The PracTEX Journal
TPJ 2005 No 02, 2005–04–15
Rev. 2005–04–20

Integrating TrueType Fonts into ConTEXt

Thomas A. Schmitz

TEX has a reputation for being difficult when it comes to font management. Many people (mainly those who haven't used any flavor of TEX in a long time) still think that only Computer Modern is available for typesetting in TEX, and there is a consistent rumor that integrating fonts is terribly difficult. While it involves a lot of steps, most of it is handled by automated tools and can be done even by inexperienced users. This tutorial will give step-by-step instructions on how to integrate TrueType fonts with your ConTEXt-installation.

1 Introduction

What is a "font"? Generally speaking, it is a typeface that defines the characters of a certain script in a particular form. There are thousands of ways to draw the letter "A," or "a," or "B" etc., and a good font has to present a coherent overall look so that all characters show a "family resemblance." When we speak of a font in terms of computers, we think not only of this visual presentation, but of the files that are necessary to produce this output. A number of these files probably came with your operating system; you can buy fonts from specialized dealers (sometimes, this will be on CDs containing large collections of different fonts), or you can download fonts on the web, many of them for free. There are three types of fonts that you are likely to find on your computer:

- Postscript type 1 fonts usually have the extension .pfb (if they are in binary format, readable only for computers) or .pfa (readable by humans; both formats hold identical information and can be converted into each other). Every Postscript font contains 256 characters (the actual shape of the character is referred to as a "glyph"), which is exactly the number of character slots that TEX can handle for every given font.

- TrueType fonts (with extension .ttf), on the other hand, can hold many thousands of glyphs when they follow the Unicode encoding, as most do. Some support characters from languages such as Chinese, Russian, or Greek; others hold special mathematical or technical symbols,

while others have additional "expert" glyphs (see below in section 4). Since Unicode aspires to define all characters for all human languages (well, most of them), there's hardly a font around that has glyphs for all the characters that the Unicode standard defines. If you open a Unicode TrueType font in a specialized tool, you will see that most fonts contain only glyphs in certain sections, e.g. for Roman alphabets and Cyrillic letters, for Greek, for Chinese.

- OpenType fonts (extension .otf) combine features of Postscript and Truetype fonts; they can hold Postscript and TrueType data in the same file and allow applications to access, e.g., alternate glyphs for the same character. OpenType is a relatively new standard, and at the time of writing, TEX support for OpenType is still in its beginnings.

TEX can handle TrueType fonts, but this requires some special considerations:

- TEX accepts only 256 characters per font. If we have a TrueType font that holds more than this number of glyphs, we'll have to create more than one file for TEX to produce these different glyphs. This tutorial will show how to create, install, and use the various font files.

- Only pdfTEX can use TrueType fonts. ConTEXt produces pdf-output by default, but if you want to produce dvi or use plain ole LATEX, you can't use TrueType fonts.

- TEX has the ability to manipulate Postscript fonts: it can slant them, extend them or shrink them. This is not possible with TrueType fonts; they can only be displayed "as is."

2 Preparing the Fonts

So let's assume you have found a nice TrueType font somewhere. You have used it with other applications, you love it, and you'd like to use it with TEX as well. We will have to use the command line to produce the files that TEX needs to work with the font. It is best not to do the conversion and creation of the font files within your TEX-installation—if you make a mistake, you'd have to find and delete all the files you installed. Instead, I would recommend you create a directory somewhere on your computer to do all the dirty work. On UNIX-like systems, the best place to do this would be the `/tmp` directory; if you do your work there, the computer will clean up after you the next time you restart. In our example here, we will use the font "Bembo" which you can obtain from many vendors. If you buy this font in TrueType format, you will probably find four files: `Bembo.ttf`, `Bembo-Italic.ttf`, `Bembo-Bold.ttf`, `Bembo-BoldItalic.ttf`, and it's easy to guess what they contain. So create your working directory and put the appropriate font files into it:

```
cd /tmp
mkdir work
cd work
cp PATH_TO_Bembo.ttf .
```

If you're using a Windows or Mac OS X system, we have to take care of some special aspects: In Windows, paths in directory structures are separated by a backslash character (\), and copying is done via the command `copy`, so the section above should read like this:

```
cd \tmp
mkdir work
cd work
copy PATH_TO_fontfoo.ttf .
```

If you're using a Mac, we have to pay attention to other details: Macs traditionally use a "resource fork" that holds all the information about a font. Moreover, chances are that you'll see just one font file without an extension. Macs typically pack several subsets into one font "suitcase," and you'll have to unpack it in order to manipulate the font. To do this you use the utility fondu. If you have Open Office installed on your computer, fondu came with it; you can also install it via Gerben Wierda's excellent i-installer or package-managers such as fink or darwinports. Here's how to do it:

```
cd /tmp
mkdir work
cd /work
fondu PATH_TO_Bembo
```

If all goes well, fondu will extract all font files from the suitcase, and you will usually end up with the same .ttf-files as described above.

Some premodern operating systems could not use file names with more than eight characters, so TEX had to take this restriction into account to be cross-platform compatible. That's why Karl Berry developed his famous naming scheme that would provide a unique eight-character name for every font that would tell experts what they could expect from this font. On today's systems, this restriction doesn't apply any longer, and ConTEXt doesn't use the Berry naming scheme, so you don't have to rename the fonts and can just use the names you find. Nevertheless, I would apply a few cautions: if the name contains spaces and/or special characters (like underscores), remove them. If the extension is in uppercase letters, transform it to lowercase. Apart from that, any name should be fine.

3 Creating Support Files for TEX

Now, we will be using `texfont`, a tool that comes with ConTEXt (for the curious, there's a manual **TeXfont explained** on the pragma website explaining the finer points of this application). It is the equivalent of `fontinst` on the LATEX side, and it can create (almost) all the necessary files if it has a ttf and an afm. So now call texfont (the entire command has to be written on one line):

```
texfont --fontroot=/tmp/work --ve=Vendor --co=Bembo
--en=texnansi
```

Let's have a look at this command. The switch `–fontroot=` specifies the directory into which the files should be installed. I'm kind of paranoid, so I prefer to leave everything in the working directory and copy it over to my texmf-tree once I see that everything was OK. `–ve` is the vendor of the font, `–co` the collection; both are only used to create the appropriate subdirectories in the texmf-tree, so you can call this anything you like. `–en` tells texfont which "encoding" to use. The encoding defines which characters will be included in your file with the 256 characters. Texnansi contains many useful characters for European languages such as OE-ligatures (Œ), accented vowels (à) etc., and it's the favored encoding for ConTEXt. But you could equally well use `–en=EC` to obtain the "T1" encoding popular with LATEX. If you call texfont as described above, without adding the name of the font to convert, it will automatically try to convert every font in the current directory for which a .ttf file exists. You'll end up with a subdirectory "fonts" inside your directory "work" which will contain almost everything you need.

4 Additional Files: Small Caps and Oldstyle-Numerals

I have already mentioned "expert" fonts. They offer additional glyphs, and the most important ones are: additional ligatures (standard fonts usually have ligatures for the combinations "fi" and "fl," expert fonts typically have "ff," "ffi," and "ffl" in addition to those), both normal or "lining" (67890) and oldstyle numerals (67890), and dedicated glyphs for small caps (many applications, TEX included, can fake small caps by taking the uppercase letters and making them a bit smaller, but the typographical quality of FAKE SMALL CAPS will never be as high as with REAL SMALL CAPS). Our Bembo TrueType font has all these expert features, so we will now make them work with TEX.

Texnansi takes care of one aspect of "expert" features in your TrueType fonts: it integrates the ligatures "ff," "ffi," and "ffl" into its character set. But you won't be able to access the

small caps and old-style numerals since they are not included in the files you just produced. In order to do this, we will create a slightly modified encoding. You could do this yourself, or you can simply use the one attached to this article (**texnansi-sc.enc**). If you open the file `texnansi.enc` (which, on my system, resides in the main texmf-tree under `fonts/enc/dvips/psnfssx/texnansi.enc`), you will see how it works. An encoding is basically a list of all the glyphs that the font (as used by TEX) contains, arranging them into certain defined positions or "slots." Characters 48–57 are defined like this:

```
/zero  %       48
/one   %       49
/two   %       50
/three %       51
/four  %       52
/five  %       53
/six   %       54
/seven %       55
/eight %       56
/nine  %       57
```

We now want a new encoding that will put the oldstyle numerals in this position. Most fonts just append the word "oldstyle" to the name of the numeral, so we replace this section with the following definitions:

```
/zerooldstyle  %       48
/oneoldstyle   %       49
/twooldstyle   %       50
/threeoldstyle %       51
/fouroldstyle  %       52
/fiveoldstyle  %       53
/sixoldstyle   %       54
/sevenoldstyle %       55
/eightoldstyle %       56
/nineoldstyle  %       57
```

This works in the case of our Bembo font. However, in some cases, you may have to create your own encoding: some fonts use slightly different names—it could be `one.oldstyle` or `one_oldstyle` or even something like `one.alt`. If this solution does not work for your font, you will have to find out which names your font uses. In order to do this, we create an afm-file for our font; this file contains the metrical information for all the glyphs in our fonts and their postscript-names. Then, we use the utility "grep" to search for possible extensions. Run this command:

```
ttf2afm -o FONTNAME.afm FONTNAME.ttf
grep "seven" FONTNAME.afm
```

This will list all the characters whose name contains "seven." The list you get in return might look like this:

```
C -1 ; WX 490 ; N seven ; B 37 -14 439 622 ;
C -1 ; WX 301 ; N seven.superior ; B 21 298 270 629 ;
C -1 ; WX 301 ; N seven.inferior ; B 22 -6 271 324 ;
C -1 ; WX 718 ; N seveneighths ; B 35 -26 689 653 ;
C -1 ; WX 468 ; N seven.oldstyle ; B 43 -166 436 425 ;
```

In this case, then, we should have "zero.oldstyle" etc. in our modified texnansi-encoding. Next, small caps. The idea is the same: if your font has real small caps, they will have the name of the uppercase letters with an extension, so it should be something like `Asmall`, but it could also be `A.small` or `A_small`. Again, you could grep for a certain letter, say "Z," and see which extensions your font has for it. You will have to replace lowercase letters with these small caps variants in several slots of the texnansi encoding: "ae" etc. in slot 26–8; all the normal lowercase letters in position 64–90, "lslash" in position 144, "scaron" in position 154, "zcaron" in position 157, and all the accented lowercase letters in positions 224–255, so `/otilde% 245` will become `/Otildesmall % 245`. Modify the texnansi encoding file, save it as texnansi-sc.enc and put it into your working directory. An example showing what such a **texnansi-sc.enc** might look like is appended to this article.

Texfont will recognize this encoding as a variant of the texnansi encoding, so we run the same command as before, but tell texfont to use this variant:

```
texfont --fontroot=/tmp/work --ve=Vendor --co=Bembo
--en=texnansi --var=sc
```

5 Installing and Testing the New Font

After every run, texfont will give a brief summary of what it did:

```
TeXFont 2.2.1 - ConTeXt / PRAGMA ADE 2000-2004

mktexlsr: Updating /tmp/work/ls-R...
mktexlsr: Done.
       encoding vector : texnansi
           vendor name : Vendor
```

```
           source path : .
         font collection : Bembo
         texmf font root : /tmp/work
         pdftex map file : texnansi-Vendor-Bembo.map
        processing files : all on afm path
      locating afm files : using pattern ./*.afm
      locating afm files : using ttf files
     generating afm file : ./Bembo.afm

          no map file at : /tmp/work/fonts/map/pdftex/context/
                           texnansi-Vendor-Bembo.map

         font identifier : Bembo-Bold -> text -> tfm + vf
generating raw tfm/vpl : texnansi-raw-Bembo-Bold (from Bembo-Bold)
     generating new vf : texnansi-Bembo-Bold (from texnansi-raw-
                           Bembo-Bold)

              generating : ls-r databases

mktexlsr: Updating /tmp/work/ls-R...
mktexlsr: Done.
```

If everything went well, you'll end up with a collection of files in the subdirectory /tmp/work/fonts/tfm/Vendor/Bembo. These are tfm files (TEX font metrics) that TEX will need to use the font. Copy them into an appropriate directory of your local or home-texmf. If you don't know where this might be, run this on the command-line:

```
kpsewhich -expand-var='$HOMETEXMF'
```

On most linux- or UNIX-systems, this will expand to "/home/username/texmf." So go ahead and copy all the font files you just created (both commands have to go on one line; they are split here for readability):

```
cp -r fonts/tfm/Vendor/Bembo
     /home/username/texmf/fonts/tfm/Vendor/Bembo

cp -r fonts/vf/Vendor/Bembo
     /home/username/texmf/fonts/vf/Vendor/Bembo
```

If you have installed fonts before, there will already be a "font/tfm" subdirectory in your home-texmf, so all you have to do is copy the new "Bembo" directory over; the same is true for the files

in /tmp/work/fonts/vf/Vendor/Bembo. If you've never installed fonts before and are certain that there is no "fonts" subdirectory in your home-texmf, simply copy the entire /tmp/work/fonts subdirectory over.

Then, copy all the TrueType-fonts in /tmp/work to your home-texmf/fonts/TrueType. Your new texnansi-sc.enc goes into home-texmf/fonts/enc/dvips/base/. The last step is adding the mapfile for the new fonts. Texfont creates such a mapfile itself. However, some versions of of pdfTEX have problems with one detail of these mapfiles. The file will look like this:

```
% This file is generated by the TeXFont Perl script.
%
% You need to add the following line to pdftex.cfg:
%
%   map +texnansi-Vendor-Bembo.map
%
% Alternatively in your TeX source you can say:
%
%   \pdf    {+texnansi-Vendor-Bembo.map}
%
% In ConTeXt you can best use:
%
%   \loadmapfile[texnansi-Vendor-Bembo.map]

texnansi-raw-Bembo-Bold Bembo-Bold 4 < Bembo-Bold.ttf
   texnansi.enc
texnansi-raw-Bembo-BoldItalic Bembo-BoldItalic 4
   < Bembo-BoldItalic.ttf texnansi.enc
texnansi-raw-Bembo-Italic Bembo-Italic 4
   < Bembo-Italic.ttf texnansi.enc
texnansi-raw-Bembo Bembo 4 < Bembo.ttf texnansi.enc
```

Some versions of pdfTEX don't like it when there is a space between the < and the name of the .ttf-file, so just remove this space. If you ran texfont with texnansi and texnansi-sc encoding, you will end up having two mapfiles in `/tmp/work/fonts/map/pdftex/context`. I would advise copying all the relevant lines (those not starting with the comment sign %) into one map file Bembo.map and copying it to home-texmf/fonts/map/. After applying these corrections, your mapfile will contain the following lines:

```
texnansi-sc-raw-Bembo-Bold Bembo-Bold 4
   <Bembo-Bold.ttf texnansi-sc.enc
texnansi-sc-raw-Bembo-BoldItalic Bembo-BoldItalic 4
   <Bembo-BoldItalic.ttf texnansi-sc.enc
```

```
texnansi-sc-raw-Bembo-Italic Bembo-Italic 4
   <Bembo-Italic.ttf texnansi-sc.enc
texnansi-sc-raw-Bembo Bembo 4 <Bembo.ttf texnansi-sc.enc
texnansi-raw-Bembo-Bold Bembo-Bold 4 <Bembo-Bold.ttf
   texnansi.enc
texnansi-raw-Bembo-BoldItalic Bembo-BoldItalic 4
   <Bembo-BoldItalic.ttf texnansi.enc
texnansi-raw-Bembo-Italic Bembo-Italic 4
   <Bembo-Italic.ttf texnansi.enc
texnansi-raw-Bembo Bembo 4 <Bembo.ttf texnansi.enc
```

Newer versions of pdfTEX can load map files "on the fly," and ConTEXt has a mechanism for using this feature. However, if you want to use the font with other TEX applications, it is safer to include it into the system-wide configuration files. Do this by running this command (if it complains about missing permissions, you may have to prefix it with "sudo"):

```
updmap --enable Map Bembo.map
```

We now need to test whether pdfTEX can find and use the font. We create a very simple testfile for ConTEXt which contains just these lines:

```
\starttext
\showfont[texnansi-Bembo]
\stoptext
```

Typeset this file with ConTEXt: On the command line, run

```
texexec --pdf  --nonstopmode test.tex
```

If everything is OK, the pdf will show a nice table with all the glyphs in your font. If something goes wrong, check the log-file for errors: does pdfTEX recognize the font? If it doesn't, you may have to run `texhash` on the command line to update TEX's database, but you should also double-check if all the files are in the right subdirectories. Or does it find the font but complains that it doesn't know how to create a font from it? The most likely reason for this is a typo in your mapfile. If TEX says it can't open the ttf, there may be a typo in your map, or the ttf may not be in the proper directory, or you forgot to copy it over. If you need to modify the mapfile, you have to make sure that TEX will now be using the new version, so you have to run two commands:

```
updmap --disable Bembo.map
updmap --enable Map Bembo.map
```

To be absolutely certain that everything works, I recommend you test all the fonts (Italic, Bold, BoldItalic), but be sure to test at least the texnansi-sc variants to see if your font recognizes the small caps and the oldstyle numerals. Often, Italic and Bold variants do not contain small caps, so you might as well discard these files and delete the entry in the map.

6 Preparing support for ConTEXt

If these tests are successful, you could in theory now use the font with ConTEXt. However, you probably want an easy way to make it the bodyfont of your documents. In ConTEXt, this is achieved via typescripts. Unfortunately, texfont does not create them automatically (it should), and typescripts are one of the more arcane parts of ConTEXt. However, writing a basic typescript for using one font family isn't too hard. In your text editor, open a new file and write this:

```
\usetypescriptfile[type-buy]

\starttypescript [serif] [bembo] [texnansi]
  \definefontsynonym [Bembo-Roman]       [texnansi-Bembo]
                                           [encoding=texnansi]
  \definefontsynonym [Bembo-Bold]       [texnansi-Bembo-Bold]
                                           [encoding=texnansi]
  \definefontsynonym [Bembo-Italic]     [texnansi-Bembo-Italic]
                                           [encoding=texnansi]
  \definefontsynonym [Bembo-Bold-Italic]
      [texnansi-Bembo-BoldItalic]          [encoding=texnansi]
  \definefontsynonym [Bembo-Roman-SmallCaps]
      [texnansi-sc-Bembo]                [encoding=texnansi-sc]
  \definefontsynonym [Bembo-Roman-OSF] [texnansi-sc-Bembo]
                                         [encoding=texnansi-sc]
  \definefontsynonym [Bembo-Italic-OSF]
      [texnansi-sc-Bembo-Italic]       [encoding=texnansi-sc]
  \definefontsynonym [Bembo-Bold-OSF]
      [texnansi-sc-Bembo-Italic]       [encoding=texnansi-sc]
  \definefontsynonym [Bembo-Bold-Italic-OSF]
      [texnansi-sc-Bembo-Bolditalic]   [encoding=texnansi-sc]
 \stoptypescript

\starttypescript [serif] [bembo] [name]
  \usetypescript[serif][fallback]
  \definefontsynonym [Serif]           [Bembo-Roman]
```

```
  \definefontsynonym [SerifItalic]     [Bembo-Italic]
  \definefontsynonym [SerifBold]       [Bembo-Bold]
  \definefontsynonym [SerifBoldItalic] [Bembo-Bold-Italic]
  \definefontsynonym [SerifCaps]       [Bembo-Roman-SmallCaps]
  \definefontsynonym [OldStyle]        [Bembo-Roman-OSF]
  \stoptypescript

\starttypescript [Bembo]
  \definetypeface [MyBembo] [rm] [serif] [bembo] [default]
                  [encoding=texnansi]
\stoptypescript
```

Save this file into a directory where TEX can see it (like home-texmf/tex/context/) and call it "type-bembo.tex." I won't go into the details here. The typescript defines a "typeface" that gathers information about all the fonts in the family "Bembo." First, we define which tfm should be used for every member of this family. Then, the names of the different parts of the Bembo-family are mapped to the generic names like "SerifItalic" or "SerifCaps" so ConTEXt knows which font to choose when you issue a command like `\em` or `\os` in your source. In order to use the new font as your bodyfont, you have to include this in the preamble of your document:

```
\usetypescriptfile[type-bembo]
\usetypescript[Bembo]
\setupbodyfont[MyBembo,12pt]
```

If you want to know more about typescripts and how to use them in ConTEXt, have a look at some of the manuals on the pragma site, e.g. **Fonts in ConTeXt. Examples of Using Typescripts** or **Fonts in ConTeXt**.

Let's see if the new font works as expected. We take our file test.tex, include the instructions about the typescripts and modify the part between `\starttext` and `\stoptext` so it looks like this:

```
Very simple test file.

Difficult font definitions baffle users---flee them.

This is in {\sc small caps}, and these are oldstyle numerals:
{\os 12345}.

ToVA -- T{}o{}V{}A
```

Compile this document and double-check whether the font has all the features you expected:

can you see the ligatures “ffl,” “ffi” and —? Do the small caps and the oldstyle numerals work? Can you see the difference between ToVA (which should be kerned, so the “T” reaches slightly above the “o” and “V” and “A” are close together) and ToVA, where this does not happen? If everything is OK: congratulations! You have just finished the complete installation of a TrueType font. Actually, it wasn't that hard, was it?

7 Using the New Font with LaTeX

Are there any LaTeX-users who have followed this article up to this point? Don't despair: since all the different parts of the font now work with TeX, they will work with LaTeX as well. We just have to write a font definition, the LaTeX-counterpart of a ConTeXt typescript. If you have a basic understanding of what the typescript does, you'll recognize familiar structures in the font definition. In your editor, open a new document and write this:

```
\ProvidesFile{ly1bembo.fd}

\DeclareFontFamily{LY1}{bembo}{}
\DeclareFontShape{LY1}{bembo}{m}{n} {
               <-> texnansi-Bembo}{}
\DeclareFontShape{LY1}{bembo}{m}{it}{
               <-> texnansi-Bembo-Italic}{}
\DeclareFontShape{LY1}{bembo}{b}{n}{
               <-> texnansi-Bembo-Bold}{}
\DeclareFontShape{LY1}{bembo}{b}{it}{
               <-> texnansi-Bembo-BoldItalic}{}
\DeclareFontShape{LY1}{bembo}{m}{sc}{
               <-> texnansi-sc-Bembo}{}
\DeclareFontShape{LY1}{bembo}{b}{sc}{
               <-> texnansi-sc-Bembo-Bold}{}
\DeclareFontShape{LY1}{bembo}{bx}{it}{<->ssub * bembo/b/it}{}
\DeclareFontShape{LY1}{bembo}{m}{sl}{<->ssub * bembo/m/it}{}
\DeclareFontShape{LY1}{bembo}{bx}{sl}{<->ssub * bembo/b/it}{}
\DeclareFontShape{LY1}{bembo}{b}{sl}{<->ssub * bembo/b/it}{}
\DeclareFontShape{LY1}{bembo}{bx}{n}{<->ssub * bembo/b/n}{}
\DeclareFontShape{LY1}{bembo}{bx}{sc}{<->ssub * bembo/m/sc}{}

\endinput
```

Save this file as `ly1bembo.fd` into home-texmf/tex/latex/. To make it easy to use the new

font, open another file in your editor. This will contain just a few lines:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{Bembo}

\RequirePackage{texnansi}
\renewcommand{\rmdefault}{bembo}
\endinput
```

This file should be saved as `Bembo.sty` into home-texmf/tex/latex. Let's test again. We open a new testfile with this content:

```
\documentclass[12pt]{article}

\usepackage{Bembo}

\begin{document}

Very simple test file.

Difficult font definitions baffle users---flee them.

This is in \textsc{small caps}, and these are oldstyle numerals:
\textsc{12345}.

ToVA -- T{}o{}V{}A

\end{document}
```

If you compile this file with pdfLaTeX(remember, TrueType fonts do not work with the vanilla TeX, and hence with the vanilla LaTeX), you should get another nice pdf-file that will look *almost* as good as the one produced with ConTeXt. `\usepackage{Bembo}` will switch to the new font (beware: this is simply a text font, so if you're a LaTeX user who needs lots of math, this font will not be for you!). The font as we adapted it follows the texnansi-encoding, that's why we need to call the `texnansi` package in our `Bembo.sty`.

8 Beyond texnansi

If you've been successful so far, you now have a bodyfont that will work for normal documents

and provide all the characters present in the texnansi-encoding. But didn't I say that many TrueType fonts contain characters beyond this range? Wouldn't it be nice if we could access them, too? The good news is: you can. The bad news is: this is a bit more complicated, and it is not automatized, so you may have to write some files manually. You want to give it a go? OK, let's see how this works. But first, I'll explain why this is a bit more complicated.

The characters we have been using so far are pretty standard, and accordingly, they have standard names (defined by the software company Adobe) that *most* font designers (not all, unfortunately) use. These names are used by pdfTEX to extract the corresponding glyphs from our TrueType file `Bembo.ttf` and put them in the output. How does TEX know which glyph we want? Well, you can have a look at the encoding file which is attached: it provides an association between a *name* (let's say "Eacute") and a certain *position* in the encoding (here, number 201). On the other hand, TEX knows that, when you have something like `\'{E}` or (with the proper input regime or inputenc-option if you're using LATEX) É in your source, this should translate to character 201 in your given font. So a very simplified image of the process would be:

1. Your source has an accented character or a TEX-command to build an accented character; the interior mechanism of TEX translates this to "character 201."

2. The encoding file specified in the font encoding translates this character to the name of a glyph.

3. The pdf-driver takes the glyph with this name from our TrueType font and puts it into the pdf.

The same is, of course, possible for every character in a TrueType font: you can use an encoding file that will tell TEX the name of the glyph you're referring to, and TEX will be able to extract this glyph. You will find these names in the afm-file we produced in section 2. Open it in your favorite text editor and have a look at it. You'll find lines like this:

```
C -1 ; WX 832 ; N Uogonek ; B 11 -232 831 690 ;
```

This gives the metrics for a certain character and its name; in this case "Uogonek." The information in this file is also contained in the ttf (remember: we extracted the afm from the ttf), so this is how TEX finds the glyphs. But here's the catch: remember that I said there are standard names for the glyphs in texnansi? Well guess what: for most other glyphs, that's not true. Once you venture outside these sections into something like Hebrew or accented Greek characters, font designers ususally make up their own names. Some of them do a good job and use either a descriptive name for their glyphs (something like "alphalenisacute" for the Greek letter alpha with a soft breathing and an acute accent), or they simply form a name from the Unicode value of the character (something like "u1F04" for the character just mentioned). Others are extremely sloppy and use random names, or even the same name for several glyphs (recently, I

tried to convert a font where most characters beyond the normal range were invariably called "newline"). In the latter case, abandon all hope—short of editing the font with a specialized tool like fontforge and providing names yourself, you won't be able to use it with TEX. If your font uses reasonable names, however, and if you're willing to spend some time on making it work, here's how to do it.

First, in the afm-file, look up the names of the characters you want to use. You can use up to 256. Copy these names into a new file, and precede every one of them with a slash. I would recommend appending a number to every character, after a comment sign: this number will not be read by TEX, but it helps you keep track. So a small section of this file could look like this:

```
/Alpha                          % 65
/Beta                           % 66
```

This would put the glyph named "Alpha" in slot 65 (that's where you usually find "A"), "Beta" in 66 etc. You will need to provide a file with exactly 256 names preceded by a slash, starting with number 0 up to number 255. If there's a slot you can't (or don't want to) fill, call it `/.notdef`. The encoding has to start with a line like this:

```
/myencoding[
```

and end with a line like this:

```
] def
```

Save this file as "myecoding.enc" and put it with the other encodings in your home-texmf.

You can now use this encoding just as we used texnansi: run texfont on Bembo, but use `–en=myencoding`. This will give new tfms and vfs and a new mapfile; copy all of them into the appropriate directories, run `updmap` and test this new "font" (for TEX, this tfm constitutes a new font, even if it's just a subset of our TrueType Bembo). Finally, write a typescript file `type-bembospecial` that defines this font. In your document, you could then access the glyphs by putting this into the preamble:

```
\usetypescriptfile[type-bembospecial]
\usetypescript[Bembospecial]
```

In order to use these glyphs, just issue this command in your TEX-file:

```
switchtobodyfont [MyBembospecial]
```

9 In Conclusion

Like most aspects of TEX & friends, managing fonts can be a bit intimidating at first. Unlike WYSIWYG applications where you can install fonts by just dragging and dropping them into certain directories, TEX needs some preparation. But the outstanding typographical quality of TEX output fully justifies investing so much time; TEX lets you use fonts to their fullest potential.

Happy TEXing!