, e.g., one file per page. The dvips command to accomplish this is:

```
$ dvips -i -S 1 catalog.dvi
```

This results in a number of postscript files catalog.001, catalog.002, etc, which may be printed much easier.

The font catalog gives an impression of each font, but this is not sufficient for the symbol fonts. For symbol fonts we need to print a table showing each glyph and the char codes to be used in your TEX documents. The easiest way to print the table for a font is to use the interactive latex file nfssfont:

```
$ latex nfssfont
```

Conclusion

Installing fonts doesn't have to be complicated when you know how to do it. Some scripts were shown to aid the installation. Some tips were mentioned that hint at hours of despair by the author, and he hopes that you will be able to avoid that experience.

Getting free fonts, buying more fonts, and installing those, however, is only the preparation for using your fonts. Now is the time to add a book on typography to your TEX bookshelf, and to create even more beautiful books.

CaslonTwoTwentyFour-Book pc2k8r, Adobe 950-00 c2w

CaslonTwoTwentyFour-BookIt pc2ki8r, Adobe 950-00 c2wi

CaslonOpenFace pcarl8r, Adobe 950-00 ca

CooperBlack pcbc8r, Adobe 950-00 cb

CooperBlack-Italic pcbci8r, Adobe 950-00 cbi

Cheltenham-Bold pctb8r, Adobe 950-00 chb

Cheltenham-BoldItalic pctbi8r, Adobe 950-00 chbi

Usherwood-Book puwk8r, Adobe 950-00 usw

Usherwood-BookItalic puwki8r, Adobe 950-00 uswi

ROSEWOOD-FILL prf8r, Adobe 950-00 uxfi

ROSEWOOD-REGULAR prr8r, Adobe 950-00 uxrg

Veljovic-Bold pvjb8r, Adobe 950-00 vlb

Veljovic-BoldItalic pvjbi8r, Adobe 950-00 vlbi

Veljovic-Black pvjc8r, Adobe 950-00 vlbl