METAFONT for Beginners

Second Draft

Geoffrey TOBIN*†

Scope

This is not a tutorial in METAFONT. It is an attempted description of how some of the pitfalls in running the program may, hopefully, be avoided.

Reference

"The METAFONTbook", by Donald Ervin KNUTH, published by the American Mathematical Society and Addison Wesley Publishing Company. First edition, 1986, covers METAFONT 1.0. Later editions cover METAFONT 2.0 and above. This file is based on the 1986 edition.[1]

Motivation

It's a common experience to have initial (and medial and final `:-)`) difficulty with running METAFONT, and not all "TEXperts" are as familiar with MF as they are with TEX. Still, nothing ventured, nothing gained. So let's be of good cheer, and get down to work.

1 Overview

1.1 Font files

METAFONT is a program for making bitmap fonts for use by TEX, its viewers, printer drivers, and related programs. It interprets a drawing language with

*ecsgrt@luxor.latrobe.edu.au

†With three corrections from David KASTRUP (dak@kaa.informatik.rwth-aachen.de).

[1]Opinion: I actually enjoy reading The METAFONTbook, whereas The TEXbook confuses me no end.

a syntax slightly like Pascal's. The input can be interactive, or from a source file. METAFONT source files are usually suffixed ".mf". The output is a GF ("generic font") file, suffixed ".gf", which contains the bitmap. This may be compressed to a PK ("packed") font by the auxiliary program GFtoPK.

A bitmap is all that's needed for large-scale proofs, as produced by the GFtoDVI utility, but for TEX to typeset a font it needs a TFM ("TEX Font Metric") file to describe the dimensions, ligatures and kerns of the font. METAFONT can be told to make a TFM file, by turning on fontmaking. Most output device modes (see part 1.2 below) do this.

More detailed descriptions of TFM, GF and proof mode are found in Appendices F, G, and H, respectively of "The METAFONTbook".

The DVI Drivers Standard, Level 0, draft 0.05, includes precise definitions of the file formats of TFM, GF and PK fonts. That document is obtainable from such TEX archives as ftp.uni-stuttgart.de, where it is currently found as the several files in the directory
soft/TEX/drivers/dvi-standard/level-0.

1.2 Base files and Modes

The METAFONT language contains two or three hundred primitives, which are the commands preceded by an asterisk in the Index (Appendix I) to The METAFONTbook. From these we can build more complex operations, using macros. In METAFONT macros have some of the desirable characteristics of functions in other languages. Collections of macros can be stored in METAFONT source files.

Base files are "precompiled binaries" that METAFONT loads faster than it loads the original source files.

The plain base provides the commands that The METAFONTbook describes. (See Appendix B of "The METAFONTbook", if you have it around - maybe a library has it - I'm learning from a copy borrowed from the local university's library.)

When it starts, METAFONT automatically loads[2] the "plain" base. This is usually called "plain.base". In Karl Berry's web2c distribution for unix, it's named "mf.base". EmTEX for DOS calls it "plain.bas", due to filename truncation.

[2]There are releases of METAFONT that contain the plain base, and so don't have to load it. However, on most computers, including personal computers, reading bases is so fast that such a "preloaded" base is unnecessary.

The plain base is made from a METAFONT file named “plain.mf” and, commonly, from some other file, often called “local.mf” or “modes.mf”.

The local modes file lists printers (and monitors), giving each a font-making “mode”, containing a description of some refinements that must be made in order to produce good-looking output. For example, how dark to make the characters.

If you want to make a base, you need a variant of the METAFONT program called “inimf”. (See The METAFONTbook, p 279.) For example, “plain.base” can be made in unix by typing:

```
inimf 'plain; input local; dump'
```

If using the emTEX version of METAFONT for a PC, type:

```
mf/i plain; input local; dump
```

1.3 Proof Mode

The first obstacle to beware of is that plain METAFONT uses “proof” mode by default. (The METAFONTbook, page 270, defines this mode.) That means writing unmagnified font files with a resolution of 2601.72 dots per inch (dpi); that’s 36 pixels per point. (One point is 1/72.27 of an inch.) Proof mode does **not** produce a TFM file.

What good is proof mode, and why is it the default? Proofs are blown up copies of characters used by font designers to judge whether they like the results of their work. Naturally, proofs come first, and normal sized character production later - if you’re a font designer.

So there are two clues that proof mode is on: font files with extensions like “.2602gf” (or on DOS, “.260”), and the “failure” to produce any TFM file.

1.4 Localfont Mode

When using a stable font, we don’t want proof mode, we want our local printer’s mode. Usually, METAFONT is installed with a “localfont” assigned in the local/modes file. On our department’s Sun Network, we have assigned “localfont:=CanonCX”. We use Karl Berry’s “modes.mf”, which contains modes for many, many printers. We chose the CanonCX mode because “modes.mf” recommends it for Apple Laserwriters and HP Laserjets, which we use.

On some installations of METAFONT, if the name of a METAFONT source file is typed on the command line, then "localfont" will be used:

```
mf myfont10
```

This processes file "myfont10.mf", and should produce a GF font file, "myfont10.300gf" and a TFM file, "myfont10.tfm".

But other installations need the local font specified. On unix:

```
mf '\mode=localfont; input myfont10'
```

On MS-DOS, type it without the apostrophes:

```
mf \mode=localfont; input myfont10
```

These command lines are a bit long, very often used, and rather intolerant of mistakes (see section 2.3 below), so you might type the repetitive parts into a unix shell script or a DOS batch file, as appropriate.

1.5 Font Naming

By the way, if you plan to modify a CM font, you must give it a new name. This is an honest practice, and will avoid later confusion.

1.6 Using a New Font in TeX

To use a new font in a TeX document, select it specifically. Example: in a TeX macro file, or in a LaTeX style file, to define `\mine` as a font-selection command for "myfont10.tfm", say:

```
\font\mine=myfont10
```

Then to typeset "Mary had a little lamb," in the "myfont10" font, and then to revert to the previous font, type

```
{\mine Mary had a little lamb,}
```

1.7 Magnification (and Resolution)

Now suppose that you want myfont10 to be magnified, say to magstep 1 (magnified by 1.2), for a "jumbo" printer. Assuming that the local/modes file has a mode for the jumbo printer, you need then to run METAFONT as follows. On DOS:

```
mf \mode=jumbo; mag=magstep(1); input myfont10
```

to produce “myfile10.tfm” (again!) and a GF font, “myfile10.360”.

On unix, the csh and sh interpret semicolons, backslashes and parentheses specially, so these will have to be 'quoted'. Either

```
mf '\mode=jumbo; mag=magstep(1);' input myfont10
```

or

```
mf '\mode=jumbo; mag=magstep(1); input myfont10'
```

will do. This will produce “myfile10.tfm” and the GF font “myfile10.360gf”.

The “360” is “300 * 1.2”, indicating the magnification. A 360 dpi font can be used either as a magnification 1.2 font on a 300 dpi printer or as a normal sized font on a 360 dpi printer.

1.8 GFtoPK

TEX uses only the TFM file, which METAFONT will produce if it's in a font-making mode. (The METAFONTbook, Appendix F.) Most DVI drivers read the PK font format, but METAFONT makes a GF (Generic Font) file. So we need also to apply the “GFtoPK” utility:

```
{\sf GFtoPK} myfile10.300gf
```

to produce the wanted “myfile.300pk” (or, on DOS, “myfile.pk”) PK font.

1.9 Storing the Fonts

Now we have the fonts, where do we store them? TEX and METAFONT are compiled with default locations written in. These can be overridden by certain environment variables. The names of these variables differ between systems, but on unix they might be “TEXTFMS” and “TEXFONTS”, for example. You can find out what environment variables you now have, by typing “set”. (This works in both unix and DOS.)

If you want TEX and METAFONT to find files in the current directory, you must have “.” in the paths they use, either compiled into TEX and METAFONT, or in the environment variables that they read. (This is true for both unix and DOS.)

On the other hand, you may be content with your new font, and you may have write access to the place where most of the fonts are stored. In that case, copy your font to there. There will be a place for the TFM files, and another for the PK files. It's up to you or your local system administrator(s) to know where these directories are, because their names are very locale dependent.

1.10 The "**" prompt

Instead of typing on the command line, as most METAFONT programs will permit, you are always allowed to type:

```
mf
```

Then mf displays a "**" prompt, to which you respond with the command line arguments:

```
\mode=localfont; mag=magstep(1); input fred
```

In unix, this has the advantage that those pesky apostrophes are not needed. However, for shell scripts (and for batch files in DOS), the command line is a boon.

1.11 Using a different base

Suppose that you made another base, "joe.base", to replace plain. Typing

```
mf &joe
```

ejects the "plain" base, and loads the "joe" base.

On some systems, the "cm" base, for making the Computer Modern fonts is loaded in this way:

```
mf &cm
```

On other systems, a separate program, "cmmf", exists, which automatically loads "cm.base". In "web2c", cmmf is a link to the same program as mf, and the program reads its own command line name to determine which base to load first; thus, "cmmf" loads "cmmf.base", which is web2c's usual name for the CM base.

2 What went wrong?

The complexity of wrong things far exceeds that of things intended.

Small typing errors are so common, and yet undocumented (why are common mistakes not documented?), that I thought I'd list several that have tripped me up on innumerable occasions. After all, why reinvent the car crash?

References for some of these points:

The METAFONTbook, chapter 5, "Running METAFONT", contains instructive examples, and supposedly "dangerous", but actually basic and useful, notes.

In that chapter, and in chapter 7, "Recovery from Errors", KNUTH discusses the diagnosis of METAFONT's error messages. I find this perhaps the hardest part of the book - if not of using METAFONT.

2.1 Big fonts, but unwanted

Recently, I found myself accidentally producing fonts with extensions like "3122gf". How?

MF will take **anything** as an excuse to revert to **proof mode**. The "3122" is a magstep 1 proof mode. It's

```
(1.2)^1 * 2601.72  = 3122.16  dots per inch.
```

My intention was for METAFONT on a PC to use an HP Laserjet mode in place of proof mode. However, METAFONT's command line resembles the law: every stroke of the pen is significant. What I had forgotten was that on my setup, "localfont" must be explicitly requested.

EmTEX's METAFONT, with plain.mf, defaults to proof mode. However, I usually want a local printer's font-making mode. So to process "pics.mf" correctly, I need to say:

```
mf '\mode=localfont; input pics'
```

2.2 Consequences of Some Typing Errors on METAFONT's command line

Consider a source file "pics.mf" that contains "mag=1200/1000;", so it is automatically scaled by 1.2 (ie, by magstep 1). If the target printer has 300 dpi, then a 360 dpi GF font is wanted.

Here is the gist of what happens for various typing errors, when using emTEX's "mf186" on a 286 PC to process "pics.mf".

1. `mf186` ⟹ METAFONT will keep prompting for arguments:

   ```
   **
   ```

   We can type the contents of the command line here; for example, I can now type "pics". In fact, even if you use the command line, the .log ("transcript") file shows METAFONT echoing its interpretation of the command line to a ** prompt.

2. `mf186 pics` ⟹ proof mode:

   ```
   ! Value is too large (5184)
   ```

   No tfm is produced, and the gf file has resolution 3122 dpi. (3121.72 dpi, to be precise.)

3. `mf186 mode=localfont; input pics` ⟹ misinterpretation:

   ```
   ! I can't find file 'modes=localfont.mf'.
   ```

   So, "modes" needs that backslash, otherwise mf thinks it's the start of a source font's filename. Backslash ("\") and ampersand ("&") are escapes from this standard interpretation by METAFONT of the first argument.

4. `mf186 \mode=localfont input pics` ⟹ weird effect:

   ```
   >> unknown string mode_name1.2
   ! Not a string
   <to be read again>
                    ;
   mode_setup-> ...ode)else:mode_name[mode]fi;
   l.6 mode_setup
                ;
   ```

   Wow! What a difference a semicolon can make!

5. `mf186 \mode=localfont pics` ⟹ almost nothing happens:

```
** \mode=localfont pics

*
```

There's the echo I mentioned. From the lack of activity, pics evidently needs to be "input".

6. `mf186 \mode=localfont; pics` ⟹

   Same as 5.

   So, yes, when the mode is specified, we need "input" before "pics".

7. `mf186 &plain \mode=localfont; input pics` ⟹

   Works.

   Just as without the "&plain", it writes a GF file, "pics.360gf", which is correct. (DOS truncates the name to "pics.360".) So, redundancy seems okay. Does it waste time, though?

2.3 Finding the Fonts

Finding the fonts (*.mf, *.tfm, *.gf, *.pk) trips up TEX, METAFONT, GFtoPK and the output drivers continually. "pics.tfm" needs to be put where TEX will look for TFMs, so I needed to ensure that "." was in the appropriate path environment variable. Similarly for the MF, GF and PK font files.

Environment variables can be tricky. For example, emTEX's font-making automation program "mfjob" cannot make fonts in the current directory unless both "." and ".." are added to MFINPUT. This was not documented.

Also, some popular TEX output drivers, such as the emTEX drivers on DOS and OS/2, and Tomas Rokicki's "dvips", which has been ported to many systems, make missing fonts automatically - provided that they can find the necessary mf source files. Again, making fonts in the current directory can require some tweaking.

On unix, dvips uses a shell script "MakeTeXPK" (on DOS, a batch file "MakeTeXP"), which creates a temporary directory, which it then changes to. This can cause METAFONT not to find mf font sources in what **used** to be the current directory. Gert W. Bultman (bultman@dgw.rws.nl) has suggested adding a line to MakeTeXPK:

```
set MFINPUTS=${MFINPUTS}:`pwd`;  export MFINPUTS
```

to add the current directory to the search path, before the change to the temporary directory.

3 Conclusion

METAFONT, like TeX and many another "portable" program of any complexity, merit the warning: "Watch out for the first step."

It's my hope that a document like this may help to prevent domestic accidents involving METAFONT, and so contribute to making the task of using and designing meta-fonts an enjoyable one. My brief experience with METAFONT suggests that it can be so.

All the Best!