

METAFONT for Beginners

Geoffrey Tobin

ecsgrt@luxor.latrobe.edu.au

Scope

This is not a tutorial on METAFONT.¹ It is an attempt to describe how some of the pitfalls in running the program may, hopefully, be avoided.

It is a common experience to have initial (and medial and final) difficulty with running METAFONT, and not all ‘T_EXnicians’ are as familiar with METAFONT as they are with T_EX. 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 T_EX, its viewers, printer drivers, and related programs. It interprets a drawing language with a syntax apparently derived in part from the Algol 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 METAFONTbook*.

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 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.)

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 T_EX to typeset a font it needs a TFM (*T_EX 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 below) do this.

Remember that T_EX 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.

T_EX 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.

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. Note that the only way to be sure that a TFM file is device-independent is to create the font in different modes and compare the resulting TFM’s, perhaps using *tftopl*.

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

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

¹This article is a trimmed version of the tutorial file made available on T_EX archives; a current full version can be obtained from the *documentation* directory on any CTAN archive (see elsewhere in this issue for more details of CTAN).

The METAFONTbook, 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 METAFONTbook*, 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 requires the METAFONT command `mode`, and output magnification 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, command shells typically 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 is 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 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 is 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.

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 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 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 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 loads² the `plain` base. This is usually called `plain.base`, or sometimes only `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 METAFONTbook* (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!

3.3 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 recognized 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, 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.)

²There 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.

3.4 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 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
```

4 Fonts

4.1 Proof Mode

The purpose of METAFONT is to make fonts. For aesthetically 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 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 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 is 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`’³, 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 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.

4.5 GFtoPK

TeX uses only the TFM file, which METAFONT will produce if it is 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:

```
gftopk myfile10.300gf
```

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

³Available at `ftp.cs.umb.edu` in the `pub/tex` directory.

4.6 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'.

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 that the programs read, to override the defaults.

METAFONT as well as the DVI drivers, can also be given full path specifications for input files.

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 is up to you or your local system administrator(s) to know where these directories are, because their names are very locale dependent.

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 METAFONTbook*, 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 corrections⁴ 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 is 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.

6 What Went Wrong?

The complexity of wrong things far exceeds that of things intended. *The METAFONTbook*, 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 is

$$(1.2)^1 * 2601.72 = 3122.164 \text{ dots per inch.}$$

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.

Em \TeX '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'
```

⁴Respectively, charht, chardp and charic values.

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 \Rightarrow 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 \Rightarrow 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 \Rightarrow misinterpretation:

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

So, 'modes' needs that backslash, otherwise METAFONT thinks it is 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 \Rightarrow weird effect:
>> unknown string mode_name1.2
! Not a string
<to be read again>
;
mode_setup->
...ode)else:mode_name[mode]fi;
1.6 mode_setup
;

Wow! What a difference a semicolon can make!

5. mf186 \mode=localfont pics \Rightarrow 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 \Rightarrow Same as 5.

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

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

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 T_EX, METAFONT, *GFtoPK* and the output drivers continually. 'pics.tfm' needs to be put where T_EX will look for T_FM's, 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 T_EX 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 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 (photo-typesetter 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>
SET METAFONT MAIL ACK
```

The address of the list is `metafont@ens.fr` (at the notorious Ecole Normale Supérieure 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 T_EX 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!