METAFONT for Beginners

Third Draft, Revision 'T'

(15:47 GMT +10:00 Sat 26 June 1993)

Geoffrey Tobin (ecsgrt@luxor.latrobe.edu.au)

Contents

Scope

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

Where you can obtain this file

METAFONT for Beginners can be obtained by ftp from the CTAN Comprehensive TeX Archive Network sites:

```
ftp.uni-stuttgart.de : soft/tex
ftp.tex.ac.uk : pub/archive
ftp.shsu.edu : tex-archive
```

in the

```
documentation
```

subdirectory, as the file

```
metafont-for-beginners.tex
```

Also from:

```
niord.shsu.edu : faq/faq.mf
```

For those without ftp, it can be received by email from:

```
fileserv@shsu.edu
```

by sending the one-line message:

```
sendme faq.mf
```

Reference

The METAFONT*book*, 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, except where indicated otherwise, on the 1986 edition.[1]

[1] Opinion: I actually enjoy reading *The* METAFONT*book*, whereas *The TeXbook* confuses me no end.

Acknowledgements

Additions and corrections were kindly contributed by:

Bill Alford (`bill@phys.anu.edu.au`),
Tim A. H. Bell (`bhat@mundil.cs.mu.oz.au`),
Karl Berry (`karl@cs.umb.edu`),
Gert W. Bultman (`bultman@dgw.rws.nl`),
Anita Zanolini Hoover (`anita@ravel.udel.edu`),
Berthold K. P. Horn (`bkph@kauai.ai.mit.edu`),
Michał Jaegermann (`ntomczak@vm.ucs.ualberta.ca`),
and
David Kastrup (`dak@pool.informatik.rwth-aachen.de`).

Typesetting was initiated by

Yannis Haralambous (`yannis@gat.citilille.fr`).

Motivation

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

1 What is METAFONT?

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 a syntax apparently derived in part from the `Algol`[2] family of programming languages, of which C, C++, Pascal and Modula-2 are members.

The input can be interactive, or from a source file. METAFONT source files are usually suffixed '`.mf`'.

METAFONT sources can utilize scaling, rotation, reflection, skewing and shifting, and other complex transformations in obvious and intuitive ways. But that is another story, told (in part) by *The* METAFONT*book*.

METAFONT's bitmap output is a GF (*generic font*) file. This may be compressed to an equivalent PK (*packed*) font by the auxiliary program GFtoPK.

Why doesn't METAFONT output PK fonts directly? Firstly, Tomas Rokicki had not invented PK at the time Donald E. Knuth was writing METAFONT. Secondly, to change METAFONT now would be too big a change in Knuth's opinion. (Knuth is a very conservative programmer; this fact is a two-sided coin.)

[2]Around 1960, Donald Knuth worked as an `Algol` compiler designer.

GF and PK files are suffixed '.*gf' and '.*pk' respectively, where, in a typical UNIX installation, the '*' stands for the font resolution. (Resolution will be explained below.) MS-DOS truncates file name suffixes to three characters, so a font suffix '.1200gf' becomes '.120' — beware of this!

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 making the internal variable '`fontmaking`' positive. Most output device modes (see subsection 3.4 below) do this.

Remember that TEX reads only the TFM files. The *glyphs*, or forms of the characters, as stored in GF or PK font files, do not enter the picture (I mean, are not read) until the DVI drivers are run.

TEX can scale TFM files. Unfortunately, bitmaps such as GF and PK are not scalable. However, METAFONT files can be compiled into fonts of arbitrary scale by METAFONT, even by non-programmers — see subsection 4.5.

Incidentally, properly constructed TFM files are device-independent, so running METAFONT with different modes normally produces the identical TFM. Dimensions in TFM files are specified to METAFONT in device independent 'sharped' dimensions (commonly suffixed by #), where a value of 1 corresponds to the dimension of `1pt` (typographical point). Most of METAFONT's calculations are done with (resolution and device dependent) pixels as units. Care must be taken by font designers to *always* calculate unsharped dimensions from sharped ones, and never the other way round, so as to keep roundoff errors or similar effects from influencing the TFM files to depend on resolution or device. Although type quality will be influenced only in minuscule ways, this is one of the more common reasons for checksum errors reported by printer drivers.

More detailed descriptions of TFM and GF files, and of *proof* mode, are found in Appendices F, G, and H, respectively of *The* METAFONT*book*.

The TUG DVI *Drivers Standard, Level 0*, draft 0.05, includes precise definitions of the file structure of TFM metrics and of GF and PK bitmap fonts. That document is obtainable from the TEX archive at

```
ftp.uni-stuttgart.de
```

where it is currently found as the several files in the directory

```
soft/tex/dviware/driv-standard/level-0
```

Related information is contained in the documents in the 'sister' directory

```
soft/tex/dviware/driv-standard/papers
```

2 Getting METAFONT's Attention

2.1 Typing at METAFONT's '**' prompt

If you type the name of the METAFONT program alone on the command line:

```
mf
```

then `mf` displays a '`**`' prompt, which 'is METAFONT's way of asking you for an input file name'. (See *The* METAFONT*book*, Chapter 5: 'Running METAFONT'.) Thus, to process a METAFONT file named `fred.mf`, you may type:

```
fred
```

A backslash ('\') can also be typed here. This causes all subsequent commands at the prompt line to be interpreted as in a METAFONT file. (Concerning the backslash, see *The* META FONT*book*, Chapter 20: 'More About Macros', pages 179 and 180 in the 1986 edition.) Thus we can respond to the `**` prompt with:

```
\ input fred
```

or even:

```
\ ; input fred
```

The backslash is useful because certain commands are often executed before a METAFONT file is input. In particular, quality printing (see subsection 3.4 below) requires the METAFONT command `mode`, and output magnification (subsection 4.5) employs the `mag` command. For example:

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

To read MS-DOS pathnames at the `**` prompt, this satisfies METAFONT:

```
\input \seldom\fred.mf
```

as does:

```
d:\seldom\fred.mf
```

2.2 Typing on the Command Line

Most METAFONT implementations permit you to type METAFONT commands on the command line, instead of at the `**` prompt. (Rather, it is automatically passed to that prompt.)

On MS-DOS, type commands as at the `**` prompt:

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

On UNIX, the C (`csh`) and Bourne (`sh`) command shells interpret semicolons, backslashes and parentheses specially, unless they are 'quoted'. So, when typing those characters as part of instructions to METAFONT on the UNIX command line, it's wise to accustom yourself to protecting them with *apostrophes*:

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

If `localfont` makes fonts for a 300 dots per inch (dpi) device, this should produce a TFM file, '`myfont10.tfm`', and a 300 dpi GF font file, '`myfont10.300gf`'. Almost all of the following will presume a 300 dpi device, and other resolution devices will have appropriately different font file names.

These command lines are a bit long, very often used, and rather intolerant of mistakes (see subsection 6.2 below), so you might type the repetitive parts into a UNIX shell script or an MS-DOS batch file, as appropriate.

In UNIX, the `**` prompt has the advantage that those pesky apostrophes are not needed. (Indeed, those apostrophes are always wrong at the `**` prompt — METAFONT doesn't understand them. It would not understand them on the command line either—it's just that the shell does not hand them over to METAFONT.) However, for shell scripts (and for batch files in MS-DOS), the command line is a boon.

For the `Macintosh`, which is not command line based, Tim BELL reports that one port of METAFONT (by Timothy MURPHY `<tim@maths.tcd.ie>` 22 January 1993) simulates the command line within the program (using a special THINK C library written just for that). But what you type goes through some string processing, so you need double '`\`'s. Thus your example line reads:

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

2.3 '`Please type another input file name:` '

When METAFONT cannot find the main source file, it doesn't quit. For example, when I typed `mf fred`, METAFONT said:

```
This is METAFONT ...
**fred
! I can't find file 'fred.mf'.
<*> fred

Please type another input file name:
```

The usual program interrupts (eg, Control-C) don't work here, and the '`Please type ...`' prompt does not understand METAFONT commands: it will read only the first word, and insist on interpreting this as a file name.

Beginners faced with this often wonder how to avoid an endless loop or a reboot, or try to think of a METAFONT file that they do have in METAFONT's path. In the latter case, the canonical name to use is '`null`', standing for the file '`null.mf`'.

In fact, the solution is much easier: on the systems that I have tried, a simple end of file marker ('control-Z' in MS-DOS, 'control-D' in UNIX) stops METAFONT in its tracks:

```
! Emergency stop.
<*> fred

End of file on the terminal!
```

3 Base files

In versions 2.7 and 2.71, the METAFONT language contains 224 (previous versions had fewer) primitives, which are the commands preceded by an asterisk in the Index (Appendix I) to *The* METAFONT*book*. 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 internal tables* that METAFONT loads faster than it loads the original METAFONT source files. Thus, they are closely analogous to TEX's *format* files.

3.1 The `plain` base

The `plain` base provides the commands that *The* METAFONT*book* describes. (See Appendix B of *The* METAFONT*book*, 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[3] the `plain` base. This is usually called `plain.base`, or sometimes only (see subsection 3.3 for why this works) `mf.base`, although for those systems concerned (such as UNIX), both file names should really be present.

EmTEX for MS-DOS calls the plain base `plain.bas`, due to filename truncation.

3.2 Loading a Different Base

Suppose that you have a base named `joe.base`. Typing

```
mf &joe
```

or (on unix, where we must either quote or escape the ampersand)

```
mf \&joe
```

or responding

```
&joe
```

to the `**` prompt, omits loading `plain` base, and loads the `joe` base instead. Typically, however, the `joe.mf` file which originally produced the `joe` base will have included `plain.mf`, because working without the `plain` base macros would be too cumbersome. (Refer to *The* METAFONT*book* (1986), Chapter 5: 'Running METAFONT', page 35, 'dangerous bend' number two.)

The '`cm`' base, for making the COMPUTER MODERN fonts, can be loaded in that way:

```
mf &cm
```

Remember to quote the ampersand under UNIX!

[3]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.

3.3 The Linkage Trick

On systems such as UNIX where programs can read their own command line name, and where files may be linked to two or more names, then programs can modify their behavior according to the name by which they are called. Many UNIX TEX and METAFONT installations exploit this in order to load different *format* and *base* files, one for each of the various names to which TEX and METAFONT are linked. Such installations can often be recognised by the presence of the executable '`virmf`' in one of the directories in the `PATH`.

For example, if a base file called '`third.base`' resides where METAFONT can find it (see section 4.8 below), then `virmf` can be linked to `third`. In UNIX, a *hard link* is formed by

```
ln virmf third
```

On systems supporting *symbolic links*, you should make all of these links symbolic, rather than hard, or else you will have to redo them every time you install a new copy of `virmf`; see below. In UNIX, this is done by

```
ln -s virmf third
```

Normally one wants `mf` to load the `plain` base, so in such installations one links `plain.base` to `mf.base`:

```
ln plain.base mf.base
```

Again, you'd best make that link symbolic. This comment applies for the rest of this section as well.

As another example, take the '`cm`' base. In web2c:

```
ln virmf cmmf
ln cm.base cmmf.base
```

so that '`cmmf`' automatically loads '`cm.base`'.

This applies equally to TEX, which is why `tex` and `latex` are then links to `virtex`, `tex.fmt` is a link to `plain.fmt`, and `latex.fmt` is a link to lplain.fmt:

```
ln virtex tex
ln plain.fmt tex.fmt

ln virtex latex
ln lplain.fmt latex.fmt
```

Karl BERRY's web2c distribution for UNIX uses this '*linkage trick*'.

If you used symbolic links, you can laugh off the following

WARNING: This linkage is convenient, but watch out during updates! If `mf.base` is a *hard link* to `plain.base`, then replacing `plain.base` with its new version severs the link: `mf` will still load `mf.base`, but it will be the old version! The proper procedure is to remove the old `mf.base`, and relink. On UNIX:

```
rm mf.base
ln plain.base mf.base
```

On most UNIX systems, `ln -f` will automatically remove the second file (if present) — in this case, mf.base — before linking.

Alternatively, `web2c` will update '`plain.base`' (and '`plain.fmt`', and so on) for you, if you tell `web2c`'s `Makefile` to

```
make install
```

Symbolic links, on systems that have them, are probably the best method of handling updates, at least when doing them manually. (Consult your system administrator for details.)

3.4 Making a Base; the Local Modes file

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 output device a font-making *mode*, containing a description of some refinements that must be made in order to produce good-looking output. For instance, how to make the characters just dark enough, and how to make diagonal lines come out sharply.

If you want to make a base, you need a variant of the METAFONT program called '`inimf`'. (See *The* METAFONT*book*, 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
```

4 Fonts

4.1 Proof Mode

The purpose of METAFONT is to make fonts. For æsthetically pleasing PK bitmaps, the correct device mode must be selected.

An obstacle to beware of is that `plain` METAFONT uses *proof* mode by default. (*The* META FONT*book*, 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 MS-DOS, '`.260`'), and the 'failure' to produce any TFM file.

On some systems, such as X11, a third clue is that the proof font may be drawn on the screen — it's so large, you can't miss it!

4.2 Localfont Mode

When using a stable font, or when testing the output of a new font, we *don't* want proof mode, we want our local output device'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`'[4], which contains modes for many, many devices. We chose the `CanonCX` mode because '`modes.mf`' recommends it for Apple Laserwriters and HP Laserjet II printers, which we use.

To process a METAFONT source file named '`myfont10.mf`' for the most usual local device, specify the local mode to `mf` before inputting the font name:

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

This should produce a GF font file, '`myfont10.300gf`' ('`myfont10.300`' in MS-DOS), and a TFM file, '`myfont10.tfm`'.

4.3 Font Naming

By the way, if you modify an existing, say a COMPUTER MODERN (CM), font, you must give it a new name. This is an honest practice, and will avoid confusion.

4.4 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,}
```

Note, however, that this will not change the line spacing parameters of TEX as well. If your lines appear a little too cramped and unevenly spaced vertically, it is very probable that you need to increase `\baselineskip`. For LATEX users, a simple remedy is to just select a larger font before your own. Also, end your paragraph by an empty line or a `\par` command before the closing brace, or your line spacing changes will be cancelled before the paragraph has a chance of being typeset.

[4] Available at `ftp.cs.umb.edu` in directory `pub/tex`

4.5 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 may then run METAFONT with the following three commands:

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

to produce '`myfile10.tfm`' (again!) and a GF font, '`myfile10.360gf`'. On MS-DOS, the file names will be truncated; for example, '`myfile10.360`'.

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.

Note, however, that the METAFONT language includes special hints for each output device which clue METAFONT as to the reactions of the output device to pixel-sized minuscule changes.

So for highest quality, you would not even want to mix the fonts for two 300 dpi printers, unless they share the same mode and most probably the same print engine.

4.6 GFtoPK

TeX uses only the TFM file, which METAFONT will produce if it's in a font-making mode. (*The* METAFONT*book*, 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:

```
gftopk myfile10.300gf
```

to produce the wanted '`myfile.300pk`' (or, on MS-DOS, '`myfile.pk`') PK font.

4.7 Storing the Fonts

Now we have the fonts, where do we store them? TeX, METAFONT and the various driver programs 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, for example, be 'TEXFONTS' for the TFM files, and either 'PKFONTS' or 'TEXPKS' (or both of those) — before searching 'TEXFONTS' — for PK fonts. You can find out what environment variables you now have by typing '`set`' in MS-DOS and '`env`' in the Bourne shell, sh, in UNIX. In the UNIX C shell, `csh`, type '`setenv`'.

Michał JAEGERMANN notes that on a 'virgin' installation — in which everything is in default directories and no environment variables have yet been set — that won't succeed. Presumably we're talking to system installers now. So, as a first resort:

Read The Manual.

As a last resort, one can discover default values and environment variable names by using a command like UNIX's `strings` on the executable files. For instance:

```
strings -6 /bin/virmf | less
```

(Use 'more' or 'pg' for paging, if 'less' is not available.) Seeking 6-letter names is about right, as "`TEXPKS`" has six letters, while `strings`' default of four collects too much random noise. Environment variables are usually in upper case, and their names strongly hint at their purposes. Default locations may be discovered by looking for path name strings.

Using this advice may show some undocumented names. If you have the program sources, you may check their purpose. Otherwise, not to worry, the important ones should be self-evident. As an illustration, here are some environment variable names found by applying "strings -6" to ROKICKI'S `dvips`:

```
* DVIPSHEADERS
  HOME
* PKFONTS
  PRINTER
  TEXCONFIG
  TEXFONTS
  TEXINPUTS
* TEXPACKED
* TEXPICTS
  TEXPKS
  VFFONTS
```

The four starred names are not documented by the `dvips` manual (for version 5.484). In BERRY'S `dvipsk` (version 5.515), `PKFONTS` and `TEXPICTS` *are* documented, while `DVIPSHEADERS` and `TEXPACKED` are *not used*.

If you want TEX and METAFONT to find files in the current directory (as you almost certainly do!), then one way is to put '`.`' into their search paths. (Both UNIX and MS-DOS accept the . notation for the current directory.) Default search paths are compiled into TEX and METAFONT, but users can customise the environment variables (see subsection 4.8) that the programs read, to override the defaults.

METAFONT (as illustrated in section 2 above), as well as the DVI drivers, can also be given full path specifications for input files. (On most systems, so can TEX, but, as Berthold K. P. HORN (`bkph@kauai.ai.mit.edu`) has observed, MS-DOS poses the problem that the backslash '\' used in MS-DOS path names is very special in TEX input. However, I'll leave solving that one to the TEXackers.)

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.

4.8 Environment Variables for emTEX and web2c

Environment variables often cause confusion, as they vary unpredictably — sometimes subtly, sometimes widely — between systems.

EmTEX for MS-DOS and web2c for UNIX are two popular distributions of TEX, METAFONT, and associated programs. It's worthwhile therefore to compare their environment variables.

Firstly, the variables used leading up to the production of the DVI file:

TEX, BIBTEX, METAFONT and MFjob		
Seeking	emTEX	web2c
TEX Pool file	TEXFMT, BTEXFMT	TEXPOOL
TEX Formats	TEXFMT, BTEXFMT	TEXFORMATS
TEX Inputs	TEXINPUT	TEXINPUTS
TEX Font Metrics	TEXTFM	TEXFONTS
BIBTEX bst	TEXINPUT	BSTINPUTS, TEXINPUTS
BIBTEX bib	BIBINPUT	BIBINPUTS
METAFONT Pool	MFBAS, BMFBAS	MFPOOL
METAFONT Bases	MFBAS, BMFBAS	MFBASES
METAFONT Inputs	MFINPUT	MFINPUTS
MFjob Inputs	MFJOB	—

The second table compares the environment variables used by emTEX's DVI drivers with those for Tomas ROKICKI's portable PostScript driver, dvips.

DVI Drivers		
Seeking	emTEX Drivers	dvips
DVI files	DVIDRVINPUT	*current directory*
PK Fonts	DVIDRVFONTS	TEXPKS, PKFONTS
Bitmap Graphics	DVIDRVGRAPH	—
Virtual Fonts	set by /pv option	VFFONTS, TEXFONTS
MakeTeXPK	—	MAKETEXPK
config.ps	—	TEXCONFIG
PS files	—	TEXINPUTS

Where two or more variables are listed together, they are searched from left to right. For example, dvips seeks PK fonts first in TEXPKS, then in PKFONTS. By the way, if no PK fonts can be found, then dvips uses the TFM files to determine spacing, and leaves the characters blank.

Karl BERRY's dvipsk, a variant of dvips, seeks PK fonts in whichever *one* of TEXPKS, PKFONTS and TEXFONTS is set and of the highest priority. If a font cannot be found there, then the compile-time system default paths are searched; any lower priority font path environment variables are ignored. This may also be the behavior of Rokicki's dvips, but readers are encouraged to discover the truth for themselves.

MFjob and MakeTeXPK have a similar function: to create PK fonts from METAFONT files. When PK fonts are missing, but the METAFONT font sources are available, MFjob can be called by recent versions (1.4r and above) of the emTEX drivers to create the missing fonts. MakeTeXPK is called by dvips for the same purpose.

Karl BERRY has recently released `web2c 5.851d` and a new version of `dvipsk`[5] in which `MakeTeXTFM, MakeTeXTeX, and MakeTeXMF` are called by `dvipsk` for missing TFM, TEX, and METAFONT files, respectively. `MakeTeXTFM`, like `MakeTeXPK`, can call METAFONT. Design of `MakeTeXTeX` and `MakeTeXMF` are up to the user's imagination — Karl says that one possibility is to employ `ftp`.

5 Some Limitations of METAFONT

METAFONT contains some builtin limitations, some obvious, others less so.

Parts of the following list are most useful to budding programmers, though casual users may wish to read it to learn whether an error message produced by somebody else's METAFONT file is very serious or not.

1. All valid numbers are strictly less than 4096.
2. *The* METAFONT*book*, in 'Appendix F: Font Metric Information', warns of one limitation that I've met when processing some fonts.

   'At most 15 different nonzero heights, 15 different nonzero depths, and 63 different nonzero italic corrections[6] may appear in a single font. if these limits are exceeded, METAFONT will change one or more values, by as little as possible, until the restriction holds. A warning message is issued if such changes are necessary; for example

   ```
   (some charht values had to be adjusted by as much as 0.12pt)
   ```

   means that you had too many different nonzero heights, but METAFONT found a way to reduce the number to at most 15 by changing some of them; none of them had to be changed by more than 0.12 points. No warning is actually given unless the maximum amount of perturbation exceeds $\frac{1}{16}$ pt.'

   Every correct implementation of METAFONT will adjust character box dimensions by the same amount, giving the same TFM files, so we ignore small perturbations in other people's fonts. When designing your own fonts, however, I think it's courteous to keep within the limits, so as not to worry inexperienced users.
3. In the `addto` picture command, `withweight` only accepts values that round to `-3`, `-2`, `-1`, `+1`, `+2`, or `+3`. To obtain other pixel weights, you can apply further `addto` commands.
4. The memory size of the version of METAFONT you use is an evident, implementation dependent restriction, but it may be, as in TeX, that memory is not enough simply because, if you'll pardon my saying so, some of your coding may be seriously inefficient or logically invalid.

[5] Both available at `ftp.cs.umb.edu` in directory `pub/tex`

[6] Respectively, `charht`, `chardp` and `charic` values.

6 What Went Wrong?

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

References for some of the subsequent points:

The METAFONT*book*, chapter 5, 'Running METAFONT', contains instructive examples, and supposedly 'dangerous', but actually basic and useful, notes.

In that chapter, and in chapter 27, '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.

Incidentally, METAFONT's error messages are contained in an ASCII file called '`mf.pool`'. Reading the `pool` file can be entertaining.

6.1 Big fonts, but Unwanted

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

METAFONT *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  = 3122.164  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'
```

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

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?

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. (Ampersand is in fact only a temporary escape, as METAFONT resumes the `mf` filename prompting attitude as soon as a base is read.)

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. (MS-DOS truncates the name to '`pics.360`'.) So, redundancy seems okay. Does it waste time, though?

6.3 Finding the Fonts

Finding the fonts (`*.mf`, `*.tfm`, `*.gf`, and `*.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 METAFONT, GF and PK font files.

Environment variables can be tricky. For instance, 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 MS-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 METAFONT source files. Again, making fonts in the current directory can require some tweaking.

6.4 `MakeTeXPK`

On UNIX, when fonts are missing, `dvips` calls a Bourne shell script, '`MakeTeXPK`', which creates a temporary directory, which it then changes to, before calling METAFONT to make the missing fonts. The change of directory can cause METAFONT not to find font sources lying in what **used** to be the current directory.

Gert W. BULTMAN (`bultman@dgw.rws.nl`) has suggested the following modification to `MakeTeXPK`:

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

to add the current directory to the search path, *before* the change to the temporary directory:

```
cd $TEMPDIR
```

Michał JAEGERMANN (`ntomczak@vm.ucs.ualberta.ca`) has pointed out that:

'This will not work very well in a situation when the `MFINPUTS` variable is not set, and you rely instead on METAFONT files being in a default location. The problem is that in such a situation, after an execution of the line above, you will end up with ONLY your 'current working directory' in the `MFINPUTS` path, [which still leaves you without access to the standard METAFONT files].

'For the Bourne shell, `sh`, this line should rather read somewhat[7] like:

```
MFINPUTS=`pwd`:${MFINPUTS-/usr/lib/mf/inputs}
export MFINPUTS
```

which gives you a fallback position. Of course,

```
/usr/lib/mf/inputs
```

[7]gt: I've separated this into two (valid, unix Bourne shell) lines, to fit into the text width of this document.

should be replaced by a default value for the `MFINPUTS` path.

'This problem is highly likely to affect budding METAFONT hackers on NeXT, for example.'

Michał's suggestion gives priority to METAFONT files in the directory that is current when MakeTeXPK is called, which is the usual preference. In `sh`, the '`$A-B`' construction has the value of `A`, if `A` is defined, and the value of `B`, otherwise.

Karl BERRY advises that for `web2c 5.851c` and above, a leading or trailing colon in a path is replaced by the compile-time default path. For `web2c` he suggests:

```
if test -z "$MFINPUTS"; then
  MFINPUTS=`pwd`:
else
  MFINPUTS=`pwd`:$MFINPUTS:
fi
```

Test these ideas on your system, to see what is most applicable.

Incidentally, on MS-DOS, `dvips` calls a batch file, '`MAKETEXP.BAT`', but, in the MS-DOS versions I've seen, this lacks the change to a temporary directory that causes the problem that occurs both in the UNIX versions of `dvips` and in emTEX's MFjob.

6.5 Strange Paths

METAFONT satisfactorily fills simple closed curves, like '`O`' and '`D`', but filling a figure eight, '`8`', causes a complaint:

```
Strange path (turning number is zero)
```

because METAFONT's rules for distinguishing inside from outside might or might not give what you want for an '`8`', as there is more than one conceivable answer. You can use the 'positive turning rule' for all cases, and also turn off complaints, by setting

```
turningcheck := 0;
```

Chapter 13: 'Drawing, Filling, and Erasing', and Chapter 27: 'Recovery from Errors', discuss `strange paths` in greater depth.

Sometimes, when making a perfectly valid font, but in *low* resolutions, as for previewers (eg, VGA has 96 dpi), one may get flak about a '`Strange path`' or '`Not a cycle`' or something similar. Don't be alarmed. Fonts for previewing will still be OK even if not perfect.

Consequently, it is an idea to make low resolution fonts in METAFONT's `nonstopmode`.

Examples of fonts that give messages of this nature are the pleasant Pandora, and — from memory — the commendable Ralf Smith's Formal Script (`rsfs`). Everything is fine at higher resolutions.

Mind you, some fonts provoke sporadic (that is, design size dependent) strange path messages at 300 dpi (phototypesetter users would consider that low resolution), yet the printed appearance showed no visible defect.

Why do strange paths occur? One cause is rounding error on relatively coarse grids.

To summarize, if your viewed or printed bitmaps are fine, then you are OK.

7 METAFONT Mail List

Since 10 December 1992, there has been an e-mail discussion list for METAFONT, created:

1. as a means of communication between hooked METAFONTers;
2. as a way to bring the "rest of us" closer to them;
3. as a means to get quick and efficient answers to questions such as:
   - why do I always get a ".2602gf" file?
   - what is a "strange path", and what can I do to avoid it?
   - is there a way to go from METAFONT to PostScript and vice-versa?
   - where can I find a Stempel Garamond font written in METAFONT?
   - what is metaness?
4. and finally, as a first step to encourage people to undertake METAFONTing, and start a new post-Computer Modern era of METAFONT!

To subscribe to this list, send the following two lines to "`listserv@ens.fr`" on the Internet:
SUBSCRIBE METAFONT <Your name and affiliation>
SET METAFONT MAIL ACK

The address of the list is "`metafont@ens.fr`" (at the notorious Ecole Normale Superieure de Paris). Owner of the list is Jacques BEIGBEDER ("`beig@ens.fr`"), coordinator is Yannis HARALAMBOUS ("`yannis@gat.citilille.fr`"). Language of the list is English; intelligent mottos are encouraged.

8 Conclusion

METAFONT, like TEX and many another 'portable' program of any complexity, merits the warning: '*Watch out for the first step*'.

I 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!