Some Experiences in Running METAFONT and MetaPost

Peter Wilson*
CUA
now at: `peter.r.wilson@boeing.com`

10 June 2000

Abstract

Although the manuals describing the METAFONT and MetaPost languages are good, the instructions for actually running the programs are somewhat weaker. This article describes some of the things that I have learnt in this respect, and may be helpful to others who are starting along the Meta trail.

Contents

List of Tables

*With additional insights and contributions by Daniel Luecking (`luecking@comp.uark.edu`).

1 Introduction

When I started using METAFONT, and later MetaPost, there was sufficient documentation available so that I could produce workable code. The problem was that I could not find much documentation on how to run them to see if my code actually worked. This is just a short summary of the results of keeping a watchful eye on questions and answers[1] on `comp.text.tex`, and other experiments, in order to get the systems to run. It is offered in the hope that it might be useful to others and comes with a request for corrections or additions to my remarks.

1.1 Conventions

I work on a couple of different un*x boxes and use the teTeX distribution, so some of my remarks may have to be translated if you are on another kind of box or have a different distribution. I'll use `$TETEX` as the root of the TeX directory system. In my case this is something like `/opt/teTeX`, so the directory `$TETEX/texmf` is actually `/opt/teTeX/texmf`. As another shorthand I'll use `$TEXMF` to stand for `$TETEX/texmf`.

I will also use `?` as the system prompt when I want to show an example terminal session.

2 General

The METAFONTbook [Knu86] is indispensable if you want to use METAFONT for creating your own fonts.

MetaPost is based on METAFONT and was developed by John Hobby [Hob92]. Although the user manual explains pretty much everything about the language I found the METAFONTbook was a useful supplement. Both [GRM97] and [Hoe98] give examples of MetaPost code; the cover illustration on Hoenig's book is a MetaPost picture. MetaPost produces Encapsulated PostScript and Keith Reckdahl describes how this can be best imported into LaTeX documents [Rec97].

When documenting METAFONT or MetaPost code I use the conventions of the LaTeX DOCSTRIP utility and the doc package [GMS94]. Unsurprisingly I supplement the doc package with the docmfp package [Wil99] which I developed for METAFONT and MetaPost code.

3 METAFONT

My method of learning to use METAFONT for fontmaking was to study others' code and then try and adapt bits and pieces to my purposes. Some of the results are on CTAN. The fonts in `../fonts/archaic` are very simple rectilinear glyphs as they are based on scratched lettering on walls or pottery (from about 1000 to 400BC). Some manuscript fonts, written with a broad-nibbed pen and much more curved, are in `../fonts/bookhands`. The Trajan

[1] Among others, Brian Elmegaard (`be@et.dtu.dk`) has been very responsive to queries about MetaPost.

font is more complex again as it is a constructed uppercase seriffed font based on a Roman inscription dated 114AD and is in `../fonts/trajan`.

The command to run METAFONT is `mf` on my system (but it might differ on yours), and I'll use `FNT` to stand for the name of a new font, so correspondingly the name of the main METAFONT file is `FNT.mf`.

I found it useful to create a directory called `$TEXMF/fonts/source/public/FNT` in which to do the METAFONTing.

An important part of the font design process is viewing large proofs of the characters. A special gray font is used for this purpose. I use the common *ljfour* (600dpi) printer mode, but the first time I tried to produce a proof I got an error regarding the mode, and it took a while to get over this hurdle. The gray fonts are in `$TEXMF/fonts/source/public/misc` but as provided they are only for an *imagen* (300dpi) printer. To fix the problem I did the following.

- Copied `gray.mf` to `gray.original.mf`
- Edited `gray.mf` by commenting out the `imagen` line and added an `ljfour` line, so that the file then looked like:

```
% gray.mf
% input grayimagen  % ‘standard’ gray font is for imagen
input grayljfour    % gray font for ljfour mode
% end gray.mf
```

- Copied `grayimagen.mf` to `grayljfour.mf` and edited this to become:

```
% grayljfour.mf
if mode<>ljfour: errmessage ‘‘This file is for ljfour only’’; fi
font_identifier ‘‘GRAYLJFOUR’’;
boolean lightweight;
input grayf
% end grayljfour.mf
```

- I kept all the files in the same `../public/misc` directory.

To generate a proof font I do

```
? mf FNT
? gftodvi FNT.2602gf
```

and the proof output is written to `FNT.dvi`. It can then be viewed in the normal manner; normally I find that using xdvi is usually adequate instead of hardcopy. Actually, I put the commands into a script as any new font needs lots of checking and revision. For example

```
# script pFNT to generate proof font for FNT
mf FNT
gftodvi FNT.2602gf
```

```
echo
echo Proof output in FNT.dvi
# end pFNT script
```

After I got some proof characters that I was happy with I then used them in a normal LaTeX document to see how they appeared at normal size and how they fitted together. This involved writing a small test file, a package to enable the new font to be used easily, and font description (`.fd`) files. I also found that I had to add line(s) like:

```
FNT.mf      public      FNT
```

to the file `$TEXMF/fontname/special.map`, otherwise the automatic font creation utilities could not find the `FNT.mf` file. The Appendix includes examples of the various files for one of my manuscript fonts.

The automatic font creation procedure creates files in directories `$TEXMF/fonts/tfm/public/FNT` for the `.tfm` files, and in `$TEXMF/fonts/pk/ljfour/public/FNT` for the `.pk` files (if you use a mode other than ljfour then the `.pk` files will be in a correspondingly different directory). If `.tfm` and `.pk` files exists in these directories then METAFONT will not be called. So, when any changes are made to the `FNT.mf` file the contents of these two directories must be deleted before the test document is LaTeXed. I wrote another script that emptied the directories and then ran the test document (called `tryfont.tex` for this example). This looks like:

```
# script TRYIT to run testfont document
echo removing FNT.tfm and FNT.pk files
rm $TEXMF/fonts/tfm/public/FNT/*.tfm
rm $TEXMF/fonts/pk/ljfour/public/FNT/*.pk
echo typesetting
latex tryfont
echo Result in FNT.dvi
# end TRYIT script
```

LaTeX comes with a file called `nfssfont.tex` which is designed for interactively testing a new font, either instead of or as well as using a test file of your own making. A typical session might look like this, where the font under test is called `rust10`:

```
? latex nfssfont
Normal latex startup messages

Name of the new font to test = rust10
Now type a test command (\help for help):)
*\table
[1]
*\bye
(see the transcript file for additional information)
Output written on nfssfont.dvi (1 page, 6040 bytes).
Transcript written on nfssfont.log
? dvips nfssfont
```

After latex prints the usual startup messages you are asked for the name of a font to test. In the example I responded with `rust10` as the name of the font (note that there is no file extension). You are then invited to type a command (the nfssfont prompt is `*`). I responded with `\table` which generates a table containing all the characters in the font. At the next prompt I said `\bye` which ends the session. The resulting `nfssfont.dvi` file can then be printed for inspection. Typing `\help` at any `*` prompt will list a brief description of the available commands on the terminal.

Finally, iterate and iterate and iterate until you are satisfied with your new font in all its possible uses.

4 MetaPost

The first problem that I had when trying to run MetaPost was that the command given in the user manual, namely `mp`, wrote a bunch of PostScript code to the terminal but otherwise was no help. I eventually figured out that the correct command for my system is `mpost`. Having a file `pic.mp` containing MetaPost figures and running:

```
? mpost pic
```

will produce a set of output files called `pic.N`, where each `N` corresponds to the `beginfig(N)` command in `pic.mp`. These are Encapsulated PostScript files and can be used within a LaTeX document via the normal `\includegraphics` command. For example:

```
\begin{figure}
\centering
\includegraphics{pic.3}
\caption{...}
\end{figure}
```

For proof viewing of MetaPost results there is a file called `mproof.tex` which should be in your LaTeX distribution. Doing:

```
? tex mproof pic.1 pic.2 ...
```

where `pic.1`, `pic.2`, etc., are MetaPost output files, will generate `mproof.dvi` containing the MetaPost generated Encapsulated PostScript diagrams. This can then be viewed in the normal[2] manner.

4.1 (pdf)LaTeX

pdfLaTeX can not normally handle Encapsulated PostScript files but can handle MetaPost output files because of their particularly simple PostScript usage. However, pdfLaTeX expects these files to have a `.mps` suffix. Thus, `pic.3` would have to be renamed to someting like `pic3.mps` to be used by pdfLaTeX. In the case of a single MetaPost diagram this is simple enough but could get tedious if there were many of them. There are two solutions to this problem:

[2] Not all screen viewers support Encapsulated PostScript, so you may have to print the file.

1. Noting that LaTeX will accept MetaPost output files with a `.mps` extension, write a script that will copy all files of the form `pic.N` to `picN.mps`. Then you can use `\includegraphics{pic3.mps}` without being concerned whether it will be processed by LaTeX or by pdfLaTeX.

   Here is a pair of scripts that will do this. One is a un*x shell script and the other is a Perl script.

```
#! /bin/sh
######################################################################
# Shell script n2mps
# Call as: n2mps basename
# List each file basename.* in the directory and run the Perl script
# to copy each basename.N to basenameN.mps, where N is an integer
#
# Copyright 2000, Mauro S. Costa and Peter R. Wilson
######################################################################
basename=${1:?"A file basename is required."}
extname=N.mps
echo Files $basename.N, where N is a number, will be copied to $basename$extname
for file in `ls $basename.*`
do
 n2mpsprl.prl $file
done
######################## end shell script ###########################


#!/usr/local/bin/perl -w
####################################################################
# Perl script: n2mpsprl.prl
# Call as: n2mpsprl.prl filename
# If filename is of the form basename.N, where N is an integer,
# copies file basename.N to file basenameN.mps
#
# Copyright 2000, Mauro S. Costa and Peter R. Wilson
###################################################################

# Test for correct number of input parameters
die "Invalid command line arguments.\nTry $0 <src> \n" if($#ARGV > 1);
die "Invalid command line arguments.\nTry $0 <src> \n" if($#ARGV < 0);

# Assign input file name to variable
$input_file = $ARGV[0];

# test if ends with a number, exit if not
if ($input_file =~ /\w\.\d/) { ; } else { exit; }

# Remove the "dot" from the string variable
```

```
# holding the input file name
$input_file =~ s/\.// ;

# Create a list variable composed of the string variable holding
# the concatenated input file name and the extension ".mps"
@name_list = ($input_file,'.mps') ;

# Join the string variables in the name_list variable into
# a single string variable
$output_file = join("",@name_list) ;

# create a list variable composed to the parameters needed
# for the system copy command excution
@exec_list = ('cp', $ARGV[0], $output_file) ;

# Execute the system copy  ("cp") command
system(@exec_list) ;
############################ end perl script ##########################
```

To copy the `pic.N` files all you have to do is:

```
? n2mps pic
```

2. The second method, as pointed out by Daniel Luecking, is to declare the extension via `\DeclareGraphicsRule` at some point in the document before any graphics are included (or in a seperate graphics configuration file). His example is:

   > For example,
   >
   > `\DeclareGraphicsRule{.foo}{mps}{.bar}{}`
   >
   > says to treat file `graphic.foo` just like `graphic.mps`, and to look for the Bounding Box in `graphic.bar`. The special form
   >
   > `\DeclareGraphicsRule{*}{mps}{*}{}`
   >
   > says to treat any extension not otherwise declared as if it were `.mps`, and search the graphic file itself for the Bounding Box. The full story is in `grfguide.ps` [Car99][3].

4.2 TeX or LaTeX

By default MetaPost uses the TeX `cmr10` font for typesetting labels (text). There are basically two ways of specifying text. One is by enclosing the characters within double quotes (e.g., `"the text"`), which I will term *quoted*. The other is by enclosing the characters within a `btex ...etex` pairing (e.g., `btex the text etex`), which I will term *btexed*.

[3]This should be available in any modern LaTeX distribution.

In the quoted case the characters appear to be output just as written. This may cause problems as the `cm` fonts do not include spaces and the glyph mapping is somewhat idiosyncratic. This means that you may not see what you expected in the final picture. However, if the quoted text consists only of the alphanumeric characters (i.e., no spaces, punctuation, etc.) then I haven't found any problems.

In the btexed case MetaPost, by some process I do not understand[4], uses TeX to generate the typeset text. TeX commands can be used in the normal manner as part of the text. If you prefer LaTeX to TeX, then the default has to be changed. This requires two things to be done.

1. MetaPost uses an environment variable called `TEX`. If this is either unset or equals `tex` then TeX is used for typesetting. To use LaTeX this must be set to `latex`. On my system this can be done in either a login file or temporarily for a session. In my case the commands for a session are:

   ```
   ? TEX=latex
   ? export TEX
   ```

2. At the start of a MetaPost file, TeX or LaTeX commands can be put between a `verbatimtex ...etex` pairing. The appropriate LaTeX commands must be put there if LaTeX is to be used. For example:

   ```
   verbatimtex
     \documentclass{article}
     % \usepackage{...}
     % \newcommand{...}{...}
     \begin{document}
   etex
   ```

If you are using LaTeX there is the `emp` package [Ohl97] which lets you embed MetaPost code within the LaTeX document itself rather than having it as seperate file(s). If you use this approach, then the `TEX` environment variable must, of course, be set to `latex`. The user manual should be consulted for how to deal with the `verbatimtex ...etex` group.

When the normal TeX fonts are used, they are not included within the generated Encapsulated PostScript files. This is allright if they are to be used in a TeX document, but often not if they are to be used in other ways, perhaps just being printed by themselves. I generate a stand-alone PostScript file by LaTeXing a document that consists *only* of the MetaPost graphic — no page numbers, no captions, no nothing else. I then use `dvips` to convert the dvi file to PostScript, (which will now include all the font information). I can then use this as I want, but read Reckdahl's advice [Rec97] as sometimes further manipulation is required.

[4]But see later on page 16.

4.3 PostScript fonts

MetaPost can be made to use PostScript fonts and, providing the texts don't require any special (TeX) processing, then use of TeX or LaTeX may be avoided altogether. In order to do this, certain requirements have to be put on the contents of the MetaPost file.

1. The file must begin with the line
   `prologues:=2;`
   which will add a font-related prologue to the generated PS files.

2. The default font must be changed. What has to be changed depends on the desired font. For example, to use Times Roman, put
   `defaultfont:="ptm8r";`
   just after the `prologues` line.

   For this incantation to work, the file `ptmr8mr.tfm` must exist in a location where MetaPost looks for `.tfm` files. On my system this file is in directory `$TEXMF/fonts/tfm/adobe/times`. Also, the file `psfonts.map` must include a line like
   `ptm8r      Times-Roman ...`
   Again, on my system `psfonts.map` is in directory `$TEXMF/dvips/misc`.

I don't know whether this all works if you don't have a LaTeX installation, but if you are are reading this it's unlikely that you don't have one.

Daniel Luecking has the following comments on the above.

> You don't necessarily need to change the default font. What you need is a line in `psfonts.map`, say if the BSR[5] fonts are used:
>
> ```
> cmr10    CMR10...
> ```
>
> *and* support in your PostScript viewer for CMR10. For Ghostscript, a line like the following in `Fontmap`:
>
> ```
> /CMR10    (cmr10.pfb);
> ```
>
> *and* the directory containing `cmr10.pfb` in the Ghostscript search path. You still need `prologues:=2;`.
>
> You don't need a LaTeX installation to use MetaPost in this way, but you still need a `.tfm` file, and `psfonts.map`, and MetaPost must be looking for them in the right places. It is extremely unlikely that you will have all this without it being part of a system that includes LaTeX.

[5] Blue Sky Research Type 1 PostScript versions of the cm fonts.

4.4 Font styles and sizes

I have spent some time trying to work out how to change the size of the fonts for MetaPost text. The answer depends on what you are doing and I ran a series of experiments to try and make some sense out of what was and was not happening when I changed things. Some things are still a mystery to me, though.

I wrote a short MetaPost file that included all the combinations of ways I have used to specify labels. These are:

- Text specified as either quoted or btexed;
- Fixed text within a macro;
- A macro with a text argument;
- The normal MetaPost `label` in the 'body' of a diagram.

The following is a snapshot of the MetaPost file used for the experiments; in its state below it tests for using LaTeX with the times package and the default font set to `ptmr8r` (also for Times).

```
% try.mp  fiddle with MetaPost fonts

prologues:=2;

verbatimtex
  \documentclass{article}
%  \documentclass[12pt]{article}
  \usepackage{times}
%  \renewcommand{\familydefault}{cmss}
  \begin{document}
%  \tiny
etex

defaultfont := "ptmr8r";  % uses PostScript font
%defaultfont := "cmss10";  % uses different TeX font
%defaultscale := 1.2;      % scales font

%%%%%%  next set of definitions have been put into file trymac.mp
%numeric u; u:=1mm;
%
%def labelQMAC(suffix $) =            % fixed quoted macro
%  label.rt("QMAC", z$);
%enddef;
%
%def labelBMAC(suffix $) =            % fixed btexed macro
%  label.rt(btex BMAC etex, z$);
%enddef;
```

```
%
%def labelTEXT(suffix $)(text str)=   % macro with text argument
%  label.rt(str, z$);
%enddef;
%%%%%% end of contents of trymac.mp

input trymac

beginfig(1)

%%%% Fixed text in a macro
z11=(0,0);
labelQMAC(11);

z12=(0,10u);
labelBMAC(12);

%%%% Macro with a text argument
z13=(0,20u);
labelTEXT(13)("QTEXT");

z14=(0,30u);
labelTEXT(14)(btex BTEXT etex);

%%%% normal labeling
z21=(50u,0);
label.rt("QOUT", z21);

z22=(50u,10u);
label.rt(btex BOUT etex, z22);

endfig;               % end fig 1

end
```

Different conditions were tested by commenting and uncommenting appropriate lines at the start of the file. As another axis to the experiments, they were performed once with the macro definitions `input` from the seperate `trymac.mp` and repeated with the macro definitions explicitly written (integrated) in `try.mp`.

In the table of results, Table 1, a check mark in a column indicates that the typeset label was modified as a result of changing the font and/or size values. For labels typeset with the PostScript Times font being specified, 'T' indicates the Times font and 'C' indicates a Courier-like font.

The following summarises the experimental results which are common to both TeX and LaTeX typesetting.

- Only quoted text (e.g., `"quoted text"`) is affected by changes to the `defaultfont`

Table 1: MetaPost label typesetting

Setting	QMAC	BMAC	QTEXT	BTEXT	QOUT	BOUT
TeX and LaTeX, integrated and input macro definitions						
`scale: 1.2`	X		X		X	
`font: "cmss10"`	X		X		X	
`font: "ptmr8r"`	T	C	T	C	T	C
LaTeX, integrated macro definitions						
`[12pt]`		X		X		X
`tiny`		X		X		X
`family: cmss`		X		X		X
`package: times`	C	T	C	T	C	T
`times & "ptmr8r"`	T	T	T	T	T	T
LaTeX, input macro definitions						
`[12pt]`				X		X
`tiny`				X		X
`family: cmss`				X		X
`package: times`	C	C	C	T	C	T
`times & "ptmr8r"`	T	C	T	T	T	T

and `defaultscale` values.

- When using the PostScript Times font the quoted text is typeset in Times. Oddly, the btexed text (e.g., `btex text etex`) appears to be typeset in a kind of Courier font.
- The effects of changing the `defaultfont` and `defaultscale` are independent of whether macros are `input` from another file or are integral to the diagram file.

Some of the results when LaTeX font changing commands were used are more problematical. First, though, I came across a peculiarity when `input`ing the `trymac.mp` file. When the first attempt at doing this in a working session was using LaTeX typesetting, MetaPost always reported errors at the `labelBMAC` macro definition and was unable to complete the processing of `trymac.mp`. An extract from a typical `mpxerr.log` file looked like:

```
This is TeX, version 3.14159 (C version 6.1) (format=latex 98.11.5)
**\batchmode\input mpx4662
(mpx4662.tex
Missing character: There is no B in font nullfont!
...
[1]
! I can't find file `mpx4662.aux'.
...
*** (job aborted, file error in nonstop mode)
```

Also, each time the `'mpx4662'` was reported with a different number (e.g., `'mpx4926'`).

However, if I first used TeX typesetting while `input`ing `trymac.mp` no errors were reported. I could then switch to using LaTeX and again no errors were reported. Note that there was never a problem if the macros in `trymac.mp` were integrated into `try.mp`.

When the macros were integrated in `try.mp` the results were:

- The LaTeX size and non-PS font changing commands only affected the btexed text. Typical of these commands are:
  `\documentclass[12pt]{...}`,
  `\renewcommand{\familydefault}{cmss}`, and
  `\tiny` after the `\begin{document}`.
- Using the `times` package (to use the PostScript Times font) resulted in the btexed text being set in Times and the quoted text in Courier.
- Using both the `times` package and `ptmr8r` as the `defaultfont` resulted in all labels typeset in Times.

Slightly different results occurred when the macros were `input`.

- The LaTeX size and non-PS font changing commands did not affect the text (either quoted or btexed) that was embedded in the `labelQMAC` and `labelBMAC` macros.
- Using the `times` package resulted in the btexed text in macro `labelBMAC` being set in Courier instead of Times.
- Using both the `times` package and `ptmr8r` as the `defaultfont` resulted in all labels typeset in Times except for that in the `labelBMAC` macro which was set in Courier.

I can only surmise that the texts in the `labelQMAC` and `labelBMAC` macros were fixed at the time when TeX processed `trymac.mp`.

My overall conclusions from the experiments are:

- Changing the `defaultfont` and/or `defaultscale` only affects quoted text.
- LaTeX and TeX commands only affect btexed text.
- Be vigilant when using both `input` macros and LaTeX.

Sven de Vries[6] discovered the following when using the MetaPost `graph` package and using `cmr7` as the default font. He said

> I realized that in:

```
defaultfont:="cmr7";

for u=20 step 80 until 500:
  otick.bot(format("%ge3",(u/10)*(u/10)/10),u);
endfor;
```

[6] Email dated 3 December 1999, `devries@mathematik.tu-muenchen.de`.

only the letters e3 are changed to cmr7.

While searching through the format file I found that

```
Fmfont_:="cmr7";
```

is a way to fix it.

A Example files for a METAFONT font

Here is an example of various files for one of the manuscript fonts, namely the Roman Rustic font. The documented originals of these are on CTAN in `../fonts/bookhands/rustic/rustic.dtx`. Full descriptions of how to install new fonts for use by LaTeX can be found in any of [Lat98, GMS94, Hoe98].

The first file is a package file that caters for both OT1 and T1 font encoding schemes.

```
%% This is file ‘rustic.sty’,
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{rustic}[1999/05/01 v1.0 package for Roman Rustic fonts]
\providecommand{\Tienc}{T1}
\ifx\Tienc\encodingdefault
  \newcommand{\rustfamily}{\usefont{T1}{rust}{m}{n}}
\else
  \newcommand{\rustfamily}{\usefont{OT1}{rust}{m}{n}}
\fi
\DeclareTextFontCommand{\textrust}{\rustfamily}
\endinput
%% End of file ‘rustic.sty’.
```

Next is an example `.fd` file for the T1 encoding. The `ot1rust.fd` file for the OT1 encoding is exactly the same except that everywhere `T1` is replaced by `OT1`. Note that there is both a normal and a bold version of the font, and it comes in three different sizes; that is, there are METAFONT files `rust7.mf`, `rust10.mf` and `rust17.mf` for the normal font, and files `rustb7.mf`, `rustb10.mf` and `rustb17.mf` for the bold version. Slanted styles (and distinctive capital letters) were unkown at the time this bookhand was in use.

```
%% This is file ‘t1rust.fd’,
\DeclareFontFamily{T1}{rust}{}
  \DeclareFontShape{T1}{rust}{m}{n}{ <-8.5> rust7 <8.5-15> rust10 <15-> rust17 }{}
  \DeclareFontShape{T1}{rust}{m}{sl}{ <-> sub rust/m/n }{}
  \DeclareFontShape{T1}{rust}{m}{it}{ <-> sub rust/m/n }{}
  \DeclareFontShape{T1}{rust}{m}{sc}{ <-> sub rust/m/n }{}
  \DeclareFontShape{T1}{rust}{m}{u}{ <-> sub rust/m/n }{}
  \DeclareFontShape{T1}{rust}{bx}{n}{ <-8.5> rustb7 <8.5-15> rustb10 <15-> rustb17 }{}
  \DeclareFontShape{T1}{rust}{bx}{it}{ <-> sub rust/bx/n }{}
  \DeclareFontShape{T1}{rust}{bx}{sl}{ <-> sub rust/bx/n }{}
  \DeclareFontShape{T1}{rust}{b}{n}{ <-> sub rust/bx/n }{}
\endinput
%% End of file ‘t1rust.fd’.
```

Finally, here is a LaTeX test file. I tend to use the same basic file for all my fonts, just changing the package, font commands and font name appropriately.

```
%%% tryfont.tex    Test Roman Rustic fonts
\documentclass{article}
\usepackage{rustic}

\newcommand{\ABC}{ABCDEFGHIJKL MNOPQRSTUVWXYZ}
\newcommand{\abc}{abcdefghijkl mnopqrstuvwxyz}
\newcommand{\punct}{.,;:!?`' \&{} () []}
\newcommand{\figs}{0123456789}
\newcommand{\dashes}{- -- ---}
\newcommand{\sentence}{%
this is an example of the roman rustic font. now is the time for all good
men, and women, to come to the aid of the party while the quick brown fox
jumps over the lazy dog:}

\newcommand{\Sentence}{%
This is an example of the Roman Rustic font. Now is the time for all good
men, and women, to come to the aid of the party while the quick brown fox
jumps over the lazy dog:}

\title{Try Roman Rustic Fonts}
\author{}
\date{}
\begin{document}
\maketitle

    This provides a short test of the characters in the Roman Rustic fonts
--- the \verb|rust| font family.

\begin{center}
The Rustic Huge normal font. \\ \par
{\rustfamily\Huge \ABC\\ \abc\\ \punct{}\dashes\\ \figs\\ \par }
\end{center}

\begin{center}
The Rustic font in its normal size \\
\textrust{\ABC{} \abc{} \figs} \\
\end{center}

\begin{center}
The bold minuscule font, the normal minuscule font, and the bold Computer Modern
Roman, all in the normal size \\
\textrust{\textbf{\abc{} \figs}} \\
\textrust{\abc{} \figs} \\
\textbf{\abc{} \figs} \\
\end{center}
```

```
\begin{center}
The bold versions, in Huge and tiny sizes. \par
\rustfamily\bfseries
\Huge \abc{} \figs \par
\tiny \abc{} \figs \par
\end{center}

\begin{center}
The font in the tiny size \\ \par
{\rustfamily\tiny \ABC{} \\ \abc\\ \figs\\ \par }
\end{center}

\begin{center}
    Some ligatures in the normal font \\
\textrust{``the lazy dog --- but quick fox?''}
\end{center}

{
\rustfamily
\sentence{}

\Sentence{}
}

    This is the end of the test file.
\end{document}
```

B MetaPost btexed processing

In response to an earlier version of this document, Daniel Luecking provided the following explanation of btexed processing.

When MetaPost inputs a file and it encounters a `btex ...etex` group not in quotes, it calls MPtoTeX to scan the whole file and extract `btex ...etex` groups, and writes a temporary `.tex` file (with a name like `mpx4527.tex`, though the exact rule for generating a unique name is either system dependent or a compile time choice). Any `verbatimtex` block is copied verbatim, but the `btex` blocks are wrapped in some special TeX code.

For example, suppose the `.mp` file contains

```
  btex \hskip 2.2bp Q etex
```

then the `.tex` file gets something like the following

```
\shipout\hbox{\smash{\hbox{\hbox{% line 10 test.mp
\hskip 2.2bp Q}\vrule width1sp}}}
\end{document}
```

You can run MPtoTeX seperately and see exactly what is written in any case. Note the `\end{document}`; in plain TeX everything after `\end` is ignored.

After *all* the `btex ...etex` groups are processed in this way, TeX is called and a `.dvi` file is produced. Then the program DVItoMP is called to process the `.dvi` file and produce an `.mpx` file. In the `.dvi` file the following information resides: (1) the relative movement needed to get from the reference point of the `\hbox` to the reference point of the lettter 'Q' (given the above example), (2) the font it is from, (3) the location of the bottom and top of the rule, and (4) a magnification. This enables DVItoMP to determine the location of 'Q' and the bounding box of the whole `\hbox`, and then write to the `.mpx` file something equivalent to

```
picture _p;_p:=nullpicture;
addto -P also "Q" infont "cmr10" scaled 1 shifted (2.2,0);
setbounds _p to ...;
```

After this, the `btex ...etex` group is internally replaced by a reference to the picture `_p`.

Note that MPtoTeX is called whenever a new file is input, and it processes the entire file in one pass. If you have another file input to the main `.mp` file, this is a seperate process. A consequence of this is that *the input file needs its own `verbatimtex` block.*

This explains the problem with your (PW) experiment running MetaPost on `try.mp`: MPtoTeX writes a `.tex` file containing the `verbatimtex` plus the two `btex` groups. LaTeX processes this file OK. Now MetaPost inputs `trymac.mp` and encounters no `verbatimtex` but three `btex`'s. It writes a second `.tex` file with no LaTeX header and runs LaTeX (since that was what the `TEX` environment variable was set to) and LaTeX encounters `\hbox{BTEXT...}`. Since no LaTeX setup has occurred there is no default font and you get the complaint `'There is no B in nullfont!'`. Since none of the LaTeX preamble occurs, no `.aux` file was written, and `\end{document}` (which MPtoTeX puts at the end of all its `.tex` files — just in case!) complains that there is no `.aux` file.

When you set `TEX=tex` and run MetaPost first, the `trymac.mpx` file is created. When you then set `TEX=latex`, `trymac.mpx` already exists. As long as it is newer than `trymac.mp`, then MPtoTeX, etc., will not be called to make it and that run seems to succeed, except none of the LaTeX commands ever got into that `.tex` file.

I'm not sure why you got Courier font in some examples, unless there is a substitution rule somewhere in the chain from MetaPost to viewing for fonts that can't be found. I usually get an error from Ghostscript.

References

[Car99] D. P. Carlisle. *Packages in the 'graphics' bundle.* January 1999. (Available from CTAN as `../macros/latex/required/graphics/grfguide.ps`).

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, 1994.

[GRM97] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, 1997.

[Hob92] John D. Hobby. *A user's manual for MetaPost*. Computing Science Technical Report no. 162, AT&T Bell Laboratories, Murray Hill, NJ, April 1992. (Available from CTAN with the MetaPost distribution at `.../graphics/metapost`).

[Hoe98] Alan Hoenig. *TeX Unbound*. Oxford University Press, 1998.

[Knu86] Donald E Knuth. *The METAFONTbook*. Addison-Wesley Publishing Company, 1986.

[Lat98] LaTeX3 Project Team. *LaTeX2e font selection*. April 1998. (Available from CTAN as `...macros/latex/doc/fntguide.ps`).

[Ohl97] Thorsten Ohl. *EMP: Encapsulated MetaPost for LaTeX*. November, 1997. (Available from CTAN at `../latex/macros/contrib/supported/emp`).

[Rec97] Keith Reckdahl. *Using EPS Graphics in LaTeX2e Documents*. February 1997. (Available from CTAN as `../info/epslatex.ps`).

[Wil99] Peter Wilson. *The docmfp package*. 1999. (Available from CTAN at `../latex/macros/contrib/supported/docmfp`).

Note See `http://www.tug.org` for information on accessing CTAN — the Comprehensive TeX Archive Network.