This is a very basic look at using METAFONT and gnuplot to make figures for use in LaTeX. I am using Linux, but the same process ought to work for other LaTeX environments; indeed, that ought to be one of its strengths.

1. First, on firing up gnuplot, type this:

   ```
   gnuplot> help set term mf detailed
   ```

   and copy the result to a text file (I just used cut and paste off the screen). It is 90% of what you need, maybe more if you are more knowledgeable than me.

2. Then type something like this:

   ```
   gnuplot> set xrange [0:3.14159]
   gnuplot> plot sin(x), cos(x)
   gnuplot> set term mf
   gnuplot> set output "test2.mf"
   gnuplot> replot
   gnuplot> exit
   ```

3. Then edit the resulting file `test2.mf` and read the bit that says:

   ```
   %Include next eight lines if you have problems with the mode on your
   %system..
   proofing:=0;
   fontmaking:=1;
   tracingtitles:=0;
   pixels_per_inch:=600;
   blacker:=0;
   fillin:=.2;
   o_correction:=.6;
   fix_units;
   ```

   and think about uncommenting bits. I uncommented all of it, as you can see.

4. Then type something like:

   ```
   $ mf '\mode=ljfour; input test2'
   $ gftopk test2.600gf
   ```

   and have a look at the files you have created; you ought to see `test2.600pk/gf` and `test2.tfm`.

5. If you like at this point you can try:

```
$ gftodvi test2.600gf
$ xdvi test
```

and you should see the picture appear in the `.dvi` file.

6. Now, you want your LaTeX file to be able to use your figure. It does so as a font rather than as a graphic. Since LaTeX (and TeX) can handle fonts natively but uses external packages to import other graphic formats, this is a strength of the approach, though less important in these days of very powerful machines.

   Create a simple `.tex` file, something like this one you are reading now. Put the below in the preamble (though it can go anywhere you like as long as it's *before* you want to use the figure):

   ```
   \font\gnufigs=test2
   ```

   This tells the typesetting program that you want to use a font called 'gnufigs', and that the information about this font is in files called `test2.tfm` and `test2.XXXpk` where `XXX` is the dots-per-inch (600 in the example above).

   The picture is character 0 (zero) in your 'font'. Indeed, if you've only done as I suggest here, it is the only character. To insert it you just type:

   ```
   {\gnufigs \char0}
   ```

   somewhere in your file. You can embed this in a `figure` environment if you want the usual LaTeX cross referencing. See for example figure 1. Note that this does not use `\includegraphics`, so does not have access to all the formatting, scaling, cropping and other options of that command.

7. What if it doesn't work?

   If your dvi viewer or whatever you are using to convert the `.dvi` file to another format (PDF, PostScript, etc) complains that it can't find the font, it may mean that is is not clever enough to generate the pk file that it needs from the METAFONT output. What I have done above generates 600 dpi output, but your system may want something else. Best thing is to put the required dpi (it should be apparent in the error messages) into the top of the `.mf` file — as noted above, there are some fields to uncomment if you like, and the can also be edited. Then run gftopk on the resulting `.XXXgf` file and you should be away. The mode flag on the METAFONT command line may need to be altered or removed, I'm not sure.

   You may find that dvipdfmx does a better job of making a PDF from the `.dvi` file than going via PostScript. I find that the PostScript from dvips looks good but ps2pdf sometimes does not give a good PDF file where dvipdfmx does.

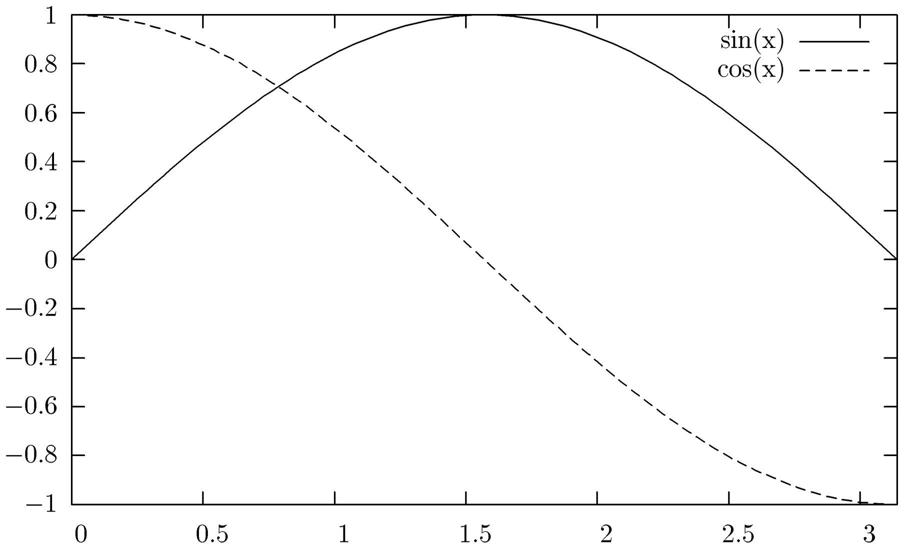

Figure 1: Here is my pointless plot.

8. Some more details.

   LaTeX (and TeX) need only the `.tfm` file to typeset the document — it contains the information on the sizes and so forth of the characters. But rendering the 'glyph' needs the `pk` file.

   I find it simplest to leave these font files in the working directory. It makes little sense to install a graphic as a font system-wide.

   Keep in mind that gnuplot is scriptable. It can be run from the command line. All the other steps can be similarly automated. That means that this is a nice path for incorporating graphics that may themselves need to be updated. Say you are using gnuplot to plot a text file full of data. If that text file changes, you can just rerun a script that calls gnuplot and runs METAFONT and any ancillary programs you need. This allows the graphics to be dynamically updated without having to manually update them. A similar thing can be done with other forms of output from gnuplot, of course.