FontKit

Melissa O'Neill ⟨oneill@cs.hmc.edu⟩

March 22, 2002 Release

Contents

1 Purpose of FontKit

FontKit is a tool that provides the "glue" allowing multiple-master fonts to be used with the TeX typesetting system, and, in particular, the LaTeX system. As such, FontKit

- Generates TFM files (TeX Font Metrics) from AFM (Adobe Font Metrics) data
- Creates virtual fonts for TeX so that Adobe fonts can be used in a "natural" way in TeX

- Provides descriptions of multiple-master fonts in the form of `.fd` files. These files provide the translation from NFSS font names to Karl Berry names (e.g., `LY1/pkp9/bsx/sw/10.95` (meaning LY1-encoded Adobe Kepler with old-style figures (bold, semi-extended swash) at size 10.95 pt) to the Karl Berry name `pkpbw9ys1095`)

Multiple-master fonts are unlike regular PostScript fonts because a single PFA file can be the source of a plethora of font instances. Under Unix TEX systems based on the Web2C TEX sources (such as teTEX), FontKit can generate metrics on the fly, rather than forcing the user to generate the metrics for all possible font instances in advance. To continue the example above, when TEX finds that no font-metrics file `pkpbw9ys1095.tfm` exists, it runs the command `mktextfm pkpbw9ys1095`. FontKit provides an implementation of this command that decodes the cryptic Karl Berry name and determines that a request was made for "Kepler MM 10.95pt Bold SemiExtended Swash", uses its internal tables to determine the weight and width values Kepler uses for "Bold" and "SemiExtended", and then runs the command `mmafm KeplMM-Sw_650_710_10.95_` to generate Adobe font metrics for this font instance, which it then converts to TEX metrics (`pkpbw9ys1095.tfm`). Depending on the setup, it may also generate a PFA font (non–multiple-master) called `pkpbw9ys1095.pfa` using `mmpfb`.

Tools such as `fontinst` can do some of these things, but have no need to generate metrics on the fly and rarely have to deal with the multiplicity of fonts that arise with mutiple-master fonts. They also don't usually have to deal with non-integer values in font-metrics descriptions because such values don't usually arise for ordinary fonts.

FontKit can write metrics for fonts that aren't multiple-master fonts (i.e., single-master fonts). Often, however, it's just as easy to download those metrics from other sources or generate them with another tool, such as `fontinst`.

1.1 Why Does FontKit Exist?

I wrote FontKit because I wanted to use my multiple-master fonts with TeX. At the time I began writing FontKit, tools such as `fontinst` were inadequate for working with multiple-master fonts. I had no desire to reinvent the wheel, and initially, hoped to extend `fontinst`. Having used `fontinst`, there were many things I liked about it, including the way the rules for installing a particular font were written in a declarative style. Although `fontinst` had some design pluses for users, its raw TEX implementation was hard to follow. TEX is, after all, designed for typesetting, not general-purpose programming. I realized that I would find it easier to write my own tools from scratch than attempt to grok the deep TEX hackery that went on inside `fontinst`.

The code for FontKit was written for my own use with little thought given to the needs of other people. But I found that people kept asking me if I could let them use the same tool I used. I wasn't completely happy about sharing my code because all I had was something I considered "very preliminary, but working", but in the end I realized that even with all the rough edges, a smart person could get it working and have as much fun with multiple-master fonts as I was having.

2 Installation

Once FontKit is properly installed you can more-or-less forget it. Font instances are created on-the-fly as necessary and you don't need to worry about what is where or how it all works. But first we need to get to that point....

2.1 Before Installing FontKit

Before installing FontKit, you must have downloaded and installed Eddie Kohler's `mminstance` multiple-master font tools (consisting of the `mmafm` and `mmpfb` programs).

Depending on the output method you choose (see Section 2.4), you may also want to install Eddie's `t1utils` package, consisting of several utilities for manipulating and converting binary and ASCII fonts. See Section 6 for information about obtaining and installing these packages.

Finally, because FontKit is written in Perl, you'll also need a recent (> 5.6) version of Perl on your system.

This documentation assumes you'll be using FontKit with teTEX on a Unix-like system, such as Linux, FreeBSD, Solaris, or Mac OS X. If you're planning on using FontKit with some other TEX system, some of its parts may be usable, but it won't be able to generate font instances on-the-fly.

2.2 Choosing Where to Put Things

On a Unix system, you should be able to download FontKit, unpack it in your home directory, and edit your `.cshrc` (or `.login`) as follows:

```
setenv FONTKIT_BASE ~/FontKit
set path = ( $FONTKIT_BASE/bin $path )
```

(Note that it is vital that FontKit's `mktextfm` is found in your search path before the `mktextfm` program supplied by teTEX; the `set path` line shown above should achieve that goal. If you decide to install FontKit in another location, be sure to set `FONTKIT_BASE` appropriately.)

You then need to add lines to your `.cshrc` or `.login` file so that TEX and friends can find the FontKit files, as follows:

```
set noglob
    set standard_texmf = ‘kpsewhich --expand-var='$TEXMF'‘
unset noglob
setenv TEXMF        "$FONTKIT_BASE/texmf,$standard_texmf"
```

(Assuming that you've installed FontKit at the root of your home directory, of course; adjust the paths as necessary.)

To make it easy to copy these lines into your `.login`, you can find the above code in `fontkit-setup.sh` in the `contrib` directory.

2.2.1 Installing Files in Other `texmf` Trees

If you don't want all the FontKit files to be self-contained in their own hierarchy (say, for example, that you want FontKit to write files into your personal or system `texmf` tree or store your fonts somewhere else, you can instead add

```
setenv FONTKIT_TEXMF   ~/texmf
setenv FONTKIT_PSFONTS ~/mm-fonts
```

modifying the paths as appropriate and copying the FontKit `texmf` files into the `texmf` tree you plan to use.

2.3 Installing the Font Metrics

Next you have to install the font metrics for the fonts you will be using. Because FontKit uses `mmafm` to fetch font metrics for instances, you have some degree of latitude as to how you install your font metrics—provided `mmafm` can find your metrics without any special command-line arguments, FontKit should be happy.

Nevertheless, it usually makes sense to install the font-metrics files for your multiple-master fonts in FontKit's `font` directory. On my machine, I use a convention where the directory structure based on the font names, thus the AMFM file for JensonMM Italic is

```
fonts/AJensonMM-It.font/AJensonMM-It.amfm
```

If you continue to use this same structure, you need not worry about AFM, AMFM, and AMCP files for Cronos, Jenson, Kepler, Minion, Myriad, Nueva, or Tekton because these files are already provided. If you want to adapt the directory structure for your own files, you may be glad to know that the directory can use any structure and location you like, provided that you are able to generate an Adobe-style `PSres.upr` file to describe what files are there and that you define `FONTKIT_PSFONTS` to point to the directory.

2.4 Deciding Between Output Using Master Fonts Directly or Generated Instance Fonts

When printed on a PostScript printer, the printer can create all the instances of a given multiple-master font from the master font itself. However, many tools do not know how to handle multiple-master fonts with the seamless ease possible in PostScript. Early versions of `ps2pdf`, for example, choke on multiple-master fonts, and Mac OS X cannot abide them. `PDFTeX`, too, throws up its hands in horror at the sight of a multiple-master font. Thus it can often be a good idea to let FontKit generate free-standing PostScript fonts for all the instances you need from the multiple-master font. By default, FontKit generates these instances.

The downside of generating instance fonts is their size—each one is close to the size of the master font itself. Even though instances are only made on demand, it is quite easy to create a large number of instances with even simple LaTeX documents.

My own collection of PFA instances is currently about 120 MB. If you wish to disable PFA instance generation, you can comment out the definition of `$instancedir` in `mktextfm`.

2.4.1 Installing PFA Files

You will, of course, also need the Type 1 multiple-master font files for your fonts. FontKit expects to work with all fonts in PFA form (PFB fonts may work too, with a few tweaks, but such a configuration is untested). If you don't already have your fonts in PFA format, you may wish to use the `t1unmac` program to convert Macintosh font files into PFA format, or the `t1ascii` program to convert PFB (Windows) files into PFA files. Both programs are part of Eddie Kohler's `t1utils` package of font utilities (see Section 6).

At present, depending on how you use FontKit (see Section 2.4), you may need to install your PFA files so that `dvips` can find them (if you will be using master PFAs for output), or so that Eddie Kohler's `mmpfb` program can find them (if you will be using instance PFAs for output).

If you want to install the master PFAs so that `dvips` can find them, they need to be named using the cryptic Karl Berry names (see Section 6 for references) and be stored in a subdirectory of the `fonts/type1/` area of a `texmf` tree in TEX's search path (your personal `texmf` tree, the one included in the FontKit installation directory, or your system's `texmf` tree).

If you plan to use `mmpfb` (which is probably the best choice if disk space isn't an issue), the master PFAs need to be in the FontKit directory and named using the conventions given in Section 2.3).

The easiest solution is to install the PFA files in FontKit's `fonts` directory using the easier to read file names and make symbolic links to these files with the Karl Berry names. If you are using Cronos, Jenson, Kepler, Minion, Myriad, Nueva, or Tekton, you may find that you can trivially adapt the shell script `make-kb-links` to do exactly what you want. Similarly, the `make-pfa-links` script can do the work for a brand new font.

2.5 Testing the Installation

If you've installed everything in the right places and set up your environment variables properly, you can try typing

```
mktextfm pkpr8y1000
```

to create an instance of Kepler at 10 pt size. (If you don't have the PFAs for Kepler, of course, you'll have to use a different font—for example, `pmnr8y1000` for Minion or `pajr8y1000` for Jenson—see Table 1 to figure out a good name for a font that you have.)

2.6 Adding the Map Files

dvips and PDFTEX need *map files* that tell them where to find the multiple-master font instances mentioned in the DVI file. (These map files—mm.map, mm-loading.map, and mm-instance.map—are maintained by FontKit. Bad things can happen if you make random changes to these files by hand and get them out of sync with the TFM and PFA files they refer to.)

With dvips, you can use

1. `dvips -Ppdf -Pmm-loading myfile.dvi`
   To create a PS file that uses the master PFA files
2. `dvips -Ppdf -Pmm-instance myfile.dvi`
   To create a PS file that uses the instance PFA files
3. `dvips -Ppdf -Pmm myfile.dvi`
   To create a PS file that assumes that the PFAs are already downloaded into the printer

PDFTEX should automatically use the pdftex.cfg file that came with FontKit, which directs it to use mm-instance.map. Unfortunately, PDFTEX does not support a command-line switches to provide supplementary configuration files, so the pdftex.cfg file provided with FontKit actually overrides any existing pdftex.cfg file you may have. If you do not wish lose any customizations you may have made to your pdftex.cfg file, you can instead delete the pdftex.cfg file that comes with FontKit and instead add the following lines to your own pdftex.cfg:

```
map +mm-instance.map
map +ly1-loading.map
```

If you use teTeX or TeXLive, you should *not* use the updmap command to put the map file contents into the system pdftex.map because mm-instance.map *changes* over time as font instances are generated.

3 Using Multiple-Master Fonts with LATEX

Once you have FontKit's core working, you should be able to typeset documents using multiple-master fonts, and FontKit will generate instances for all the different font shapes, styles, and sizes used in your document.

3.1 Style Files Included with FontKit

Style files that automatically set the default font-families are provided for Kepler, Jenson and Minion (kepler.sty, jenson.sty, and minion.sty, respectively). Load them with \usepackage as you would with any other package.

By default, kepler.sty and minion.sty specify Myriad as the sans-serif typeface. All three packages specify Courier as the "typewriter text" font. You may want to use these style files as models for packages supporting other multiple-master typefaces you may have installed.

3.2 Ornaments and Swash Italics

The expert sets for Cronos, Kepler, and Minion include "ornament" fonts with various borders, bullets, fleurons, and so forth. These can be used by loading the appropriate package (`cronosornaments.sty`, `keplerornaments.sty`, and `minionornaments.sty`, respectively).

Some of the expert sets include "swash" italics—alternate letterforms for italic capitals ("*Yo!*" rather than "*Yo!*"). You can use the swash fonts by writing `\textsw{Yo!}`. The multiple-master support code for LaTeX also redefines the `\textsc` command so that it pays attention to whether it is being used in italic text, and uses `\itscshape` rather than `\scshape` when necessary. An additional command, `\textlc`, is provided to switch out of small caps to regular lowercase letters. Due to limitations in LaTeX, italics and small caps are not completely independent—if you use the lower-level font selection commands you will not get this behavior.

4 Fonts Available in LaTeX

Table 1 lists the fonts that are available when using FontKit as installed, assuming that the corresponding PFA font files are installed. All fonts in Table 1 provide small caps and other font-specific features. (In the case of Nueva, Myriad, and Tekton, their availability depends on appropriate MM parameters being present in `MMdesc.pm`).

5 Undocumented Things and Rough Edges

When I wrote FontKit, I heeded Fred Brooks's advice from *The Mythical Man Month*, where he says "Plan to throw one away, you will anyhow". Thus FontKit was written, at least in part, to understand the problem space and not to be a perfectly clean design. Alas, the prototype has become the product. It works for me and I just don't have the time to perform the ground-up rewrite that I thought necessary for a public release of the code. Nevertheless, enough people have begged me for the code that I've decided that I should release it as-is and hope that someone else has the time to improve on it.

There is much left to document, unfortunately. Ideally, to use FontKit you should be prepared to get your hands dirty figuring out Perl code.

The worst aspect of FontKit is the `mktextfm` script. It is a mess of exceptions and uncommented code (not that the rest is much better), in part because it provides glue between TEX, `mminstance`, `dvips`, and the features available in different Adobe fonts. Quite probably there is no way to make this glue elegant.

Similarly, there needs to be more discussion of the rationale behind some of the design choices. For example, the supplied FontKit scripts use the LY1 encoding (see ⟨http://www.yandy.com/ly1.htm⟩ for details about this encoding) rather than T1+TS1, mostly because LY1 is much easier to deal with and every bit as powerful in practice.

Berry Name	Font	Notes
pkc9	Adobe Cronos	Text with old-style figures
pkc	Adobe Cronos	Text with titling numerals
pkc0	Adobe Cronos	Inferior numerals (subscripts)
pkc1	Adobe Cronos	Superior numerals
paj9	Adobe Jenson	Text with old-style figures
paj	Adobe Jenson	Text with titling numerals
paj0	Adobe Jenson	Inferior numerals (subscripts)
paj1	Adobe Jenson	Superior numerals
paj6	Adobe Jenson	Alternate glyphs and ligatures
pkp9	Adobe Kepler	Text with old-style figures
pkp	Adobe Kepler	Text with titling numerals
pkp0	Adobe Kepler	Inferior numerals (subscripts)
pkp1	Adobe Kepler	Superior numerals
pmn9	Adobe Minion	Text with old-style figures
pmn	Adobe Minion	Text with titling numerals
pmn0	Adobe Minion	Inferior numerals (subscripts)
pmn1	Adobe Minion	Superior numerals
pmy	Adobe Myriad	
pne	Adobe Nueva	
ptk	Adobe Tekton	

Table 1: Fonts available with the default installation of FontKit.

6 Additional Resources

6.1 Additional Software

Eddie Kohler maintains two sets of useful programs for manipulating PostScript fonts:

- `mminstance` includes the `mmafm` program for creating single-instance AFM files from multiple-master AMFM files and the `mmpfb` program to create a single-master font instance from the PFA or PFB font file. FontKit requires `mminstance`.
- `t1utils` includes six programs for manipulating Type 1 fonts, including converting between ASCII (PFA) and binary (PFB) formats; converting files to and from Macintosh font format; and disassembling and assembling fonts. You may need some of these programs to convert commercial fonts meant for Windows or Macintosh systems to the formats FontKit requires.

Both sets of programs are available as source and RPM files from Eddie's website, at ⟨http://www.lcdf.org/~eddietwo/type/⟩. Debian users can install them from the main Debian archive by typing (in a root shell)

```
apt-get update && apt-get install mminstance t1utils.
```

6.2 Karl Berry Font Names

Karl Berry's font-naming scheme is described in a document called *Fontname* that is included in teTEX's documentation. (Look in *system_directory*/texmf/doc/fonts/fontname/.) You can also find the most recent version of this document at ⟨http://www.tug.org/fontname/⟩.

7 Sample Runs

To get a better idea of what is "normal" running LATEX with FontKit, here is an example of what it looks like to run LATEX on a small file that prints "Hello World!" in Adobe Kepler. The file is as follows:

```
\documentclass{article}
\usepackage{kepler}
\begin{document}
Hello World!
\end{document}
```

We'll examine what happens when we typeset this file on a machine where FontKit has just been installed. When we run LATEX on the document, FontKit will have to have to make the necessary font instances.

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
(hello.tex
LaTeX2e <2000/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, ngerman, i
talian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo))
(/home/melissa/FontKit/texmf/tex/latex/mm/kepler.sty
(/home/melissa/FontKit/texmf/tex/latex/mm/mmaster.sty
(/home/melissa/FontKit/texmf/tex/latex/mm/ly1enc.def)
(/home/melissa/FontKit/texmf/tex/latex/mm/ly1pkp9.fd)kpathsea: Running mktextfm
 pkpr9y1000
Requested: pkpr9y1000 (Kepler MM 10pt Regular)

Loading TeXnANSIEncoding from texnansi.enc... done.

Loading KeplMM_385_575_10_... reencoding => pkpr8y1000.
    Writing pkpr8y1000.tfm
        Missing glyph 'Uni20AC' for TeXnANSIEncoding[1]
        Missing glyph 'cwm' for TeXnANSIEncoding[10]
        Missing glyph 'ff' for TeXnANSIEncoding[11]
        Missing glyph 'ffi' for TeXnANSIEncoding[14]
        Missing glyph 'ffl' for TeXnANSIEncoding[15]
        Missing glyph 'dotlessj' for TeXnANSIEncoding[17]
        Missing glyph 'nbspace' for TeXnANSIEncoding[160]
        Missing glyph 'sfthyphen' for TeXnANSIEncoding[173]
        pltotf: rounded some heights by 2.0909004 units.
        pltotf: rounded some depths by 1.8521004 units.
```

```
Executing: mmpfb --pfa KeplMM_385_575_10_ -o /home/melissa/FontKit/texmf/fonts/t
ype1/mm-instances/pkpr8a1000.pfa
Loading KeplMM-SC_385_575_10_... reencoding => pkprc8y1000.
    Writing pkprc8y1000.tfm
        Missing glyph 'Uni20AC' for TeXnANSIEncoding[1]
        Missing glyph 'cwm' for TeXnANSIEncoding[10]
        Missing glyph 'ff' for TeXnANSIEncoding[11]
        Missing glyph 'ffi' for TeXnANSIEncoding[14]
        Missing glyph 'ffl' for TeXnANSIEncoding[15]
        Missing glyph 'dotlessj' for TeXnANSIEncoding[17]
        Missing glyph 'nbspace' for TeXnANSIEncoding[160]
        Missing glyph 'sfthyphen' for TeXnANSIEncoding[173]
        pltotf: rounded some heights by 2.0612001 units.
        pltotf: rounded some depths by 1.7244501 units.
Executing: mmpfb --pfa KeplMM-SC_385_575_10_ -o /home/melissa/FontKit/texmf/font
s/type1/mm-instances/pkprc8a1000.pfa
Loading KeplMM-Ep_385_575_10_... => pkpr8x1000.
    Writing pkpr8x1000.tfm
        pltotf: rounded some heights by 1.5485497 units.
        pltotf: rounded some depths by 1.1892004 units.
Executing: mmpfb --pfa KeplMM-Ep_385_575_10_ -o /home/melissa/FontKit/texmf/font
s/type1/mm-instances/pkpr8x1000.pfa

Creating Expertized Variant... normal glyphs... digits... extras...  done
    Writing pkpr9y1000.tfm and pkpr9y1000.vf
        Missing glyph 'Uni20AC' for TeXnANSIEncoding[1]
        Missing glyph 'cwm' for TeXnANSIEncoding[10]
        Missing glyph 'dotlessj' for TeXnANSIEncoding[17]
        Missing glyph 'nbspace' for TeXnANSIEncoding[160]
        Missing glyph 'sfthyphen' for TeXnANSIEncoding[173]
        vptovf: rounded some heights by 2.1242504 units.
        vptovf: rounded some depths by 1.8214502 units.
))
No file hello.aux.
(/usr/share/texmf/tex/latex/lucidabr/ly1ptm.fd) [1] (hello.aux) )
Output written on hello.dvi (1 page, 252 bytes).
Transcript written on hello.log.
```

TEX needed `pkpr9y1000.tfm`, didn't find it, and so asked FontKit to provide this file. FontKit did so, and in so doing also made `pkpr9y1000.vf` because the expertized variant is actually a virtual font, drawing on the glyphs from three of the Adobe fonts that make up the Adobe Kepler collection. FontKit also made TFMs for those fonts: `pkpr8x1000.tfm`, `pkpr8y1000.tfm`, and `pkprc8y1000.tfm`.

You may notice a lot of warnings about "missing glyphs". These warnings occur because the font encoding FontKit uses (Y&Y's LY1 encoding) has slots for glyphs that aren't present in standard Adobe fonts. These warnings are harmless because no code that we write in TEX will try to access any of these glyphs directly (also notice that the expertized variant has fewer of these warnings because it uses the expert fonts as a source for some of its glyphs.

FontKit also made PFA files for each of these base fonts (i.e., `pkpr8x1000.pfa`, `pkpr8y1000.pfa`, and `pkprc8y1000.pfa`) and added the lines

```
pkpr8y1000  KeplMM_385_575_10_ "TeXnANSIEncoding ReEncodeFont " <texnansi.enc < pkpr8a1000.pfa
pkprc8y1000 KeplMM-SC_385_575_10_ "TeXnANSIEncoding ReEncodeFont " <texnansi.enc < pkprc8a1000.pfa
pkpr8x1000  KeplMM-Ep_385_575_10_ < pkpr8x1000.pfa
```

to `mm-instance.map` (and analogous lines to `mm.map` and `mm-loading.map`).

Running LaTeX on the file for a second time results in no calls to FontKit because everything TEX needs is now available. To keep things interesting we'll run `pdflatex` rather than `latex`:

```
This is pdfTeX, Version 3.14159-14h-released-20010417 (Web2C 7.3.3.1)
(./hello.tex{/home/melissa/FontKit/texmf/pdftex/pdftex.cfg}
LaTeX2e <2000/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, ngerman, i
talian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo))
(/home/melissa/FontKit/texmf/tex/latex/mm/kepler.sty
(/home/melissa/FontKit/texmf/tex/latex/mm/mmaster.sty
(/home/melissa/FontKit/texmf/tex/latex/mm/ly1enc.def)
(/home/melissa/FontKit/texmf/tex/latex/mm/ly1pkp9.fd))) (./hello.aux)
(/usr/share/texmf/tex/latex/lucidabr/ly1ptm.fd) [1{/usr/share/texmf/dvips/confi
g/pdftex.map}{/home/melissa/FontKit/texmf/dvips/mm/mm-instance.map}{/home/melis
sa/FontKit/texmf/dvips/ly1/ly1-loading.map}] (./hello.aux) )</home/melissa/Font
Kit/texmf/fonts/type1/mm-instances/pkpr8x1000.pfa>{/home/melissa/FontKit/texmf/
dvips/base/texnansi.enc}</home/melissa/FontKit/texmf/fonts/type1/mm-instances/p
kpr8a1000.pfa>
Output written on hello.pdf (1 page, 34312 bytes).
Transcript written on hello.log.
```

If you're making PDF, that's about all you need to know. But if we were making PostScript, then we would want to run `dvips` to turn our DVI file into a PostScript file. Typically, you'd type

```
linux% dvips -Ppdf -Pmm-loading -o hello-masterpfa.ps hello.dvi
This is dvips(k) 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
' TeX output 2002.02.18:0119' -> hello-masterpfa.ps
dvips: ! Couldn't find header file pkpz8a.pfa
```

Oops, looks like we forgot to make the links so that `dvips` can find our PFA files for the multiple-master fonts under the cryptic Karl Berry names. Thankfully, I can just run `make-pfa-links` from the FontKit distribution because it makes the necessary symbolic links for all the fonts *I* have. That done, we can try again:

```
linux% dvips -Ppdf -Pmm-loading -o hello-masters.ps hello.dvi
This is dvips(k) 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
' TeX output 2002.02.18:0119' -> hello-masters.ps
<texc.pro><texnansi.enc><pkpz8a.pfa><pkpz8x.pfa><texps.pro>. [1]
```

We could avoid the need to make links if we make the PostScript file using the instance PFA files. We can do that by running

```
linux% dvips -Ppdf -Pmm-instance -o hello-instances.ps hello.dvi
This is dvips(k) 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
' TeX output 2002.02.18:0119' -> hello-instances.ps
<texc.pro><texnansi.enc><texps.pro>. <pkpr8x1000.pfa> Second number not found in 549 Subr string
```

Whoops—now looks like there's a bug in `dvips`, because it's choking on our PFA files. Let's turn off font subsetting and see if that helps:

```
linux%  dvips -j0 -Ppdf -Pmm-instance -o hello-instances.ps hello.dvi
This is dvips(k) 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
' TeX output 2002.02.18:0119' -> hello-instances.ps
```

Aha, problem solved. Of course, both the PostScript files are going to take lots of printer memory because they use unsubsetted fonts. Probably if we don't have an industrial-strength PostScript printer, it'd be better to just print the PDF file we'd generated with `pdflatex`.