SeeTeX Installation Guide

Dirk Grunwald
Dept. of Computer Science
University of Colorado
grunwald@foobar.colorado.edu

Chris Torek
Department of Computer Science
University of Maryland
College Park, MD 20742
chris@mimsy.umd.edu
*

1990

Contents

*To be fair, Chris didn't write this slop, except for the sections on the `fontdesc` file; I include his name here because of that section and because he wrote the 'mctex' library on which SeeTeX stuff is based, not to mention some of the tools herein

1 Introduction

SEETEX is a package of output drivers for TEX, and includes the following programs:

Program	Purpose
XTEX	Fast X11 windows based TEX previewer (formerly called TEXX2).
MFTOBDF	Program to convert METAFONT generated fonts to the `BDF` format; this is used by XTEX.
TEXX	Antiquated raw X11 windows based TEX previewer.
TEXSUN	SunView based TEX previewer.
TEX3B1	A TEX previewer for the AT&T 3B1, contributed by Andy Fyfe at CalTech.
`dviselect`	Utility to extract individual pages from a `DVI` file. This is just Torek's **`dviselect`**.
IPTEX	Output driver for Imagen printers; this is a very different and enhanced version of Torek's IPTEX.

All of these programs depend on a library of routines, called LIBTEX, that was originally written by Chris Torek at the Univ. of Maryland. SEETEX is distributed in two ways. The first, usually called something like **`SeeTeX-2.15.3.tar.Z`** is the full SEETEX distribution. The second, usually called **`xtex-2.15.3.tar.Z`** is the partial XTEX distribution that updates the version of XTEX in the full SEETEX distribution. Because the full SEETEX distribution changes infrequently, snagging only the XTEX distribution allows you to update XTEX without having to grab the full SEETEX tar. In this installation guide, I will assume you have unpacked full SEETEX distribution and have the directory structure shown in Figure 1 in the following document. Only directories and important files are included in Figure 1 .[1]

1.1 How Not To Read This Manual

SEETEX is configured to use the IMAKE installation configuration system distributed with X11. If you do not use IMAKE, you can use the script **`NoImake`** to convert the Imakefiles into Makefiles. You must edit file **`Imake.cpp`** to set various variables, and then run **`NoImake`**. You should not directly editing the Makefiles; only edit **`Imake.cpp`**. If you run X11 and do not use IMAKE, contact your vendor or system supportoid to get IMAKE installed, or use **`NoImake`**.

If you already understand TEX fonts and the **`fontdesc`** file, you can skip §2 and §3 , resp. If you are installing everything but XTEX, you only need to read §4 . If you're also installing XTEX and MFTOBDF, you need to read §6 .

[1] By the way, if you're viewing this with XTEX, you'll notice that page, figure and section references have little boxes around them. Point your mouse in the box, hold META and button button-2 (middle). You will temporarily zip to the referenced item. Releasing the mouse returns you to your original page. This is done automatically by the `xtex.sty` file packaged with xtex.

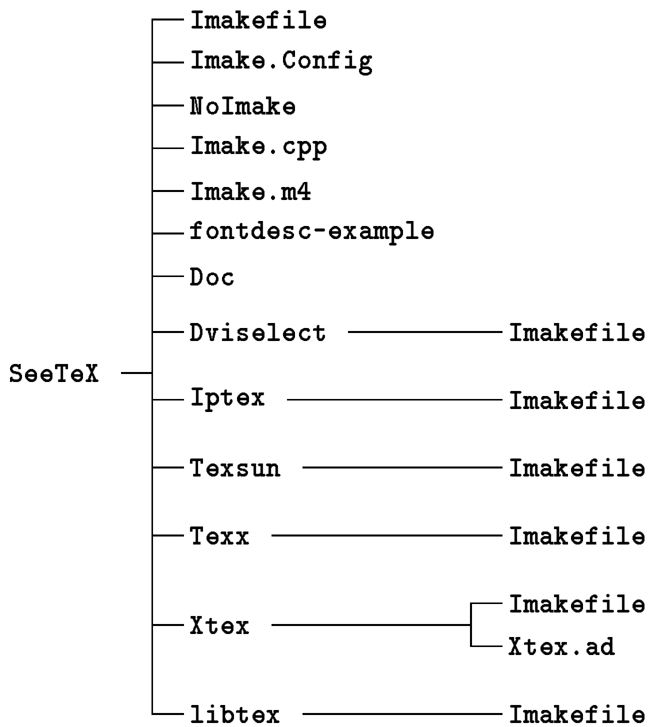

Figure 1: SeeTeX Directory Structure

2 A Little Explanation of TeX Fonts

When printing TeX documents, you use *bitmap* fonts that can be stored in the `PK`, `GF` and `PXL` formats. The bitmaps are *designed* for a specific size, e.g., `cmr10` is a 10 point Computer Modern Roman font. This font can be *rendered* at different *resolutions* and *magnifications*. For example, on a laser printer, you might render the `cmr10` font at 300dpi (dots per inch). For a computer display, you might render `cmr10` at e.g., 85dpi rather than 300dpi. Magnifications are expressed as the percentage magnification multiplied by 1000 and then truncated, so the normal magnification is 1000 (i.e. magnify by 1.0).

For the `GF` and `PK` formats, a *file suffix* is attached to each font file encoding the resolution and magnification. A similar, although completely different suffix is appended to the antiquated `PXL` format. If you still use `PXL`, flog your local system supportoid and convert those fonts to `GF` or `PK` – you'll save oodles of disk space. But lets say you render `cmr10` for a 300dpi print at magnifications of 500, 1000, 1098, 1200, 1440, 1728, 2074 and 2489. These numbers are "magic" for TeX and LaTeX, because they magnify the 10pt font by 1220 at each step. This renders our nominal 10pt font to 5pts, 10pts, 11pts, 12pts, 14pts, 17pt, 20pt and 24pt. See the TeXbook for more details. In general, a 10pt font rendered to look like a 5pt font doesn't look as good as a font designed to be a 5pt font.

For each of these magnifications, we'll get a file with a suffix that's approximately

$$\text{suffix} = \text{resolution} \times \frac{\text{magnification}}{1000}$$

and ends in either `PK` or `GF`. So, for our buddy `cmr10` rendered at 300dpi and stored in `PK` format, we'd have the following files: `cmr10.150`, `cmr10.300`, `cmr10.327`, `cmr10.432`, `cmr10.518`, `cmr10.622`, `cmr10.747`. Whether you have all these files or not is up to you, your TeX administrator and your disk. Only programs that print TeX documents need these bitmap fonts; TeX itself uses something called `TFM` files that record the width and height of each glyph in a font. This is enough information for TeX to layout your document. There is only *one* `TFM` file for each font, e.g. for `cmr10`. You do not need different `TFM` files for each magnification or resolution of that font, because the `TFM` distances are encoded in units of 10^{-7} meters. Magnifications of the font simply scale those dimensions. Different resolutions affect only the bitmap fonts, and not the absolute dimensions recorded in the `TFM` file; the 'x' in a 10pt font is always 10pt wide, or $10 \times \frac{254}{7227}$ cm wide, no matter what the actual *resolution* of the printing device. You are now a TeX font wizard.

3 Configuring and Installing `fontdesc`

When you edit `SeeTeX/Imakefile`, you will be asked to select a destination for the `fontdesc` file. Pick one now, and copy `SeeTeX/fontdesc-example` to that destination and edit the file.

Note: *The remainder of this section is taken from the MC-TeX library documentation on the fontdesc file.*

The font description file provides a sequence of directories to search for fonts. A short fontdesc is shown in Table 1 (although you should probably start with the longer one in `misc/fontdesc`). Normally, all fonts are stored in a directory hierarchy. The top-level directory contains the `.tfm` files for all fonts and a subdirectory for each font containing all the magsteps for that font. The fontdesc file also determines which font storage formats are used. The library is capable of reading all three standard TeX font formats (PK, GF, and PXL). The PK format is the best of the three. GF files are the slowest to read, and take an intermediate amount of disk space. PK files are the smallest, and intermediate in speed. PXL files are the fastest to read, but consume about six times as much storage as PK fonts. For a reasonable set of fonts, this makes the difference between 10 megabytes and 60 megabytes: the space difference is great, and the speed difference is only marginal. In addition, PXL formats are now considered obsolete and support for them may be dropped in the future without warning.

In addition to the three standard font formats, the library supports several 'special' font formats. They all work by reading a TFM (TeX font metrics) file. The `tfm` font format (also known as `invis`) carries only width information. It is intended to be used for printer-resident fonts, such as those that can be used by the PostScript driver. Since it has no rasters, it prints out as blank space, and so it can also be used in its `invis` guise in place

```
#       TYPE    SPEC    SLOP    PATH
font    pk      *       2       %f.%mpk

fontenv pk      *       2       TEXFONTS        %f.%mpk

font    pk      *       2       /usr/local/lib/tex/fonts/%f/%f.%mpk
font    gf      *       2       /usr/local/lib/tex/fonts/%f/%f.%mgf

font    box     *       0       %f.tfm
font    box     *       0       /usr/local/lib/tex/fonts/%f.tfm
```

Table 1: A Sample `fontdesc`

of SLITEX 'invisible' fonts, to save disk space. Printouts that use the invisible font format may have placement errors (of up to *maxdrift* pixels. The `blank` font format also prints out as blank space, but carries with it a warning that the desired font magnification was not found. The `box` format prints out as little rectangular boxes, rather like those in Chapter 11 of *The TEXbook*, showing where TEX thinks each character went. Like blank fonts, it carries a warning. The blank and box formats are intended as 'last resorts'; they exist only so that a `DVI` file that uses fonts in unavailable magnifications can still be printed (with the offending glyphs missing). If neither box nor blank fonts appear in the font description file, such files cannot be printed at all.

TFM fonts will always match at any size. They must therefore appear late in the fontdesc file, to keep them from 'taking over' other fonts whenever that is not desired. Of course, when it *is* desired, such as for built-in POSTSCRIPT fonts, it should appear before other fonts so that it will indeed 'take over'. If nothing else, this will improve performance (since the conversion programs will not have to hunt for nonexistent font files).

There are two kinds of font definition lines, beginning with the words `font` and `fontenv`. (Lines that start with any other word are ignored; all other words are reserved for future expansion.) The `fontenv` form differs only slightly from that of `font`, and its use will be described in a moment. For a `font`, there should be four more words following the word `font`. They are interpreted as follows:

type This is one of the seven font types (`blank`, `box`, `gf`, `invis`, `pk`, `pxl`, `tfm`).

spec This is a *print engine specifier*. TEX fonts come in several varieties. The major kinds are 'write black' and 'write white'. Each `DVI` driver has a print engine specifier (some have several); only if the program's specifier matches that in the font description will that font be used. Thus, if you have both a Canon-based (write-black) printer and a Ricoh-based (write-white) printer, you can put the write-black and write-white fonts in different subdirectories and make sure that each driver gets only the kind of fonts it uses. An asterisk '*' in the font description file will match any print engine specifier; this will suffice for many installations.

slop The slop factor is used to make up for the sins of finite-precision arithmetic. Fonts come in various magnifications, normally denoted as *magsteps*. Each magstep corresponds to a power of 1.2. For instance, `\magstep4` is 1.2^4, or 2.0736. These numbers are rounded to the nearest .001, so that this becomes 2.074. Font files are named according to this magnification multiplied by the number of dots per inch for that font.[2] For instance, the GF file for Computer Modern Roman, ten point, at `\magstep4` on a 300 dpi printer, when both the `DVI` file and user magnifications are 1000, is called `cmr10.622gf`. The 622 comes from the formula $300 \cdot 2.074 \cdot 1000/1000 \cdot 1000/1000 = 622.2$. In the course of all this arithmetic, the computer has a tendency to get the last digit wrong. The slop factor is used to correct for this. The font library will look for magnifications of up to $\pm$slop away from the number it computed. Thus, a slop of 1 will allow an off-by-one error to slip by. A slop of 0 requires an exact match. Since fake fonts—blank, box, and invis—have no inherent magnification, they should be given no slop; others should have as little slop as possible (the more there is, the slower things get).

path The library builds the path based on a *format* specifier. Characters other than the percent sign '`%`' are copied literally. The percent sign introduces a *conversion*, similar to those in the C library 'printf' routines. A single percent sign can be obtained by doubling it, `%%`. The other conversions are as follows (the examples in parentheses are for a font like `cmr10 scaled \magstep4` on a 300 dpi printer): `%f`, which gets the name of the font (`cmr10`); `%m`, which gets the magnification (`622`); and `%b`, which gets the *base name* of the font, like `%f`, but omitting any numeric suffix (`cmr`). These three suffice to express most font organisations. More formats may be added, if necessary, by changing `lib/font.c` (see the routine `pave()` [3]).

A `fontenv` is almost exactly the same as a `font`, except that the fifth word is the name of an environment variable rather than an actual path format, and that a sixth word (the *name part*) is required. If the named variable is present in the environment when the fontdesc file is being read, it is split up like any other UNIX path (i.e., at colons) and each sub-pathname has the name part appended. Each result is used as if it were the path portion of a regular `font`. For instance, given the example fontdesc above, if `$TEXFONTS` were set to `/here:/there`, the line

```
fontenv pk      *       2       TEXFONTS        %f.%mpk
```

would make MC-TEX programs act as if the pair of lines

```
font    pk      *       2       /here/%f.%mpk
font    pk      *       2       /there/%f.%mpk
```

were included instead. Typically, the name part of each `fontenv` will simply be `%f.%m`*type*. All TFM-based fonts, however, must have `%f.tfm` appended instead, so that, e.g., a `fontenv invis` for `iamsss8` looks for `iamsss8.tfm` rather than `iamsss8.0invis`.

[2] For PXL files, the number is then multiplied by 5, for reason which are neither good nor worth explaining.

[3] So named because it takes an abstract path (with percent escapes) and makes it concrete, 'paving it over' (as it were). Well, I have been told never to explain a pun....

The startup time of MC-TeX drivers is heavily influenced by the amount of directory searching done. A long fontdesc file, or a fontdesc file that uses long environment paths, can slow these programs severely. A `fontenv` whose variable is not present in the environment is ignored, at essentially zero cost in speed, so one way to reduce startup time is to use a different environment variable for each kind of font. For instance, by looking for PK fonts in `$PKFONTS` and for GF fonts in `$GFFONTS`, users with only GF fonts can avoid one of the searches entirely. In all cases, the total amount of searching can also be reduced by using a minimal 'slop' value.

Searching stops as soon as a usable font is found, so putting likely candidates first is also a good idea. User directories tend to be less likely places to find font files, so moving the main system directory to the first line will help. Unfortunately, this also makes it hard for users to override the standard font files—they will have to use different names for the fonts, or set `$TEXFONTDESC` to name a different fontdesc file. [4] A reasonable compromise can be had by using a `fontenv` line to refer to an unlikely environment variable name. Those users who *want* to override the standard font files can then set this variable, and everyone else will enjoy faster font lookups.

4 Configuring & Installing Basic SEETeX Programs Using IMAKE

To install the basic programs and libraries, you need to edit `SeeTeX/Imakefile` and install the font description file, `fontdesc`. §3 describes the latter.

Edit the file `SeeTeX/Imake.Config`. This file contains configuration parameters you need to tweak. Look for the following lines:

```
FontDesc=/usr/local/lib/tex82/fontdesc
IprBin=/usr/local/bin
```

The `FontDesc` variable names the location of a file, called `fontdesc`, that describes the location of TeX fonts on your machine. This file is usually in your standard TeX library location; edit `FontDesc` to point to your local library. If you are only going to use XTEX (and not MFTOBDF) you do not need to worry about `FontDesc`.

The `IprBin` program indicates where the `ipr` program lives; this is an interface to Imagen printers. If you will not be using IPTEX, ignore this line; otherwise, change `IprBin` to point to the `ipr` binary.

[4] The natural consequence of the latter is that, some years later, the fonts will be moved to some other directory, and the system fontdesc updated, and then several irate users will complain that the software is broken, having forgotten that they copied and altered the—now out of date—fontdesc file. The administrator will demonstrate that the software works; but the users will insist it does not. After several hours of probing about, someone will finally recall this footnote, and the mystery will at last be explained. In another few years, the cycle will then repeat.

The directories specified in `XTEX_FONT_PATH` are added to the X server font path on each invocation of XTEX. This simplifies the use of XTEX.

The optional command line specified in `Xtex*.makeFont` is used to invoke a *font creation program*. If an X version of a TeX font can not be found, XTEX invokes this program to create the font and install it on the X sever. Currently, only local servers (via XTeXtoX-font.SCRIPT in the Xtex directory) are supported; i.e. you must be able to directly manipulate the fonts on your current computer. However, file XTeXtoXfont.remote.SCRIPT gives the start for invoking the command on a remote host. If someone finishes this off, please send me mail. To properly configure the **makeFont** command, you must change the lines marked `X_FONT_SUFFIX`, `X_FONT_COMPILER`, `X_FONT_MAKE_DIR`, `X_FONT_COMPRESS`, `X_FONT_PREPEND`, and `X_FONT_RESET`.

The only tricky one is `X_FONT_COMPILER`, because e.g., the DEC **dxfc** and standard **bdftosnf** both write their data to standard out, while the OpenWindows **convertfont** needs to specify the output file. Thus, you need to assume that the input is in the shell environment variable `$BDFNAME` and the output should go to `$SNFNAME` (this variable will be using the correct suffix when it's done). See the standard entries in Imake.Config; you probably just need to select one of the existing ones, not change them.

Look in `SeeTeX/Imakefile` for the line:

```
SUBDIRS = libtex Xtex Mftobdf Dviselect Texx Iptex Fonts Texsun
```

Remove the names of any programs from the `SUBDIRS` line that you do not wish to build. You must include **libtex**, because it is used by all programs. TEXSUN still depends on SunView, and is not (yet) compatible with the more recent OpenLook X-windows interface.

Once you have changed `Imakefile`, execute the following commands: [5]

```
[SeeTeX/] xmkmf
[SeeTeX/] make Makefile
[SeeTeX/] make Makefiles
```

Throughout this document, the command prompt, e.g., `[SeeTeX/]`, indicates the directory within which the command should be executed. The variable `TOP` should point to your X11 library directory; this is usually installed in `/usr/local/X11`, but may be installed in different places. If you get error messages from **imake**, ask your local X11 guru where the configuration directory lives.

If you're not installing XTEX, you should be able to run

```
[SeeTeX/] make
```

at this point, or alternatively,

```
[SeeTeX/] make install
[SeeTeX/] make install.man
```

if you're confident that everything is correctly configured. This will build the library (LIBTEX) and each selected program.

[5] N.B. If you don't have `xmkmf` or Imake installed, reread §1.1 to see how to build the Makefiles.

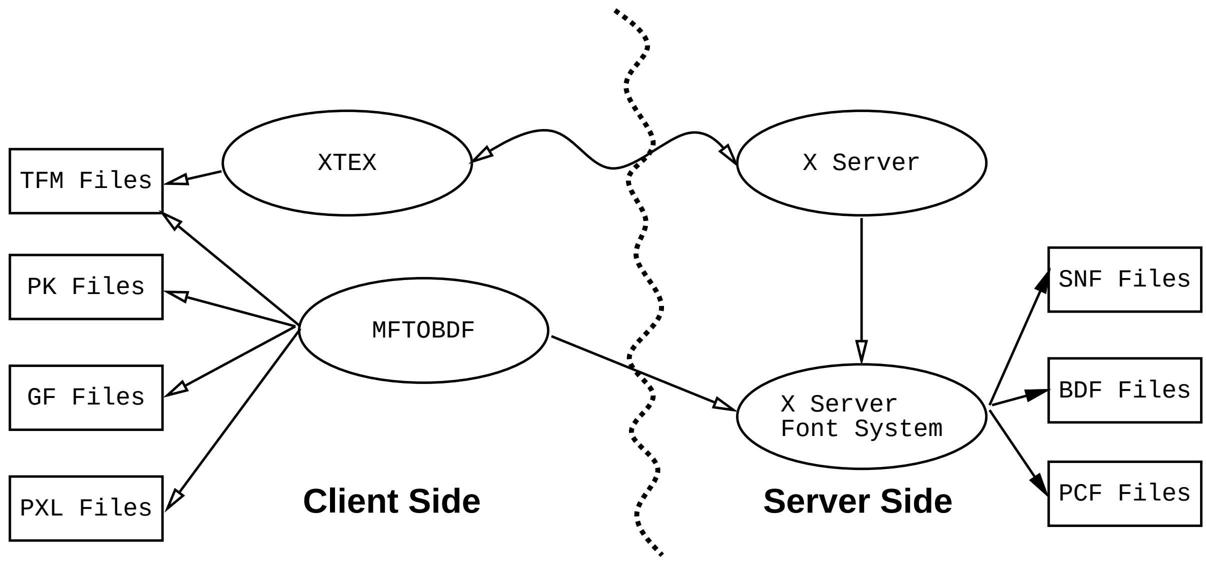

Figure 2: Schematic Usage of Fonts in X11 windows

5 A Little Explanation of X11 Fonts

XTEX opens another can of worms, because it uses X11 fonts that are created from TEX fonts via MFTOBDF. First, we must digress to talk about X11 fonts. X11 is a *network window system*; *clients* send messages to *servers* across a network, causing the server to draw things on the display. XTEX is a *client*. The *server* runs on your workstation, PC or terminal.

XTEX displays TEX output by telling the server to display specific fonts within a window. Those fonts correspond to TEX fonts, but they, like all X11 fonts, must live on the server. Figure 2 shows a schematic representation of the information needed and provided by each program. XTEX uses information from the `TFM` file as well as the X11 font to adjust the placement of individual letters on the display to match the proper TEX output within a single pixel of your display. If your server does not have the fonts that XTEX asks for, it will use a *default font* in their stead. This will look messy, but you'll still be able to preview your document. MFTOBDF converts existing *bitmap* fonts to `BDF` files, which can then be installed on your X11 window system "font system". For workstations, this is usually your normal filesystem or a shared filesystem.

5.1 Installing X11 Fonts

Typically, fonts are pre-installed in one of several directories that are listed in the server *font path*. Each directory in the font path normally contains a file, usually called `fonts.dir`,[6]that

[6]All of this "usually" and "normal" stuff is there for a reason. For example, Ultrix UWS2.0 on a VAXstation I previously used didn't use a `fonts.dir` file, and scanned all the files in each directory every time it started X. Very slow, to be sure. Sun's OpenWindows X11/NeWS server uses a `Families.list` file instead of `fonts.dir`.

maps between *font names* and *file names*. For example, the font named `cmr10.85` might live in the any of the files `cmr10.85.bdf`, `cmr10.85.snf` or `cmr10.85.snf.Z`. The `fonts.dir` file tells the X11 server where to find the font with the given name.

X11 fonts are distributed in BDF files, or *Bitmap Distribution Format* files; however, individual servers convert these portable BDF files to *server normal format*, or SNF files. Most servers can read:

1. raw BDF files (e.g., `cmr10.85.bdf`)
2. compressed BDF files (e.g., `cmr10.85.bdf.Z`)
3. raw SNF files (e.g., `cmr10.85.snf`)
4. compressed SNF files (e.g., `cmr10.85.snf.Z`)

The SNF files are *architecture specific*; in other words, you may not be able to use the same set of SNF files on all of your systems. I've found that SNF files can be shared between Sun-3 and Sun-4, but not e.g., the Sun-i386.

Certain X11 servers, e.g., DEC X11 servers, use the PCF format rather than SNF; these servers can read files like `cmr10.85.pcf` or `cmr10.85.pcf.Z`. Unlike SNF files, PCF files are *architecture independent*; a single set of PCF files suffice for all servers that can read the PCF format. Sadly, this appears to be limited to the DEC X11 servers.[7]

Sun's OpenWindows X11/NeWS server uses a different format, FB (font bitmap), for bitmap fonts.

Let's assume you've been given a BDF file describing a TeX font, and that file is called `cmr10.85.bdf`. To install this new font for X11, follow these steps:

1. First, convert the BDF file to either a SNF file (stock MIT X11 windows), a PCF file (DEC), or a FB file (Sun OpenWindows).

   SNF file : use the `bdftosnf` program that should come with your X11 windows system. For example, if you have the file `cmr10.85.bdf` execute

   ```
   bdftosnf cmr10.85.bdf > cmr10.85.snf
   ```

   PCF file : use the `dxfc` program that should come with your X11 windows system. For example, if you have the file `cmr10.85.bdf` execute

   ```
   dxfc cmr10.85.bdf > cmr10.85.pcf
   ```

   FB file : use the `convertfont` program that should come with your OpenWindows system. For example, if you have the file `cmr10.85.bdf` execute

   ```
   convertfont -f 32 cmr10.85.bdf
   ```

[7] I had made a brief stab at having the X server load PK file directly. It's mostly complete, and if you're interesting in finishing that work, send me mail.

2. Second, decide if you want to compress the `SNF/PCF` file. This does not apply for `FB` files. For fast workstations, like SPARC or DEC/RISC systems, you should probably compress the output file using `compress`; you save a lot of disk space and won't notice the time spent uncompressing the file.

3. Now, pick a directory in which you want to install the font and copy the font to that directory. You can either use a directory already on the X11 font path, or you can make a new directory and add that your font path.

   If you make a new directory, you must add it to your font path using the `xset` program, e.g. if you added the directory /usr/lib/X11/fonts/tex, you should say:

   ```
   [unix] xset +fp /usr/lib/X11/fonts/tex
   ```

   every time you start your X11 *or* X11/NeWS *server.* You can optionally change your X11 startup scripts to always include this font path.

   > `N.B.` The stock DEC X11 does not include include `xset`, but you can put the `PCF` files in either `/usr/lib/X11/fonts/local/75dpi` if you use a 19 inch monitor or `/usr/lib/X11/fonts/local/100dpi` if you use a smaller monitor. These directories are in your default font path.

4. You must rebuild the `fonts.dir` or `Families.list` file in the directory where you placed that font. Go to the directory and execute either `mkfontdir` (stock MIT X11 windows), `dxmkfontdir` (DEC X11 windows), or `bldfamily -f 32` (Sun OpenWindows).

   This will take a while, because each file in the directory is examined to see if it's a font, and the font name is added to the font directory file.

5. Now, tell your server that the font directories have changed, and that it should rescan the font directory files. Do this using `xset fp rehash`.

6. Lastly, check that the font was properly installed. The `xlsfonts` program lists all the fonts your server knows about. So, for our sample font, called `cmr10.85`, execute `xlsfonts | grep cmr10.85` and see if it's listed. You can also execute `xfd -fn cmr10.85` to see the font.

If you have a bevy of `BDF` files to convert, you may want to use the shell scripts `BDFZtoSNFZ` or `BDFZtoPCFZ` to quickly convert compressed `BDF` files to either compressed `SNF` or `PCF` files. These shell scripts are distributed with XTEX in the `SeeTeX/Fonts` directory.

Currently, host `foobar.colorado.edu` provides a large set of `BDF` files for 85dpi fonts, rendered at various magnifications. You can `FTP` those files or generate your own `BDF` files using MFTOBDF; see §7 .

6 Installing XTEX

Given all of this, the installation of XTEX is fairly simple.

The first thing you must do is decide what sizes of fonts you're going to be using. As delivered, XTEX defaults to 85dpi fonts. The two options are either screen resolution fonts (e.g., the 85dpi fonts) generated by METAFONT specificly for XTEX, or 100dpi fonts "shrunken" from existing printer fonts, usually 300dpi or 400dpi.

The advantage of the screen resolution fonts is that they're easier to read. The disadvantage is that you may need to use METAFONT to build the fonts at those sizes.

The advantage of shrunken printer fonts is that it's easy to build them, because you probably already have them. The disadvantage is that they look kind of crappy and are harder to read.

1. Edit file `SeeTeX/Xtex/Xtex.ad`, the application defaults file, and check the line that looks like:

   ```
   *.tfmPath: .:/usr/local/lib/tex82/fonts:/usr/local/lib/tex/fonts:
   /usr/lib/tex82/fonts:/usr/lib/tex/fonts:/usr/local/lib/tex/tfm:
   /usr/local/Tex/lib/tex/tfm
   ```

   This describes the path that XTEX will use to locate `TFM` files. If your default `TFM` location is not in that path, add it. An alternative is to set the `TEXFONTS` environment variable to a list similar to this. This environment variable is also used by TeX.

2. If you want to use something other than the default 85dpi fonts, edit `SeeTeX/Xtex/Xtex.ad` and comment in the following lines, and remove the lines that follow them.

   ```
   # *.mag:333
   # *.smallMag:333
   # *.largeMag:500
   # *.dpiHoriz:300
   # *.dpiVert:300
   ```

3. You may also want to look through the rest of `SeeTeX/Xtex/Xtex.ad` and edit things to your liking. Users can override the defaults using their own resource files.

4. Edit `SeeTeX/Imake.Config`.

   (a) If your system provides Adobe Display Postscript, include the line

   `#define DISPLAY_POSTSCRIPT`

   otherwise, include the line

   `#undef DISPLAY_POSTSCRIPT`

If you have Display Postscript, but are not using the DEC version thereof, you may need to fiddle with the Imakefile.

(b) XTEX uses several *popup windows* to ask you things; by default,

these are `overrideShellWidetClass` windows.

Some people prefer `transientShellWidgetClass` windows, because they can be moved around by the user. Select whichever one you prefer.

Now, return to the top level of the directory tree and execute:

```
[SeeTeX/] make Makefiles
[SeeTeX/] make install
```

Note: you cannot "test drive" XTEX. It depends very heavily on the application defaults file, `Xtex.ad` being installed in the proper place, typically `/usr/lib/X11/app-defaults/Xtex`. If this file is not installed, or if an older version is installed, you will get diagnostic messages and xtex will refuse to run.

7 Installing and Using MFTOBDF

Installing MFTOBDF is relatively stock, and it probably built just fine if you followed the steps in §4 . The hard part is using it to build BDF files. First, read §2 and §5 and §5.1 . Second, determine if you're going to use screen-resolution fonts or shrunken fonts (see §6).

7.1 Using Screen Resolution Fonts

There are three ways to get screen resolution fonts, the easy way, the really hard way and the hard way. Let's assume that you're going to generate fonts at XYZdpi, where XYZ might be 118, 85 or whatever.

7.1.1 The Easy Way

Decide that you want to use 85dpi fonts, and FTP to `foobar.colorado.edu`. Hunt around until you find a directory `Bdf-85dpi`. Go there, and grab all the fonts. Install them (see §5.1), and use them for a few weeks. Now, go to your font directory, and do

```
[unix] ls -lu | head -100  > KEEP-THESE
```

and then remove all the fonts not listed in the `KEEP_THESE` file. Then, rebuild the `fonts.dirs` file. This gives you the top 100 fonts commonly used at your site. If you are missing some fonts, just FTP to foobar again, or read the next item.

7.1.2 The Hard Way

Lets assume that you have a slew of GF or PK fonts. Perhaps you picked up 118dpi fonts from a TeX library somewhere. You need to convert those fonts to BDF files. The easiest way to do this is to go to the directory holding those fonts and say:

```
[unix] mftobdf -dpi XYZ *gf *pk
```

and sit back. If you specify hundreds of fonts, MFTOBDF may barf, because there's a memory leak that I can't find and don't care about. You can also specify, e.g,

```
[unix] mftobdf -dpi XYZ -mag 1098 cmr10
```

to convert a single font. At 85dpi, this would be equivilent to saying

```
[unix] mftobdf -dpi 85 cmr10.93pk
```

Note that you must *always* specify the dots per inch you're using. After you've generated the BDF files, you can discard the PK or GF files, if you like. XTEX does not use those directly.

7.1.3 The Really Hard Way

What if you can't find a set of fonts at your desired resolution on the network? You need to build the font fonts using METAFONT, and then convert them to BDF format.

1. Find a METAFONT wizard. All you need to know is where your configuration files are, and if they can show you how to build a font, all the better.

2. Make certain that you have a `modedef` that looks something like:

   ```
   mode_def sun =                 % bitgraph and sun screens (true size)
    proofing:=0;                  % no, we're not making proofs
    fontmaking:=1;                % yes, we are making a font
    tracingtitles:=0;             % no, don't show titles in the log
    pixels_per_inch:=85;         % try it to see if it will do
    blacker:=.35;                 % make pens a bit blacker (Karl Berry's value)
    fillin:=.1;                   % and compensate for diagonal fillin
    o_correction:=.3;             % but don't overshoot much
    enddef;
   ```

   This is the `modedef` I used to generate my 85dpi fonts. Change the `pixels_per_inch` parameter to XYZ. You may want to fiddle with the other parameters as well. It's all shots in the dark.

3. Build a version of METAFONT that understands this modedef, i.e. use `inimf` to build a new `plain.base` file.

4. Build a sample font using, e.g.

   ```
   [unix] mf '\mode:=sun; mag:= 1098/1000; \batchmode; \input cmr10'
   ```

   This should give you a file, `cmr10.ABCgf`, where the suffix should be $1098 * \frac{XYZ}{1000}$. If it's not, it's probably because the `modedef` isn't being recognized; check that you're using the right `plain.base` file.

5. Obviously, the hard part is generating all the needed `GF` files. The directory `SeeTeX/Fonts` contains some utilities that makes all this easier. See the file `SeeTeX/Fonts/README`.

   One thing to remember is that you probably want fonts at a *normal* size and a *large* size. E.g., if you're looking at your document at 85dpi and normal magnification of 1000, you might want to "blow things up" to e.g., 1440 magnification (the default in XTEX) to see details a little better. Thus, not only do you want to generate e.g., `cmr10` at 1000, you also want to generate it at 1440. Happily, the numbers that TEX and LATEX use (1000, 1220, 1440, etc) scale by 1.2, and you typically don't need to generate a another complete set at the larger magnification, because you'll already have most of them.

Now, following the directions in §7.1.2 to convert these to `BDF` files.

7.2 Using Shrunken Fonts

Converting your existing fonts is relatively easy. Go to the directory containing your `PK` or `GF` fonts. Font shrinking, or scaling, is specified similar to magnifications; however, the only valid scalings are $1000/N$ where N is an integer. Thus, 500 and 333 are most common. For 300dpi fonts, these would give you roughly 150dpi and 100dpi versions of your printer fonts. In XTEX, 500 is normally used for the "large" magnification and 333 is used for small magnification.

So, you generate the fonts at scaling XYZ from fonts that were originally ABCdpi, say:

```
[unix] mftobdf -dpi ABC -mag 1000 -scaled XYZ *pk *gf
```

The `-scaled` keyword is the important thing here. After you've gotten a passle of `BDF` files from all of this, convert them to your local server format and install them according to the directions in §5.1 .

7.3 Installing Fonts On The Fly

With version 2.17, you can now install fonts when XTEX determines they're not available. You need to change Imake.Config, following the directions mentioned there. There are currently some limitations to this mechanism. It basically does what either of the two preceeding subsection do, but on a per-font basis. However, only "workstations" are currently supported, i.e. I assume that the XTEX program is executing on the X-server that is previewing the DVI file needing the font.