Font installation the shallow way[*]

SIEP KROONENBERG
Rijksuniversiteit Groningen
Department of Economics
P.O. Box 800
9700 AV Groningen, The Netherlands
`siepo (at) cybercomm dot nl`

Abstract

For one-off projects, you can cut corners with font installation and end up with a more manageable set of files and a cleaner TEX installation. This article shows how and why.

Keywords

Font installation, afm2pl, afm2tfm, TrueType, pdftex, mapfiles

If you are putting together a flyer or invitation or book cover, then it would be nice if you could test a batch of fonts from your CorelDRAW or Illustrator CD, or your Windows font directory, without too much trouble and without polluting your TEX installation with a lot of stuff you are never going to use again.

This article takes you through the steps needed to use one or more fonts in one particular document. We won't really install the fonts; we just generate the files that TEX needs and leave them where TEX will find them, *i.e.* in the working directory. This makes it easy to take the project to another system, and easy to clean things up.

We will primarily use afm2pl to generate `.tfm` (TEX Font Metric) files. Later on, we show the steps required for afm2tfm. Both programs are simpler and much faster to use than the usual choice, fontinst. They create few intermediate or unnecessary files and do their job without virtual fonts. Virtual fonts and fontinst have their place, but sometimes there is no good reason to put up with the inevitable mess.

afm2tfm is available on all major free TEX implementations. afm2pl is part of current TEX Live distributions. Note that these programs are needed only to create the necessary font support files for TEX; once these files have been created, they can be used on any other system, whether or not it contains afm2pl or afm2tfm.

[*]This article appeared originally in slightly different form in MAPS 33, fall 2005.

An example

We use a decorative script font Pepita that Adobe bundles (or used to bundle) with some of its software.

pdftex needs the actual font file `epscr___.pfb`, its TEX font metrics file `epscr7t.tfm` and a mapfile containing an entry relating the two. First, we copy not only `epsrc___.pfb` but also `epsrc___.afm` to the working directory. We need the latter file to generate the `.tfm` file. Next, we enter the following commands on a command line:

```
afm2pl -p ot1 epscr___.afm epscr7t.pl
pltotf epscr7t
```

The extensions `.afm` and `.pl` are optional. The first command converts the `.afm` file to an (almost) human-readable text version of the desired `.tfm` file. The second command creates the more compact binary version.

Before we can use this font, we must tell LATEX about it. We do this with a font family definition file `ot1myfontfam.fd`:

```
\ProvidesFile{ot1myfontfam.fd}
\DeclareFontFamily{OT1}{myfontfam}{}
\DeclareFontShape{OT1}{myfontfam}{m}{n}{
  <-> epscr7t }{}
```

The prefix `ot1` indicates the encoding, which tells which characters occur at what positions. The next section will say more about encodings. The parameters to `\DeclareFontShape` are successively encoding, family name, weight (*e.g.* bold), shape, font file (without extension) and special options. You can normally leave this last parameter empty. With just one family member, we are not fussy about font characteristics and just pick defaults. We also leave this file in the working directory.

This is the code of our first testfile `exabasic.tex`, which uses this font:

```
\documentclass{article}
\pagestyle{empty}
\pdfmapfile{=epscr7t.map}

\newcommand{\fancyfont}%
  {\fontfamily{myfontfam}\selectfont}

\begin{document}
\fancyfont
Hello, world!

Accents: \'el\`eve bl\"of \"i;
Kerning: WAV, LTa
\end{document}
```

The `\pdfmapfile` command causes pdflatex to read the file `epscr7t.map`, which tells pdftex how to get the font into the output file. The prepended '=' tells pdftex that it should read `epscr7t.map` *in addition to*, not instead of, the default mapfile, and that in case of a conflict `epscr7t.map` wins.

Now we are ready to compile `exabasic.tex`:

```
pdflatex exabasic
```

This is the result:

Hello, world!
Accents: élève blöf ï; Kerning: WAV, LTa

Encodings

We already mentioned encodings briefly. Now it is time to dig a little deeper, because it is a topic that can easily trip you up.

An encoding defines what character corresponds to which number. Only numbers between `0` and 255 are allowed. A `.tfm` file associates character metrics directly with character positions and doesn't know what position represents what character. TEX simply makes assumptions about this correspondence or encoding, and if you disagree with those assumptions then you need to load some macro package or other to tell TEX otherwise.

We hope that mainstream TEX will eventually move to Unicode, which is a comprehensive encoding of all conceivable characters, including far-eastern alphabets and mathematical symbols. When that happens, we can forget about encodings and also do away with many applications of virtual fonts. There are already some Unicode-based variants of TEX.[2]

For a PostScript `.pfb` or `.pfa` font, character metrics are stored in a separate `.afm` file. These metrics are associated with characters, not with character positions. Therefore you should specify an encoding to afm2pl or afm2tfm.[3] The same encoding must also be specified in the mapfile entry. A PostScript font usually has more characters than fit into a single TEX encoding.

A command-line option such as '`-p texnansi`' or '`-p texnansi.enc`' means that the encoding should be read from a file `texnansi.enc`. This encoding probably has a different internal name.

OT1 encoding

If you don't tell TEX otherwise, it assumes that you are using the OT1 encoding. This encoding uses only 128 of the 256 available slots. TEX creates missing accented characters from an unaccented base character and a separate accent character. Unfortunately, this interferes with hyphenation. Apart from this, the OT1 encoding has various other oddities, and is best avoided. OT1-encoded fonts often have a TEX name ending in `7t`.[4] Note that `ot1.enc` comes with afm2pl and is probably not available if you don't have afm2pl on your system.

T1 encoding

T1 is the successor to OT1. It uses all available slots, and has lots of accented characters, including those for Eastern European languages. Because the T1 encoding left no room for symbols such as '‰' or '©' or '‡' you will need to get those from a second encoding of the same font. This second encoding is called TS1 or 'text companion'.

[2]Omega and its offshoot Aleph are Unicode-based. Users may also be interested in XƎTEX (`http://scripts.sil.org/xetex`), which is built on top of a regular TEX implementation and lets you use Unicode fonts installed on the system directly with TEX.

[3]If you don't specify an encoding, then you get the encoding from the `.afm` file, which is almost certainly not what you want.

[4]For afm2pl and afm2tfm, font names have no particular meaning. This is one more difference with fontinst. I add encoding suffixes such as `7t` and `8y` to font names just as reminders to myself.

For most traditional PostScript fonts, some of the accented characters in the T1 encoding aren't actually present and must be created with virtual font technology from a base character and an accent. Since it doesn't have to be done by TEX itself, this is no obstacle to hyphenation.

Although you can tell afm2pl to use T1 encoding, it can't create composite characters, and such composite characters will be missing unless they are already present in the original font.

T1-encoded fonts often have a TEX name ending in `8t`.

Texnansi encoding

The texnansi encoding, known as LY1 to LATEX, was introduced by Y&Y, the now-defunct company behind Y&YTEX, dviwindo and dvipsone. It combines a good selection of both accented letters and typographic symbols, and normally contains everything you need in a single encoding, at least for Western European languages. Texnansi-encoded fonts often have a name ending in `8y`.

The package texnansi selects the *texnansi* encoding and contains some additional code to smooth out incompatibilities with T1 and OT1.

A texnansi example

For this example, we choose Augie, a handwriting font from TEX Live. These are the commands for generating the `.tfm` and `.map` files:

```
afm2pl -p texnansi augie___.afm augie8y.pl
pltotf augie8y
```

This is `ly1augie.fd` (notice the `ly1` prefix):

```
\ProvidesFile{ly1augie.fd}
\DeclareFontFamily{LY1}{augie}{}
\DeclareFontShape{LY1}{augie}{m}{n}{
  <-> augie8y }{}
```

This is the LATEX code:

```
\documentclass{article}
\usepackage{texnansi}
\pagestyle{empty}
\pdfmapfile{=augie8y.map}

\newcommand{\fancyfont}%
  {\fontfamily{augie}\selectfont}

\begin{document}
\fancyfont
Hello, world!
Accents: \'el\`eve bl\"of \"i;
Symbols:
\textparagraph{} \textdaggerdbl{}
\texttrademark{} \textcopyright
\end{document}
```

And this is the result. Notice the extra symbols. These are absent from the T1 encoding and would have required a text companion font.

Hello, world!
Accents: élève blöf ï; Symbols: ¶ ‡ ™ ©

TrueType

Another scalable font format is TrueType, which is supported by pdftex but currently not by dvips. Font metrics are stored in the font file itself. Using TrueType is somewhat more work; the following commands are required to import a TrueType font such as Trebuchet:

```
ttf2afm trebuc.ttf >trebuc.afm
afm2pl -p texnansi trebuc trebuc8y
pltotf trebuc8y
<edit mapfile to replace .pfb with .ttf>
```

ttf2afm extracts the metric information from the `.ttf` file.[5]

afm2pl has no way of knowing that the `.afm` describes a TrueType font, and guesses that the actual fontfile is `trebuc.pfb`. Therefore you have to fix the mapfile afterwards.

We leave it as an exercise for the reader to write the `.fd` file and LATEX source for the following example:

Hello, world!
Accents: élève blöf ï; Kerning: WAV, LTa, WAV, LTa.
Symbols: ¶ ‡ ™ ©

Font-based uppercasing and letterspacing

afm2pl comes with an uppercased version *texnanuc* of texnansi. Uppercasing, *e.g.* in headings, works best in combination with letterspacing. For this, afm2pl has a parameter '`-m`'.

Warning: afm2pl implements letterspacing with kerns. Unfortunately, the `.tfm` format can contain only a limited number of kerns. If there are too many in the `.pl` file then all kerns and ligatures will be dropped from the generated `.tfm` file! So use this feature with care. fontinst implements letterspacing

[5]This will result in an empty encoding, unless you specify an encoding parameter. But we are going to ignore the encoding in the `.afm` anyhow.

by adding sidebearings via virtual fonts, and doesn't suffer from this limitation.

We can create a letterspaced, uppercased version of Trebuchet with the following commands:

```
ttf2afm trebuc.ttf >trebuc.afm
afm2pl -p texnanuc -m 100 trebuc trebucupp8y
pltotf trebucupp8y
<edit mapfile to replace .pfb with .ttf>
```

A corresponding fontfamily and fontshape declaration might look as follows:

```
\ProvidesFile{ly1trebuc.fd}
\DeclareFontFamily{LY1}{trebuc}{}
\DeclareFontShape{LY1}{trebuc}{m}{upp}{
  <-> trebucupp8y }{}
```

The fontshape upp for uppercasing is not an official LaTeX shape but that doesn't seem to matter. You can use the font as follows:

```
\documentclass{article}
\usepackage{texnansi}
\pagestyle{empty}
\pdfmapfile{=trebucupp8y.map}

\begin{document}
\fontfamily{trebuc}\fontshape{upp}
\selectfont
Letterspaced uppercasing
\end{document}
```

And this is the result:

```
LETTERSPACED UPPERCASING
```

A font family

The next example uses a real font family, consisting of the usual four family members plus our letterspaced font. So we will need not only trebuc.ttf, as in the previous example, but also trebucbd.ttf, trebucit.ttf, and trebucbi.ttf. For each of these we'll have to run the ttf2afm — afm2pl — pltotf sequence, and we'll have to edit each of the generated map files, or create a combined mapfile.

Here is the `.fd` file:

```
\ProvidesFile{ly1trebuc.fd}
\DeclareFontFamily{LY1}{trebuc}{}
\DeclareFontShape{LY1}{trebuc}{bx}{n}{
  <-> trebucbd8y }{}
\DeclareFontShape{LY1}{trebuc}{m}{n}{
  <-> trebuc8y }{}
\DeclareFontShape{LY1}{trebuc}{bx}{it}{
  <-> trebucbi8y }{}
\DeclareFontShape{LY1}{trebuc}{m}{it}{
  <-> trebucit8y }{}
\DeclareFontShape{LY1}{trebuc}{m}{upp}{
  <-> trebucupp8y }{}
```

And this is the LaTeX code using it:

```
\documentclass{article}
\usepackage{texnansi}
\pagestyle{empty}
% better combine these mapfiles!
\pdfmapfile{=trebuc8y.map}
\pdfmapfile{=trebucbd8y.map}
\pdfmapfile{=trebucit8y.map}
\pdfmapfile{=trebucbi8y.map}
\pdfmapfile{=trebucupp8y.map}

\begin{document}
\fontfamily{trebuc}\selectfont
Hello, \textbf{world!}

Accents: \'el\`eve bl\"of \"i;
Kerning: WAV, LTa, \textit{WAV,
                         \textbf{LTa.}}

Symbols:
\textparagraph{} \textdaggerdbl{}
\texttrademark{} \textcopyright

\fontshape{upp}\selectfont
Letterspaced uppercasing
\end{document}
```

And this is the result:

Hello, **world!**
Accents: élève blöf ï; Kerning: WAV, LTa, *WAV, **LTa.***
Symbols: ¶ ‡ ™ ©
LETTERSPACED UPPERCASING

Using dvips

If you go the dvips route, then you cannot use the `\pdfmapfile` macro. Instead, you have to enter additional mapfiles on the command line:

```
dvips -u +mapfile dvifile
```

The prefix `+` to the `mapfile` parameter is analogous to the `=` prefix for the `\pdfmapfile` macro: it tells dvips to use the named mapfile *in addition to* the default one.

Using afm2tfm

The original intention of afm2tfm was not to create fonts which are used directly by TeX. Instead, they were to serve as a basis for virtual fonts, *i.e.* recipes to compose fonts from other fonts. But it is not too difficult to subvert this intention. The less optimal way is to use the output of afm2tfm directly:

```
afm2tfm kuen408i -p texnansi.enc \
        quick.tfm >a2t.map
<edit a2t.map; see below>
```

Note that the `.afm` filename comes *before* the options. The `.enc` extension must be included.

The better way is to pretend to create a virtual font:

```
afm2tfm kuen408i -T texnansi.enc \
        -v slow.vpl null.tfm >a2t.map
vptovf slow.vpl
rm slow.vf
<edit a2t.map>
```

vptovf generates two files from `slow.vpl`, named `slow.vf` and `slow.tfm`. You should remove `slow.vf`, otherwise the dvi driver or pdftex would think that `slow` is a virtual font.

Mapfile information is written to standard output, which therefore has to be redirected, as shown above. It contains the following string:

```
quick Kuenstler480BT-Italic
  " TeXnANSIEncoding ReEncodeFont "
  <texnansi
```

(everything on one line). This has to be changed into:

```
slow Kuenstler480BT-Italic
  " TeXnANSIEncoding ReEncodeFont "
  <texnansi.enc <kuen408i.pfb
```

(also on one line).

The example below displays differences in spacing between the two: kerns and ligatures were only written to the `.vpl` file, not to `quick.tfm`.

Note. This is not an example for copying.

Hello, world!
Accents: élève blöf ï; Symbols: ¶ ‡ ™ ©;
Kerning: WAV, LTa
Kerning: WAV, LTa

Other options of afm2pl and afm2tfm

With both programs you can artificially slant, narrow and widen a font. afm2tfm can also generate artificial smallcaps. Such manipulated fonts rarely look good, though.

afm2pl also has some options for manipulating the ligkern table and for setting spacing parameters. For casual use, you don't need to bother with these.

OpenType

We are seeing more and more OpenType fonts, which are Unicode-based. These consist of either PostScript/Type 1 or TrueType outlines inside a TrueType wrapper. OpenType fonts may contain huge character sets, sometimes including smallcaps and oldstyle figures.

OpenType fonts with TrueType outlines have an extension `.ttf` and can be treated just like TrueType fonts.

OpenType fonts with Type 1 outlines have an `.otf` extension. You can get an `.afm` for an OpenType file by first converting it with FontForge to TrueType (tip from Taco Hoekwater):

```
fontforge -c 'Open($1); Generate($2);' \
          ofont.otf ofont.ttf
ttf2afm ofont.ttf >ofont.afm
afm2pl -p texnansi ofont ofont8y
pltotf ofont8y
<edit mapfile to replace '<ofont.pfb'
 with '<<ofont.otf'>
```

Note the <<, which means that the font is to be included in full. For commercial fonts, this is usually not allowed.

Or have a look at otftotfm, part of Eddie Kohler's LCDF Typetools and included in TEX Live.

Scripting

Various people have written scripts to automate font installation. ConTEXt users will be familiar with texfont, which, by the way, has an option to use afm2pl instead of afm2tfm.

Each example took several commands on a command line. So why not a script?

Actually, I did use scripts. But my scripts tend to be highly specific to the job at hand, and I keep them with those jobs. So it made more sense to me just to give the necessary commands, and let readers script their own solutions.