Font HOWTO

Table of Contents

Table of Contents

Table of Contents

Font HOWTO

Donovan Rebbechi, elflord@panix.com

6.True Type to Type1 Conversion

7.WYSIWYG Publishing and Fonts

8.Netscape

9.TeX / LaTeX

10.Getting Fonts For Linux

11.Useful Font Software for Linux

12.Ethics and Licensing Issues Related to Type

13.References

14.Glossary

1.Introduction

1.1 The Location of This Document

This document is located at my webpage

1.2 Rationale

I'm going to make this brief (−; The purpose of this document is to address what was a gaping hole in font documentation. While previously, there were several documents about fonts for Linux, I felt that none of them were comprehensive, all of them had a relatively narrow focus. Hence the goal of this document is not to provide radical new insight into the issue of font handling (though I have included material previously unavailable), the main aim is to provide a comprehensive source to act as a starting point for any and all font questions about Linux.

1.3 Credits and Acknowledgements

Special thanks are due to Rod Smith, who made several helpful suggestions, and fixed pfm2afm. Also thanks to Doug Holland, whose Font Deuglification HOWTO proved to be a good reference, John McLaughlin, author of the document that helped me come to grips with Star Office, and the Linux community for their suggestions and construtive comments.

2.Fonts 101 −− A Quick Introduction to Fonts

2.1 Types of fonts

Bitmap Fonts

A bitmap is a matrix of dots. Bitmap fonts are represented in precisely this way −− as matrices of dots. Because of this, they are *device dependent* −− they are only useful at a particular resolution. A 75 DPI screen bitmap font is still 75 DPI on your 1200 DPI printer.

There are two types of bitmap fonts −− bitmap printer fonts, such as the `pk` fonts generated by dvips, and bitmap screen fonts, used by X and the console. The bitmap screen fonts typically have a `bdf` or `pcf` extension. Bitmap screen fonts are most useful for terminal windows, consoles and text editors, where the lack of scalability and the fact that they are unprintable is not an issue.

Type1 Fonts

The Type1 font standard was devised by Adobe, and Type1 fonts are supported by Adobe's postscript standard. Because of this, they are also well supported under linux. They are supported by X and ghostscript. Postscript fonts have traditionally been the choice of font for anything on UNIX that involves printing.

Typically, a UNIX Type1 font is distributed as an `afm` (adobe font metric) file, and an outline file, which is usually a `pfb` (printer font binary) or `pfa` (printer font ascii) file. The outline file contains all the glyphs, while the metric file contains the metrics.

Type 1 fonts for other platforms may be distributed in different formats. For example, postscript fonts for windows often use a different format (`pfm`) for the metric file.

Type3 fonts

These fonts are distributed in a similar manner to Type1 files -- in groups of `afm` font metrics, and `pfa` files. While they are supported by the postscript standard, they are not supported by X, and hence have limited use.

TrueType fonts

True type fonts were developed by Apple. They made the format available to Microsoft, and succesfully challenged Adobe's grip on the font market. True type fonts store the metric and shape information in a single file (usually one with a `ttf` extension). Recently, font servers have been developed that make TrueType available to X. And postscript and ghostscript have supported TrueType fonts for some time. Because of this, TrueType fonts are becoming more popular on linux.

Type42 Fonts

Type42 fonts are actually just TrueType fonts with headers that enable them to be rendered by a postscript interpreter. Most applications, such as ghostscript and SAMBA handle these fonts transparently. However, if you have a postscript printer, it may be necessary to explicitly create Type42 font files.

Type1 vs TrueType -- a comparison

Despite the historical feuding between the proponents to Type1 and TrueType fonts, both have a lot in common. Both are scalable outline fonts. Type1 fonts use cubic as opposed to quadratic curves for the glyphs. This is in theory at least a slight advantage since they include all the curves available to TrueType fonts. In practice, it makes very little difference.

TrueType fonts have the apparent advantage that their support for hinting is better (Type1 fonts do have hinting functionality, but it is not as extensive as that of TrueType fonts). However, this is only an issue on low resolution devices, such as screens (the improved hinting makes no discernable difference on a 600dpi printer, even at small point sizes.) The other point that makes this apparent advantage somewhat questionable is the fact that well hinted TrueType fonts are rare. This is because software packages that support hinting functionality are out of the budget of most small time designers. Only a few major foundries, such as Monotype make well hinted fonts available.

In conclusion, the main differences between TrueType and Type1 fonts are in availability and application support. The widespread availability of TrueType fonts for Windows has resolted in webpages designed with the assumption that certain TrueType fonts are available. Also, many users have large numbers of TrueType fonts because they ship with the users Windows applications. However, on Linux, most applications support Type1 fonts but do not have the same level of support for TrueType. Moreover, most major font foundries still ship most of their fonts in Type1 format. For example, Adobe ship very few TrueType fonts. My recommendation to users is to use whatever works for your application, and try to avoid converting from one format to another where possible (because the format conversion is not without loss).

Metafont

Metafont was developed by Donald E Knuth as part of the TeX typesetting system. Metafont is a graphics programming language (like postscript) that has applications wider than just fonts. Metafonts exhibit some very desirable qualities. One of the important features is that metafonts can scale very gracefully. The metafont Computer Modern has different shape at 20 point and 10 point. The shape changes with size, because it is desirable for a smaller font to be proportionately wider than a larger font (this makes the larger fonts more elegant and the smaller font more readable).

Metafonts typically have a `mf` extension. They are rendered to device dependent bitmap fonts. The rendering is slow, so they are of excellent quality, but are not well suited to WYSIWYG publishing.

2.2 Families of Typefaces

Typically typically come in groups of a few variants. For example, most fonts come with a bold, italic, and bold−italic variant. Some fonts may also have small caps, and demibold variants. A group of fonts consisting of a font and its variants is called a *family* of typefaces. For example, the Garamond family consists of Garamond, Garamond−italic, Garamond−bold, Garamond bold−italic, Garamond demi−bold, and Garamond demi−bold−italic. The Adobe expert Garamond font also makes available Garamond small caps, and Garamond titling capitals.

3.Fonts 102 −− Typography

Here, we discuss some typography basics. While this information is not essential, many font lovers will find it interesting.

3.1 Classifications of Typefaces

Fixed versus variable width

There are several classifications of typefaces. Firstly, there are `fixed width` fonts, and variable width fonts. The fixed width fonts look like typewriter text, because each character is the same width. This quality is desirable for something like a text editor or a computer console, but not desirable for the body text of a long document. The other class is variable width. Most of the fonts you will use are variable width, though fixed with can be useful also (for example, all the example shell commands in this document are illustrated with a fixed with font). The most well known fixed width font is courier.

To serif or not to serif ?

Serifs are little hooks on the ends of characters. For example, the letter i in a font such as Times Roman has serifs protruding from the base of the i and the head of the i. Serif fonts are *usually* considered more readable than fonts without serifs. There are many different types of serif fonts.

Sans serif fonts do not have these little hooks, so they have a starker appearance. One usually does not write a long book using a sans serif font for the body text. There are sans serif fonts that are readable enough to be well suited to documents that are supposed to be browsed / skimmed (web pages, catalogues, marketting brochures). Another application that sans serif fonts have is as display fonts on computer screens, especially at small sizes. The lack of detail in the font can provide it with more clarity. For example, Microsoft touts Verdana as being readable at very small sizes on screen.

Notable sans serif fonts include Lucida sans, MS Comic Sans, Verdana, Myriad, Avant Garde, Arial, Century Gothic and Helvetica. By the way, Helvetica is considered harmful by typographers. It is somewhat overused, and many books by typographers plead users to stay away from it.

The old and the new −− different types of Serif fonts

Old Style

Old style fonts are based on very traditional styles dating as far back as the late 15th century. Old style fonts tend to be conservative in design, and very readable. They are well suited to writing long documents. The name ``old style'' refers to the style of the font, as opposed to the date of its design. There are classic old style fonts, such as Goudy Old Style, which wre designed in the 20th century. The old style class of fonts has the following distinguishing features:

- Well defined, shapely serifs.
- Diagonal emphasis. Imagine drawing a font with a fountain pen, where lines 45 degrees anticlockwise from vertical are heavy and lines 45 degrees clockwise from verticle are light. Old style fonts often have this appearance.
- Readability. Old style fonts are almost always very readable.
- Subtlety and lack of contrast. The old style fonts have heavy lines and light lines but the contrast in weight is subtle, not stark.

Notable Old Style fonts include Garamond, Goudy Old Style, Jenson, and Caslon (the latter is contentious −− some consider it transitional)

Moderns (or didone)

The moderns are the opposite of old style fonts. These fonts typically have more character, and more attitude than their old style counterparts, and can be used to add character to a document rather than to typeset a long piece. However, nothing is black and white –– and there are some modern fonts such as computer modern and Monotype modern, and New Century Schoolbook are very readable (the contrast between heavy and light is softened to add readability). They are based on the designs popular in the 19th century and later. Their distinguishing features include:

- Lighter serifs, often just thin horizontal lines.
- Vertical emphasis. Vertical lines are heavy, horizontal lines are light.
- Many moderns have a stark contrast between light and heavy strokes.
- Modern typefaces with high contrast between light and heavy strokes are not as readable as the old style fonts.

Bodoni is the most notable modern. Other moderns include computer modern, and Monotype modern (on which computer modern is based).

Transitional

Transitional fonts fit somewhere in between moderns and old style fonts. Many of the transitionals have the same kind of readability as the old styles. However, they are based on slightly later design. While a move in the direction of the moderns may be visible in these fonts, they are still much more subtle than the the moderns. Examples of transitionals include Times Roman, Utopia, Bulmer, and Baskerville. Of these, Times leans towards old style, while Bulmer looks very modern.

Slab Serifs

The slab serif fonts are so named because they have thick, block like serifs, as opposed to the smooth hooks of the old styles or the thin lines of some of the moderns. Slab serif fonts tend to be sturdy looking and are generally quite readable. Many of the slab serifs have Egyptian names –– such as Nile, and Egyptienne (though they are not really in any way Egyptian). These fonts are great for producing readable text that may suffer some dilution in quality (such as photocpied documents, and documents printed on newspaper). These fonts tend to look fairly sturdy. The most notable slab serif fonts are Clarendon, Memphis and Egyptienne, as well as several typewriter fonts. Many of the slab serif fonts are fixed width. Conversely, most (almost all) fixed width fonts are slab serif.

The Sans Serif Revolution

Surprisingly, the rise of sans serif fonts is a fairly recent phenomenon. The first well known sans serif fonts were designed in the 19th early 20th century. The earlier designs include Futura, Grotesque and Gill Sans. These fonts represent respectively the ``geometric'', ``grotesque'' and ``humanist'' classes of sans serif fonts.

Grotesque

The grotesques where so named because the public were initially somewhat shocked by their relatively stark design. Groteques are very bare in appearance due to the absence of serifs, and the simpler, cleaner designs. Because of their ``in your face'' appearance, grotesques are good for headlines. The more readable variations also work quite well for comic books, and marketting brochures, where the body text comes in small doses. Grotesques don't look as artsy as their geometric counterparts. Compared to the geometrics, they have more variation in weight, more strokes, they are squarer (because they don't use such circular arcs). They use a different upper case G and lower case a to the geometrics. While they are minimalistic but don't go to the same extreme as the brutally avant−garde geometrics.

Notable grotesques include the overused Helvetica, Grotesque, Arial, Franklin Gothic, and Univers.

Geometric

The Futura font came with the manifesto: *form follows function*. The geometric class of fonts has a stark minimalistic appearance. Distinguishing features include a constant line thickness (no weight). This is particularly conspicuous in the bold variants of a font. Bold groteques and humanist fonts often show some notable variation in weight while this rarely happens with the geometric fonts. Also notable is the precise minimalism of these designs. The characters almost always are made up from straight horizontal and vertical lines, and arcs that are very circular (to the point where they often look as though they were drawn with a compass). The characters have a minimal number of strokes. This gives them a contemporary look in that they embrace the minimalistic philosophy that would later take the world of modern art by storm. A tell tale sign that a font is a geometric type is the upper case ``G'', which consists of a minimalistic combination of two strokes −− a long circular arc and a horizontal line. The other character that stands out is the lower case ``a'' −− which is again two simple strokes, a straight vertical line and a circle (the other ``a'' character is more complex which is why it is not used). Notable geometrics include Avant Garde, Futura, and Century Gothic.

Humanist

As the name might suggest, humanist fonts were designed with a goal of being less mechanical in appearance. In many ways, they are more similar to the serif fonts than the geometrics and the grotesques. They are said to have a ``pen drawn'' look about them. They tend to have subtle variation in weight, especially observable in bold variants. The curve shapes are considerably less rigid than those of the geometrics. Many of them are distinguishable by the ``double story'' lower case g, which is the same shape as the g used in the old style serif fonts. The humanist typefaces are the easiest to use without producing an ugly document as they are relatively compatible with the old style fonts.

Compatible Typefaces

Grouping typefaces is not easy, so it pays to avoid using too many on the one page. A logical choice of two typefaces consists of a serif and a sans serif. Monotype's Typography 101 page provides a category−matchup. They conclude that the moderns and geometrics form good pairs, while the old styles and humanists also go together well. The transitionals are also paired with the humanists. The slab serifs are paired with the grotesques, and some variants of the slab serifs are also said to match the geometrics or humanists.

From reading this, one gets the impression that their philosophy is essentially to match the more conservative serifs with the more moderate sans serifs, and pair the wilder modern serifs with the avant garde looking (pun unavoidable) geometrics.

3.2 Ligatures, Small caps fonts and expert fonts

Ligatures

Properly spacing fonts brings with it all sorts of issues. For example, to properly typeset the letters ``fi'', the i should be very close to the f. The problem is that this causes the dot on the i to collide with the f, and the serif on the head of the i to collide with the horizontal stroke of the f. To deal with this problem, font collections include ligatures. For example, the ``fi'' ligature character is a single character that one can substitute for the the two character string ``fi''. Most fonts contain fi and fl ligatures. Expert fonts discussed later often include extra ligatures, such as ffl, ffi, and a dotless i character.

Small caps fonts

Small caps fonts are fonts that have reduced size upper case letters in place of the lower case letters. These are useful for writing headings that require emphasis (and they are often used in LaTeX). Typically, when one writes a heading in small caps, they use a large cap for the beginning of each word, and small capitals for the rest of the word (``title case''). The advantage of this over using all caps is that you get something that is much more readable (using all caps is a big typographic sin).

Expert fonts

Expert fonts consist of several extras designed to supplement a typeface. These include things like ligatures, ornaments (much like a mini−dingbats collection designed to go with the typeface), small caps fonts, and swash capitals (fancy, calligraphic letters).

3.3 Font Metrics and Shapes

Font metrics define the spacing between variable width fonts. The metrics include information about the size of the font, and *kerning* information, which assigns kerning pairs −− pairs of characters that should be given different spacing. For example, the letters ``To'' would usually belong in a kerning pair, because correctly spaced (or kerned), the o should partly sit under the T. Typesetting programs such as LaTeX need to know information about kerning so that they can make decisions about where to break lines and pages. The same applies to WYWIWYG publishing programs.

In addition to the metrics, is the font outline, or shape. The components of the fonts shape (a stroke, an accent, etc) are called ``glyphs''.

4.Making Fonts Available To X

There are a number of ways fonts can be added to X. Firstly, XFree86 has a *font path* which is just a list of several directories or *font servers* where it searches for fonts. A font server is just a background process that makes fonts available to XFree86. An advantage of font servers is that they can send fonts to remote displays.

Recently, `xfs` (the ``X font server'') has been patched to support TrueType fonts, and run as a stand−alone program. The patched version ships with Redhat and Redhat−based distributions, and is included in XFree86 3.9.17 (the latest version at the time of writing. It will also be a part of XFree86 4.0) `xfs` is actually just the standard font server that comes with XFree86. It's source code is part of the XFree86 source tree. However, distributions have recently been shipping a version that runs in stand alone mode. The standalone X font server, with the TrueType support patch (the TrueType support takes place via a font server called `xfsft`) is probably the nicest font management solution currently available. Its advantages include:

- Support for different types of fonts, including Type1, TrueType and bitmap.
- Makes fonts available to remote displays.
- Greatly simplifies editing the fontpath −− you can do it via the command line utility `chkfontpath`, as opposed to having to edit configuration files. This not only makes life easier for users, it makes packaging more safer and more scriptable for packagers.

Because different distributions ship with different configurations, it is not true that one size fits all. We can split users up into three groups:

- Your distribution ships with a standalone `xfs` and it has been patched to support TrueType. This group includes Redhat users and users of derivatives of Redhat such as Mandrake, TurboLinux, and Independence. For this group, the wisest strategy is to install both TrueType and Type1 fonts through `xfs`
- Some distributions ship with a stand alone `xfs` package, but no TrueType support (at the time of writing. Note that XFree86 supports TrueType as of version 3.9.17). This includes Debian. For these users, the best thing to do is use `xfs` to install Type1 fonts, and install TrueType fonts via `xfstt`. Debian users can seek out the TrueType Fonts in Debian mini−HOWTO for information about installing TrueType fonts in Debian.
- If you don't have `xfs` then you will need to install Type1 fonts by adding to their XFree86 font path and using xset. You should install TrueType via `xfstt`.

4.1 The font path

XFree86 finds your fonts by searching a *font path*, a list of directories (or servers −− we'll explain this further later.) containing fonts. When an application requests a font, it searches through the directories in your font path one at a time until the font is found. To make fonts available requires you to set your font path. You can add a directory to your font path with the command

```
xset fp+ directory
```

Once you have done this, you need to ask the X server to re−scan for available fonts with the command

```
xset fp rehash
```

Since you will want these commands to run automatically, you should put them in your `.xinitrc` file (or possibly your `.Xclients` or `.xsession` file −− this depends on how you start X. It's convenient to make two of these files symlinks to the other to avoid confusion). Another way to have the commands set automatically is edit XF86Config. For example, to add `/usr/share/fonts/myfonts` to the fontpath when X is started, edit `XF86Config` like this:

```
...
Section "Files"
...

FontPath /usr/share/fonts/myfonts
```

```
...
EndSection
...
```

The advantage of editing XF86Config is that the resulting changes are system wide.

4.2 Installing Type1 Fonts

Run Type1inst

The easiest way to make Type1 fonts available to X is with the help of the Type1inst utility. This is a perlscript that automatically creates the fonts.dir and fonts.scale files that you need for X to use the fonts. Simply CD to the directory, and run type1inst.

```
cd directory
type1inst
```

If You Have the `xfs` Package

Now you need to add the fonts to your fontpath. If you already have the standalone xfs running, you do this by editing your `xfs` configuration file. *Redhat users can just use chkfontpath. the format is* `chkfontpath --add directory`

Your fonts should now be available to X. Now you just run

```
xset fp rehash
```

and X will be able to find the new fonts.

If You Don't Have The `xfs` Package

In this case, you need to add the directory containing your new fonts to the font path, as described previously.

4.3 True Type Fonts

Adding TrueType fonts is a little more difficult, because you need to have a font server that is capable of serving TrueType fonts. Two font servers that do this are xfstt and xfs.

`xfstt` is a TrueType font server. While it's easy to configure, and quite useful, it appear that `xfs` is becoming more popular. The main advantage of `xfs` over `xfstt` is that it supports both Type1 and TrueType fonts.

xfstt

To set up xfstt, just download it and install it. If you have an rpm based distribution, there is a well packaged version of xfstt at http://independence.seul.org/. Once you install it, you need to do the following:

1. install fonts into the appropriate directory (read the documentation that comes with the package).
2. cd to that directory and run `xfstt --sync`. This causes it to look for the fonts and create the `fonts.dir` file.
3. Now add `unix/:7100` to your font path.

Your TrueType fonts should now display and be available to applications such as GIMP and Netscape. You may want to configure it to start every time your system starts up. Check to see if there's a startup file included (if you are using RPM, you can use `rpm -ql xfstt |grep init` and look for the file with a name something like this: `/etc/rc.d/init.d/xfstt`) If you don't have an init script, just put two lines in `/etc/rc.local` like this:

```
/usr/X11R6/bin/xfstt --sync
/usr/X11R6/bin/xfstt &
```

4.4 xfs

Some of the newer Linux distributions ship with the X font server `xfs` configured to run as a stand alone program. Notably, Redhat and all the redhat based distributions use this modularised `xfs` with TrueType compiled in. Debian also ship `xfs`, but the version they ship doesn't have built in true type support.

Running `xfs` as a stand alone server has several benefits, especially if it is compiled with TrueType support. The main advantage is that since the font server is no longer attached to the X server, it is possible to serve fonts to remote displays. Also, it makes it much easier to modify the font path.

The `xfs` Path

As a font server, `xfs` has it's own font path. One might wonder where this fits into the picture. It works like this: you can place the `xfs` font server in XFree86's font path, by adding `unix/:port` to the XFree86 font path. Once you do this, any font in the `xfs` font path automatically becomes available to XFree86.

The `xfs` font path is determined by the `xfs` configuration file, which is `/etc/X11/fs/config` on Redhat, and `/etc/X11/xfs/config` on Debian. Redhat users do not need to explicitly edit this file, they can use the `chkfontpath` utility. The syntax is simple:

```
chkfontpath --add directory
```

Users of other distributions can edit the configuration file as follows:

```
catalogue = /usr/X11R6/lib/X11/fonts/misc:unscaled,
...
/usr/share/fonts/my_new_fonts/,
...
/usr/share/fonts/some_other_directory
# in 12 points, decipoints
default-point-size = 120
...
```

The above would add `/usr/share/fonts/my_new_fonts/` to the `xfs` font path. *Note that the last line of the list of directories doesn't have a comma at the end.* For these modifications to the font path to become effective, `xfs` must be restarted. It's also a good idea to restart your X session after restarting `xfs`.

Installing a Font Into `xfs`

To prepare a font for `xfs`, you need to follow the following steps:

- If you don't have xfs installed, you need to install it.
- Put the new fonts in a directory.
- If you are installing Type1 fonts, prepare the new directory for the server by running `type1inst` in the directory.
- If you are installing TrueType fonts, (*remember, not all distributions can do TrueType via xfs !*), prepare the new directory for the server by running

  ```
  ttmkfdir -o fonts.scale
  mkfontdir
  ```

  in the directory containing your new fonts. `ttmkfdir` is part of the *freetype* package.
- Now you can add the new directory to your `xfs` search path. Users of Redhat–like distributions can do this with the `chkfontpath` utility: Other users can do this by editing their `xfs` configuration file.
- if `xfs` is already installed on your system, you should see which port it is running on. You can do this as follows:

  ```
  ps ax|grep xfs
  ```

- Then check your XFree86 font path.

  ```
  xset -q
  ```

- If your font path includes something like `unix:/`*`port_number`* were *port_number* is the port which the server is running on, then you already have xfs set up properly. Otherwise, you should add it to your XFree86 font path.

  ```
  xset fp+ unix/:port_number
  xset fp rehash
  ```

  You can add it permanently by editing your `.xinitrc` as explained previously. To add it system wide, edit your XF86Config file (probably either `/etc/X11/XF86Config`, `/etc/XF86Config` or `/usr/X11R6/lib/X11/XF86Config`), by adding a line `FontPath "unix:/port_number"` in the Files section. Here's an example:

  ```
  ...
  Section "Files"
  ...

  FontPath "unix/:-1"
  ...
  EndSection
  ...
  ```

- If `xfs` is already properly installed, then you can restart it like this:

```
/etc/rc.d/init.d/xfs restart
```

- After restarting `xfs`, it's a good idea to restart your X–session.

5.Making Fonts Available To Ghostscript

To make fonts available to ghostscript, it suffices to tell ghostscript where the files corresponding to a given font are located. The file that needs to be edited is `/usr/share/ghostscript/version/Fontmap`. The format is very simple, almost immediately self evident on perusing it.

5.1 Type1

Adding Type1 fonts is straightforward. Run `type1inst` on the directory containing the font. `type1inst` will output a file called `Fontmap`. Append this file to the ghostscript `Fontmap` file.

5.2 True Type

Adding truetype fonts is a little trickier, because we have to get the name of the TrueType font. One way (brute force, alas) to do this is using the `ttf2pt1` TrueType to Type1 converter, and grabbing the font name from the `afm` (there's got to be a more efficient way ! but this works, ugly as it is). You do it like this:

```
ttf2pt1 -A fontname - 2 > /dev/null  |grep FontName
```

Then you add an entry to the ghostscript `Fontmap` file in the correct format, eg

```
some-font       (/usr/share/fonts/subdirectory/somefont.pbf);
```

Well, that works fine, but try doing it with 500 or so fonts. This is the kind of thing that calls for a short perlscript:

```
#!/usr/bin/perl
# ttfontmap -- generate fontmap file for TrueType fonts
my $directory=shift || print STDERR "Usage: ttfontmap {directory}\n";

$directory=~s/\/$//;

for my $fontname ( glob ( "$directory/*.ttf" ) )
{
    open ( R, "sh -c \"ttf2pt1 -A $fontname - 2>/dev/null\" |" );
    while ( <R> )
    {
        if ( $_ =~ /^FontName/ )
        {
            s/^FontName\s*//;
            chomp;
            print "/" .  $_ . "    ($fontname);\n" ;
        }
    }
    close R;
}
```

You can download this script

To set this script up, all you need to do is cut and paste it into a file called `ttfontmap`, and place the file somewhere in your `PATH` (such as `/usr/bin`). You run this script like this:

```
ttfontmap directory > output_file
```

where `directory` is the directory containing the fonts. You are left with the file `output_file` which you can append to your ghostscript fontmap. Note: some will observe that you could just use

```
ttfontmap directory >> /usr/share/ghostscript/version/Fontmap
```

However, I advise against this (what would happen if you typed ``>'' instead of ``>>'' ?)

5.3 Using Ghostscript To Preview Fonts

Once you've made fonts available to ghostscript, you can preview them. Do this by running the ghostscript interpreter on the file `prfont.ps` in your ghostscript installation, and after you start it, type:

```
/Fontname DoFont
```

at the ghostscript font (where `FontName` is the ghostscript name of the font you wish to preview). There are several other ways you can invoke `gs`. For example, if you want to create a postscript file that you can look at in a nicer postscript viewer such as `gv`, you can use

```
gs -sDEVICE=pswrite -sOutputFile=somefile.ps  prfont.ps
```

Having done this, you can also print your output file.

6.True Type to Type1 Conversion

6.1 Why ?

or perhaps the right question to ask is ``why not ?'' The typical Linux user has experienced a migration from Windows, and probably has an enormous collection of TrueType fonts. Many of these fonts (eg those that ship with MS Word and Corel's products) are of fairly good quality. However, some Linux applications, such as Star Office and LaTeX do not support TrueType fonts, but do support Type1 fonts. *update: it looks like Star Office can handle TrueType fonts, but I'm still trying to work out the details. At best, it involves some fairly gruesome hacks.* This is a pity, because with ghostscript support for TrueType, and TrueType font servers, Linux has the infrastructure it needs to handle TrueType.

6.2 How ?

To convert your TrueType fonts into Type1 fonts, go to http://quadrant.netspace.net.au/ttf2pt1/ and get ttf2pt1. To convert a TrueType to a Type1 font, use the following syntax:

```
ttf2pt1 -b file.ttf name
```

Where `name` is the name of the file corresponding to the new Type1 font (ie it's arbitrary. It's a good idea to make it the same as the ttf file. eg `ttf2pt1 -b foo.ttf foo`.

Well, that worked fine for one font. If we have a lot, we need a smarter way to do it. One can just just use a loop:

```
for X in *.ttf; do ttf2pt1 -b $X ${X%%.ttf}; done
```

Alternatively, you can download the ttfutils package and use `ttf2type1` for the conversions.

```
ttf2type1 *.ttf
```

7.WYSIWYG Publishing and Fonts

7.1 Introduction and Overview

Installing fonts for WYSIWYG publishing on Linux is a relatively complex task. It typically involves three steps:

- Make the font available to the X server
- Make the font available to ghostscript
- Make the font available to the application

The main reason for the complexity is that the *font printing system* (ghostscript) is unrelated to the *screen font system*. In a way, Linux's left hand does not know what it's right hand is doing. This problem is nontrivial to solve, because it is possible that printer fonts and display fonts reside on different machines, so there is no guarantee that all fonts the XClient uses are printable.

The good news is that most WYSIWYG applications use what is a reasonable solution to this problem. The solution involves constructing some kind of mechanism that maps screen fonts to printer fonts (this is the main issue. There are also other issues, such as grouping bold, italic and roman variants into ``families'' of fonts). Unfortunately, there is no standard way to do this. It seems that font management standards which address this issue would greatly simplify the installation of fonts into WYSIWYG publishing systems, because all applications could use a system−wide (as opposed to application−specific) configuration.

7.2 Applixware

There are two ways to install fonts into Applixware. One method involves using FontTastic, which is Applixware's ``private'' font server. The other method involves editing Applixware's fontmap, to use a font already installed on the system. Installing into the font server is more convenient, but fonts installed in this manner may only be printed at 300 dpi.

FontTastic

Using FontTastic is the easy way to do it. To install new fonts like this, simply do the following:

1. Run Applixware as root
2. Click on the *tools* menu.
3. Choose ``Font Installer''
4. Check ``OK'' in the popup dialog
5. Click the ``Catalogs'' menu and choose ``create''
6. Fill in the *catalog name* box. It doesn't matter what you put there. For the rest of this example, we'll assume it's called ``foobar''
7. Select your foobar catalog from the catalog manipulations list.
8. From the ``Services'' menu, select ``install fonts into −> FontTastic font server''
9. Make sure catalog foobar is selected in the catalogs list, then press the ``select files'' button.
10. Use the select files dialog to select the fonts you want to install. Press ``OK'' when you've selected the files. For example, if you want to select arial.ttf in the directory /usr/share/fonts/ttfonts/, you would type */usr/share/fonts/ttfonts* in the ``Current Directory'' dialog, then select arial.ttf from the files dialog box, then click ``OK''. Note that you can select multiple files, but they all must come from the same directory.
11. You can edit your list by checking on the different fonts in the list box and possibly removing or renaming them.
12. When you're ready, click the ``install fonts'' button. Then click ``OK''.
13. Go to the ``services'' menu and choose ``update''. Check ``OK'' on the annoying modal dialog, then choose exit from the services menu. Exit applix.
14. Congrats, you're done ! The new fonts will be available when you restart Applix.

Using System Wide Fonts With Applixware

This method is more involved, but produces better results. I recommend that this method is used for fonts that are really important, and that you use a lot. There are a few steps to this:

Make the fonts available to X

This is explained in the previous section

Make the fonts available to ghostscript

This is explained in in the previous section

Edit the fontmap.dir

This is the final step in making your fonts available to Applix, and also the most time consuming step. The file *fontmap.dir* is in under the *axdata/fontmetrics* of your applix installation. The purpose of this step is basically to tell applix which screen fonts go with which outline fonts. This is in general a very nontrivial problem, because the screen fonts are not always on the same computer that the application is installed.

We describe how to add fonts to fontmap.dir. In this example, we add the font Baskerville Italic.

1. First, we add a line that says FontRecord = Baskerville−Normal−Italic In fact, the name we use in FontRecord is completely arbitrary. However, the font record must be unique to the font. Because of this, it's good practice to use the name that ghostscript uses for the font.
2. Next, we a line that says Family = Baskerville The family name for a font is the name that appears in Applix's font selection menu. Typically, it is *non−unique*, since bold, italic, roman and bold−italic variants of a font will typically go under the same family.
3. If the font is either a bold, or italic variant, or both, we need to add the following lines:

   ```
   Slant = 1
   ```

   if the font is italic, and

   ```
   Weight = 1
   ```

   if the font is bold. If the font is bold *and* italic, we add both lines. In this example, we need only add the line

   ```
   Slant = 1
   ```
4. We add a line that looks like this:

   ```
   ScreenName = "-paradise-baskerville-medium-i-normal--0-0-0-0-p-0-iso8859-1"
   ```

   The screen name is the name that the X−server uses for the font. We can list font names containing the string ``bask'' by typing

   ```
   xlsfonts|grep -i bask
   ```
5. Now we add a line that gives the name of the printer font: PostScriptPrinterName = Baskerville−Normal−Italic
6. Next, we need to specify the location of the font metric file and the outline file

   ```
   MetricsFile = /usr/share/fonts/misc/baskvli.afm
   Type1FontFileName = /usr/share/fonts/misc/baskvli.pfb
   ```

   If you are adding a TrueType file, you can use `ttf2pt1` to generate an `afm` file :

   ```
           ttf2pt1 -A foo.ttf - > foo.afm
   ```

   (or get the ttfutils package and use `ttf2afm`) Then you use something like this:

   ```
   MetricsFile = /usr/share/fonts/misc/foo.afm
   ```

   Do *not* include a `Type1FontFileName` directive −− let ghostscript take care of this.

That's it. Now after adding the whole family of fonts, you should have something like this:

```
FontRecord = Baskerville-Normal
Family = Baskerville
ScreenName = "-paradise-baskerville-medium-r-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Normal
MetricsFile = /usr/share/fonts/misc/baskvl.afm
Type1FontFileName = /usr/share/fonts/misc/baskvl.pfb

FontRecord = Baskerville-Normal-Italic
Family = Baskerville
Slant = 1
```

```
ScreenName = "-paradise-baskerville-medium-i-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Normal-Italic
MetricsFile = /usr/share/fonts/misc/baskvli.afm
Type1FontFileName = /usr/share/fonts/misc/baskvli.pfb

FontRecord = Baskerville-Bold
Family = Baskerville
Weight = 1
ScreenName = "-paradise-baskerville-bold-r-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Bold
MetricsFile = /usr/share/fonts/misc/baskvlb.afm
Type1FontFileName = /usr/share/fonts/misc/baskvlb.pfb

FontRecord = Baskerville-Bold-Italic
Family = Baskerville
Weight = 1
Slant = 1
ScreenName = "-paradise-baskerville-bold-i-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Bold-Italic
MetricsFile = /usr/share/fonts/misc/baskvlbi.afm
Type1FontFileName = /usr/share/fonts/misc/baskvlbi.pfb
```

It is possible to do more with this configuration file. The file itself has a *glossary* which explains the format of the configuration file.

7.3 Star Office

Here, we cover Star Office 5.0. The procedure with Star Office 5.1 is similar, but the utility is called `spadmin`, not `psetup`. It's worth mentioning up front that John McLaughlin's page is an excellent source on this issue, and it inspired most of what follows.

Having tried both Star Office 5.0, and 5.1, I have found that Star Office 5.1 seems to give me less grief when adding new fonts. I was not succesful adding true type fonts to Star Office 5.0, but it proved somewhat easier with Star Office 5.1.

Backup Your Configuration Before you Start !

It's good to make a backup in case you inadvertantly hose your configuration. Modifying fonts will impact several files in the `xp3`. You should definitely backup the file `xp3/psstd.fonts`. I recommend going further and backing up the whole `xp3` directory. You can do this by cd–ing to your Star Office directory, then using

```
tar cvzf xp3.tgz xp3
```

to create a backup. To restore a backup, delete the `xp3` directory and unpack the archive

```
rm -rf xp3
tar xvzf xp3.tgz
```

Adding Type1 Fonts to Star Office

Adding Type1 fonts to Star Office is relatively simple. If you want to use your TrueType fonts with Star Office 5.0, the best thing to do is convert them to Type1 fonts, and then follow the procedure outlined here. If you have Star Office 5.1, you might wish to use the proceedure for installing TrueType fonts instead (though it is somewhat more difficult). Firstly, do the usual thing –– make the font available to both X and ghostscript. Once this is done, the font can be installed into Star Office using the `psetup` tool. The procedure is as follows:

1. As root, run psetup (or spadmin if you have Star Office 5.1)
2. Press the ``add fonts'' button.
3. The easiest thing to do after this is press the ``initialize font paths'' button. This puts a list of all fonts in your X font path in the list box.
4. Choose the directory containing the font you wish to install (it should be in the box), and then press ``OK''.
5. Click the ``convert all font metrics button''.

That's it. You're done. You can exit (or click ``OK'' until it exits). When you restart Star Office, you will have the new fonts.

Adding TrueType Fonts to Star Office

Adding TrueType fonts to Star Office is nontrivial, but possible. After some hard work, and long hours stareing at John McLaughlin's page page, I finally got them working in Star Office 5.1. Note that this does not work with version 5.0. The following steps are appropriate it you are printing through ghostscript:

- Make the fonts available to X.
- Make the fonts available to ghostscript.
- You need to have `afm` files for the fonts you wish to add. Use

  ```
  ttf2pt1 -A foo.ttf - > foo.afm
  ```

  to create the `afm` files. Alternatively, you can get the
- ttfutils package and use `ttf2afm` The advantage of this is you can handle several at a time, eg

  ```
  ttf2afm *.ttf
  ```

- Star Office needs `pfb` files corresponding to each `ttf` file. You can create them with the command

  ```
  touch foo.pfb
  ```

  Actually, Star Office only uses these files for printing purposes. And by enterring the font in the PPD, thus duping Star Office into thinking the fonts are inside your printer (when they're actually inside ghostscript's rendering system), you get around needing to use these files. Star Office just seems to require that the `pfb` file exists to install the font.
- Now you can run `spadmin` and install the font(s).
- Now add the fonts to the PPD file corresponding to your printer configuration. The name you use for the font should be the same name Star Office uses for it, *not* the ghostscript font name. For example, if the font is `foobar.ttf` and the corresponding `afm` file is `foobar.afm`, you use the name ``foobar'' for the font in the PPD file. The entry should look something like this:

  ```
  *Font cloistrk: Standard "(001.002)" Standard ROM
  ```

On the other hand, if you are not printing from ghostscript, you have different issues to deal with. In this case, tricking Star Office into thinking that your printer has the fonts is a bad idea, because your printer does *not* have the fonts in the ROM, so while `gv` will display the postscript files nicely, your printer will not be able to print them. If you have a postscript printer, the main differences are as follows:

- Do not edit the PPD file.
- Instead of using `touch foo.pfb` to create empty `pfb` files, you need the `pfb` files to be Type42 postscript fonts. A Type42 font is really a ``printer TrueType font''. You don't really notice Type42 fonts even when you use them, because most applications handle them transparently. To create Type42 fonts, you use ttfps to create the files.

  ```
  ttfps foo.ttf foo.pfb
  ```

There are some gotchas. Sometimes, Star Office might not choose the screen font you like. It is sometimes worth checking `xp3/psstd.fonts` and possibly editing it to make sure that Star Office is really using the font you had in mind for screen display. Also, Star Office doesn't handle configuration problems gracefully. If there's something wrong with your configuration, it's possible that the word processor will not even start. This is why you should back up your `xp3` directory.

Under the Hood

If you wish to install TrueType fonts in Star Office, you may need to learn how Star Office handles things. When you run `spadmin` or `psetup`, the following happens:

- Star Office makes symbolic links to the `pfb` outline files in your `xp3/pssoftfonts` directory.
- The `afm` file is copied into the directory `xp3/fontmetrics/afm/`
- An entry is added to the `xp3/psstd.fonts` file. This file stores the names of all the screen fonts used by Star Office (in particular, it maps the screen fonts to the outline filenames).

This is why it's good to simply backup the whole `xp3` directory –– it is the only convenient way to restore Star Office to a clean configuration.

7.4 Word Perfect

Nothing yet. Rod Smith's webpage is the definitive resource regarding installing fonts on Word Perfect.

8.Netscape

Perhaps the most notorious application as far as fonts are concerned is the dreaded Netscape. However, there is a fairly simple procedure to attack Netscape font ugliness. The main problem is that Netscape wants to use 75dpi fonts which is typically too small. You can fix this by specifying the appropriate X resources in your `.Xdefaults` file:

```
Netscape*documentFonts.sizeIncrement: 20
```

```
Netscape*documentFonts.xResolution*iso-8859-1: 100
Netscape*documentFonts.yResolution*iso-8859-1: 100
```

The number 100 can be chosen arbitrarily. For example, if you like your fonts really large, like I do, then you may want to use 150 instead.

The other essential tip with regard to addressing Netscape font ugliness is this -- get the Microsoft font pack. These fonts are widely used and it makes an enormous difference if you have (or don't have) those fonts.

9.TeX / LaTeX

9.1 A Quick Primer on LaTeX/TeX fonts

Adding fonts to TeX and LaTeX is a somewhat complex procedure. However, like a lot of things, it's easy if you know how to do it. Some fonts are distributed in metafont format, and some in Type1 format. Usually, the Type1 formats are more easily available. However, metafont fonts have the distinct advantage that they can adjust their shape at different sizes, while Type1 and TrueType fonts at different point sizes are simply magnified or reduced versions of precisely the same shape. The main reason why this feature is desirable is that ideally, fonts should be (relatively) wider at smaller sizes and narrower at larger sizes.

For this discussion, we focus on Type1 fonts, since they are more widely available, and more problematic to install.

Here's a quick primer on LaTeX fonts. LaTeX uses the following types of font files for handling Type1 fonts:

- `.pl` -- property list. This is a human readable version of a tex font metric file.
- `.vpl` -- virtual property list. Human readable version of a virtual font file.
- `.fd` -- font definition. Used to define a *family* of fonts.
- `.tfm` -- tex font metric. This is a metric file, as explained in the glossary. It is completely analogous to the `.afm` files used by Type1 fonts. TeX needs the font metrics to properly layout the page.
- `.vf` -- virtual font. These files contain encoding details, and act as interpreters. TeX treats them as fonts. For example, Imagine that there's some wacky font foobar-exp.pfb which consists of a few (say 20) alternate characters, and there's a virtual font which uses a few of these alternate characters (and it gets the rest of the characters from font foobar.pfb). Dvips might say ``I want character 65 of virtual font foo.vf''. Dvips knows that 65 is always an ``a'' in TeX's scheme. Then the virtual font maps TeX's request to a request for character 14 of the Type1 font foobar.pfb (which might be the alternate ``a'' in the Type1 font foobar.pfb ). The virtual font mechanism is very flexible and allows fonts to be constructed from many different font files. This is useful when using fonts such as adobe's ``expert'' fonts.
- `.pk` -- a device dependent bitmap font. These are usually constructed on an as-needed basis (they are renderings of Type1 and metafont fonts). They are typically high resolution (about 300-1200dpi), and are intended to be rendered on a printer. Because of their high resolution, and the fact that each point size of each font requires a `.pk` file, they require a lot of disk space, so they are cached, but not stored.
- `.mf` -- metafont files. Metafont is a graphics programming language widely used for font design (though it can also be used for graphics). It has many advantages over TrueType and Type1 schemes.

> However, it's main weakness is that it is not as ubiquitous as TrueType or Type1 (and it is also not terribly well suited to WYSIWYG publishing. Of course, this isn't a major disadvantage when TeX is your typesetting system.)

It's good to know your way around the TeX directory structure. Here are the main directories you'll need to know about:

- `$TEXMF/fonts` −− the main font directory
- `$TEXMF/fonts/type1` −− the type1 font directory
- `$TEXMF/fonts/type1/foundry` −− the directory for the shape files in a given foundry
- `$TEXMF/fonts/type1/foundry/fontname` −− contains the font called *name*. The *name* is usually plain English, and needn't follow TeX's cryptic naming scheme for fonts.
- `$TEXMF/fonts/afm/foundry/fontname` −− the directory containing the `afm` files corresponding to the font `name` belonging to foundry `foundry`.
- `$TEXMF/fonts/tfm/foundry/fontname` −− analogous to the `afm` directory, but contains `tfm` files instead.
- `$TEXMF/fonts/vf/foundry/fontname` −− similar to the above, but contains the virtual fonts.
- `$TEXMF/fonts/source/foundry/fontname` −− similar to the above, but contains metafont files.
- `$TEXMF/dvips/config/psfonts.map` −− fontmap file for dvips. This file is similar in both function and format to ghostscript's Fontmap file.
- `$TEXMF/tex/latex/psnfss` −− this is where all the font definition files go.

9.2 Adding Type1 fonts

Naming the fonts

First, you need to appropriately name your fonts. See the fontinst documentation on your system for instructions on how to name fonts (it should be fontinst subdirectory of the directory containing your tetex documentation). To make a long story very short, the naming scheme is `FNW{V}E{n}` where:

- F is a one−letter abbreviation for the foundry (m = monotype, p = adobe, b = bitstream, f = free)
- N is a two letter abbreviation for the font name (for example, ag = ``avant garde'')
- W is the font weight (r = regular, b = bold, l = light d = demibold)
- V is an optional slope variant (i = italic , o = oblique)
- E is an abbreviation for the encoding (almost always `8a` which is adobe standard encoding).
- N is an optional width variant (n = narrow)

For example, the font Adobe Garamond demibold is `pgad8a`.

Creating the virtual fonts and tex font metrics

Now you can run `fontinst` as follows:

```
latex `kpsewhich fontinst.sty`
```

then you type at the prompt:

```
\latinfamily{font_name}{}\bye
```

where `font_name` is the first three letters of your font file name (for example, `pad` for adobe garamond). Now `fontinst` will generate a number of files –– font description files, property list files and virtual property list files. It also generates a lot of `.mtx` files. These are created by `fontinst`, but you don't need to use them. You need to convert the property lists and virtual property lists to metrics and virtual fonts. This is done using the utilities `vptovf` and `pltotf`.

```
for X in *.pl; do pltotf $X; done
for X in *.vpl; do vptovf $X; done
```

Then remove the old `vpl`, `pl` and `mtx` files.

Configure dvips

You will need to edit your dvips config file, `psfonts.map`. The best way to explain the format of the file is to give an example.

```
marr8r          ArialMT <8r.enc <farr8a.pfa
marbi8r         Arial_BoldItalicMT <8r.enc <farbi8a.pfa
marb8r          Arial_BoldMT <8r.enc <farb8a.pfa
marri8r         Arial_ItalicMT <8r.enc <farri8a.pfa
marr8rn         Arial_Narrow <8r.enc <farr8an.pfa
```

The `8r.enc` is simply there to inform dvips of the encoding scheme used (in all our examples, it's 8r, because of the way `fontinst` constructs the virtual fonts). The leftmost column is the font name TeX uses. The second column is the real name of the font, which is hardcoded into the font file (this name can be deduced by opening the `afm` file in a text editor, and looking for the `FontName` directive). The last column is the filename of the shape file corresponding with the font. It is not necessary to provide a directory path –– tex knows where to look.

Test the font

Try running latex on a document like this:

```
\documentclass{article}
\begin{document}
        \usefont{T1}{pga}{m}{n}\selectfont
        \huge
        Testing a new font \dots the quick red fox jumped over the lazy brown dogs
\end{document}
```

where you replace `pga` with the outline of your font. If this works, you are almost done. All you have to do now is put all the files in the right directories (as explained in the primer), then run

```
texconfig rehash
```

so that tex can update the directory lists.

Create a .sty file

You may want to create a .sty file so that you can more easily use fonts. Use the files in `$TEXMF/tex/latex/psnfss` as a template.

10.Getting Fonts For Linux

10.1 True Type

Commercial Software

True type fonts are very easy to come by, and large amounts of them are typically included in packages like Microsoft Word and Word Perfect. Getting Word Perfect is an easy way to get an enormous amount of fonts (and if you're really cheap, you could buy a legacy version of Word Perfect for windows. The fonts on the CD are readable.)

Microsoft's Font Download

Microsoft have also made several TrueType fonts available. The `.exe` file is simply an archive, you can extract it using `unzip`. You can get them from the download site

Luc's Webpage

Luc Devroye's webpage has links to several sites with free fonts available. What's unique about these fonts is that a lot of them are really free, they are not ``warez fonts''.

Web sites with truetype fonts

There are several web sites offering freely available downloadable fonts. For example, the freeware connection has links to a number of archives.

Foundries

Several foundries sell TrueType fonts. However, most of them are quite expensive, and for the same money, you'd be better of with Type1 fonts. I'll discuss these more in the Type1 fonts section. The one place that does do sell true type fonts at low prices is buyfonts. Please read the section on ethics before you buy cheap fonts.

10.2 Type 1 Fonts and Metafont

Dealing With Mac and Windows Formats

Many foundries ship fonts with Windows and Mac users in mind. This can sometimes pose a problem. Typically, the ``Windows fonts'' are fairly easy to handle, because they are packed in a `zip` file. The only work to be done is converting the `pfm` file to and `afm` file (using `pfm2afm`).

Macintosh fonts are more problematic, because they are typically made available in `.sit.bin` format −− stuffit archives. Unfortunately, there is no tool for Linux that can unpack stuffit archives created with the newer version of stuffit. The only way to do it is run Executor (Mac emulator), or try running stuffit in dosemu or Wine. Once the `sit.bin` file is unpacked, the Macintosh files can be converted using `t1unmac` which comes with the `t1utils` package.

Unfortunately, some vendors only ship Type1 fonts in Macintosh format (stuffit archives). However, according to font expert Luc Devroye, all major foundries make Type1 fonts available for Mac and Windows.

Free Stuff

ctan have a number of good fonts, many of which are free. Most of these are in Metafont format, though some are also Type1 fonts. Also, see Bluesky who have made available Type1 versions of the computer modern fonts. (The computer modern fonts are of excellent quality −− to purchase anything of comparable quality and completeness will cost you around $500−. They are comparable to the premium fonts.)

Luc Devroye's webpage has links to several sites with free fonts available. What's unique about these fonts is that a lot of them are really free, they are not ``warez fonts''.

URW have released the standard postscript fonts resident in most printers to the public domain. These fonts are quite good.

The Walnut Creek Archive has several freely available fonts, and shareware fonts. Some of these are obvious ripoffs (and not very good ones). If a font doesn't come with some kind of license, chances are it's a ripoff. Also Winsite have several Type1 fonts (in the fonts/atm subsection of their windows 3.x software). Unfortunately, several of these have afm files which have mistakes and are missing all kerning pairs (you can fix the afms by editing the "FontName" section of the afm files. It should match the fontname given in the font shape file. Of course, adding kerning pairs is a topic beyond the scope of this document.)

Luc Devroye's webpage includes several free fonts he designed, as well as a lot of links, and fascinating discussion on the topic of typography. This site is a ``must−visit''. There are also several links to many foundries.

Commercial Fonts

Value vs Premium: Why Should I buy Premium Fonts ?

So you're wondering –– why do some fonts cost a lot and others are cheap ? These fonts are the ``standard postscript fonts'' resident in most postscript printers. Also the famous Why should I buy the more expensive ones ? My take on it is that for a casual user, the value fonts ( such as those on the Bitstream CD ) are just fine. However, if you're using the fonts for ``real work'', or you're just a hard core font junkie, then the better quality fonts are a must–have –– and most of the quality fonts are either free (for example, Computer Modern), or they are upmarket commercial fonts.

The advantage of the cheaper fonts is self evident –– they are cheaper. The quality fonts also have their advantages though.

- *Ethical issues:* The cheaper fonts are almost always ripoffs. Type design takes a long time and and experienced designer. Fonts that are sold for less than $1– per font were almost certainly not designed by the vendor. CDs with insane quantities of fonts on the are almost always ripoffs (the possible exceptions being collections from major foundries that cost thousands of dollars). Usually, the ripoffs lack the quality of fonts from respectable founries.
- *Completeness:* The higher quality fonts (notably from Adobe) come in several variants, with some nice supplements to provide the user with a more complete font family. There are often bold, italic, and demibold variants, swash capitals, small caps, old style figures, and extra ligatures to supplement the font. More recently, Adobe have a multiple master technology which gives the user (almost) infinite variation within one font family.
- *Quality:* A lot of the freely available fonts or the cheap ripoffs lack fairly essential features such as kerning pairs and decent ligatures. They are basically cheap copies. In contrast, reputable designers take a lot of trouble to study the original design, and rework it to the best of their ability.
- *Authenticity:* The person who designed Adobe Garamond (Robert Slimbach) actually studied the original designs of Claude Garamond. In fact reputable foundries always carefully research their designs, rather than just swiping something off the net, and modifying it with Fontographer.

Value

- An excellent place to go for a CD packed with several Type1 fonts of reasonable quality is Bitstream. Bitstreams more noted products include their 250 font CD and their 500 font CD (the latter goes for $50– at the time of writing). These are fairly good quality fonts, and are a fairly good starting point for the casual user. The fonts used in Corel's products are (mostly) licensed from bitstream.
- Matchfonts offer more modestly priced fonts –– they are distributed in ``packs'' of about 8 fonts for $30. This includes some nice calligraphic fonts. All fonts seem to be offered in a usable format (the windows ATM fonts come in a .exe file. Don't let the extension fool you –– it's just a zip archive). These are not ripoffs as far as I can tell.
- EFF sell TrueType fonts for $2– per hit. They also have ``professional range'' postscript and TrueType fonts for $16– per typeface.

Premium

- Adobe have several high quality, fonts available at Adobe's type website. Some of these are expensive, but they have several more affordable bundles –– see Adobe Type Collections. Adobe have some of the most complete font families on the market, for example, Garamond, Caslon, and their multiple masters (Myriad and Minion, used on their website are among the nicer of their multiple masters.)

- Berthold Types Limited is a major foundry, who offer several quality fonts. Some of them are resold through Adobe, all are directly available from Berthold. Same price ballpark as Adobe.
- ITC develop several quality fonts (including some of the ones Corel ships with their products) at http://www.itcfonts.com They offer family packages for about $100–180 US. Their fonts, come in both Type1 and TrueType format. It's better to choose the ``Windows'' package, because Mac formats are difficult to handle on Linux.
- Linotype are a well known foundry who offer fonts by legendary designers including Herman Zapf. (yep, the guy ``Zapf Chancery'' is named after. He also designed Palatino.)
- Monotype develop most of the fonts shipped with Microsoft products. One of the older and well respected foundries.
- Tiro Typeworks sell good quality, if somewhat expensive typefaces. Their typefaces are very complete, for example, they include complete sets of ligatures, and smallcaps, titling fonts, etc. UNIX is listed as one of the OS options –– which is a welcome surprise after seeing the words ``Windows or Mac'' too many times..

More Links

For links to a bunch of other foundries, see Luc Devroye's page

---

11.Useful Font Software for Linux

There are several font packages for Linux. Many of them are essential.

- chkfontpath is a utility for manipulating the `xfs` configuration file.
- DTM –– the Definitive Type Manager is a global font management tool. This is a developer's release.
- fontinst is a LaTeX package designed to simplify the installation of Type1 fonts into LaTeX.
- Freetype is a TrueType library that comes with most Linux distributions
- Ghostscript is the software that is used for printing on Linux. The version of ghostscript that ships with Linux is GNU ghostscript. This is one version behind the latest release of Aladdin ghostscript (who release their old versions under the GPL)
- pfm2afm is a utility for converting windows `pfm` font metric files into `afm` metrics that can be used for Linux. This is based on the original version available at CTAN, and includes modifications from Rod Smith to make it compile under Linux.
- mminstance and t1utils are two packages for handling Type1 fonts. mminstance is for handling Adobe's multiple master Type1 fonts. t1utils is a suite of utilities for converting between the different Type1 formats.
- ttf2pt1 is a TrueType to Type1 font converter. It is useful if you have applications that require Type1 fonts.
- ttfps converts `.ttf` TrueType font files into Type42 files.
- ttfutils A package of utilities for handling TrueType fonts. This package requires `ttf2pt1`. Useful if not essential.
- type1inst is an essential package for installing Type1 fonts. It greatly simplifies the installation.
- xfstt is a TrueType font server for Linux. It's useful, but `xfs` is probably a better choice.
- xfsft The xfsft font server. Note that this is included in `xfs`.
- x–tt is a font server designed to handle Korean and Japanese fonts.

12.Ethics and Licensing Issues Related to Type

Font licensing is a very contentious issue. While it is true that there is a wealth of *freely available* fonts, the chances are that the fonts are ``ripoffs'' in some sense, unless they come with a license indicating otherwise. The issue is made more confusing by intellectual property laws regarding typefaces. Basically, in the USA, font *files* are protected by copyright, but *font renderings* are not. In other words, it's illegal to redistribute fonts, but it's perfectly legal to ``reverse−engineer'' them by printing them out on graph paper and designing the curves to match the printout. Reverse engineered fonts are typically cheap and freely available, but of poor quality. These fonts, as well as pirated fonts are often distributed on very cheap CDs containing huge amounts of fonts. So it's not always easy to tell if a font is reverse engineered, or simply pirated. This situation creates an enormous headache for anyone hoping to package free fonts for Linux.

Perhaps one of the most offensive things about the nature of font piracy is that it artificially debases the value of the work that type designers do. Pirated fonts invariably are bundled en masse onto these one zillion font CDs, with no due credit given to the original designers. In contrast, what is commendable about several legitimate font foundries is that they credit their designers.

There are many differing opinions on this issue. See typeright for an explanation of the case in favour of intellectual property rights. Also, see Southern Software, Inc for another opinion −− but don't buy any of their fonts! Their Type1 fonts (poorly reverse−engineered Adobe fonts) do not have AFMs, and are thus unusable.

The comp.fonts FAQ also discusses the issues of fonts and intellectual property, as does Luc Devroye's homepage. These references are somewhat less extreme in their views.

13.References

13.1 Font Information

- Rod Smith's homepage contains a wealth of information about using fonts and printers with Applixware and Word Perfect.
- John McLaughlin's page discusses setting up fonts with Star Office
- Jim Land's homepage contains a lot of links to sites on postscript and fonts.
- The comp.fonts FAQ is the definitive font FAQ.
- Luc Devroye's homepage Contains enough information about fonts and other things to sink a ship. This guy designed a bunch of free fonts, and his homepage has a lot of interesting links, information and commentary.
- The Font Deuglification HOWTO discusses TrueType fonts under Linux. This is the clear winner of the ``TrueType'' HOWTOs. An excellent source of information.

- TrueType Fonts in Debian mini−HOWTO discusses installing TrueType in Debian. A must−read for Debian users. Also worth reading if you have *any* distribution that doesn't have the version of `xfs` with TrueType support.
- The (preliminary) True Type HOWTO −− an incomplete HOWTO dated June 1998. Included in this list for completeness.
- TrueType for XFree86 Mini−HOWTO −− a slightly dated HOWTO. Only applicable to Redhat 5.x

13.2 Postscript and Printing Information

- Adobe's Postscript page is the definitive site on the postscript standard.
- Ghostscript's home page has a lot of information, and all the latest printer drivers.
- Jim Land's homepage contains a lot of links to sites on postscript and fonts.
- Christopher Browne's Printing FAQ

14.Glossary

- **afm** Stands for **Adobe Font Metric**. These files store information about the width and spacing associated with the font, as opposed to information about the font shape.
- **anti−aliasing** also referred to as font smoothing is a technique used to render fonts on low resolution devices (such as a monitor). The problem with rendering fonts is that the fonts consist of outlines, but the device renders in dots. The obvious way to render a font is to color black any pixel inside the outline, and leave all other dots. The problem with this is that it doesn't adequately address the pixels that are on the outline. A smarter algorithm would be to color the boundary pixels gray. Anti−aliasing essentially involves doing this.
- **bdf** fonts are a variety of bit−mapped fonts that may be used with X.
- **bitmap fonts** These fonts are simply a collection of dots. Each character of the font is stored as a dot matrix. Because of this, bitmap fonts are device dependent, so you can't use the same bitmap fonts on a screen and a printer. Examples of bitmap screen fonts include .pcf and .bdf fonts used by X. Examples of printer bitmap fonts include TeX's PK fonts.
- **didone** see modern.
- **DPI** Dots Per Inch. Monitors typically display at 75−100 DPI, while modern printers vary from 300−1200 DPI
- **expert fonts** are collections of additional characters that supplement a font. They include small caps fonts, ornaments, extra ligatures, and variable width digits. Many of Adobe's fonts have expert fonts available.
- **font server** a background program that makes fonts available to XFree86.
- **glyph** A glyph is a fancy word for a shape. It is a component that makes up an outline font. For example, the dot on the letter ``i'' is a glyph, as is the vertical line, as are the serifs. Glyphs determine the shape of the font.
- **kerning** In variable width fonts, different pairs of characters are spaced differently. The font metric files store information regarding spacing between pairs of characters, called **kerning pairs**.
- **ligature** A ligature is a special character that is used to represent a sequence of characters. This is best explained by example −− when the letter fi are rendered, the dot on the ``i'' collides with the ``f'', and the serif on the top left of the i can also collide with the horizontal stroke of the f. The fi ligature is a single character that can be used in the place of a single f followed by a single i. There are also

ligatures for fl, ffi, and ffl. Most fonts only include the fi and fl ligatures. The other ligatures may be made available in an **expert font**.

- **metafont** A graphics language used for creating fonts. Metafont has a lot of nice features, the main one being that fonts created with metafont need not just scale linearly. That is, a 17 point computer modern font generated by metafont is not the same as a magnified 10 point computer modern font. Prior to Adobe's multiple master technology, metafont was unique with respect to having this feature. Metafonts main advantage is that it produces high quality fonts. The disadvantage is that generating bitmaps from the outline fonts is slow, so they aren't feasible for WYSIWYG publishing.
- **metric** this stores information about how much space a font takes up. A font metric is like a box that one can embed the font in. Font metrics are essential for the purpose of laying out fonts on a page, while the font shape itself is not. So typically, variable width fonts have metric information as well as shape information. The metric also includes kerning information.
- **modern** fonts are fonts based on designs developed in the 19th century or later. The moderns have a solid appearance due to their vertical stress. They tend to have more ``character'' or ``attitude'' than the old styles and transitionals, but still carry a certain amount of dignity and formality. They are not suited for writing long passages, but they are useful for adding character to a piece of writing. Bodoni is a notable modern typeface.
- **old style** fonts are a traditional class of typeface. The old style fonts are based on designs from as far back as the late 15th century. Old style fonts are great for writing long documents (such as books). While the old style fonts are designed in the tradition of the earlier designers, some of them were designed quite recently. Notably, the face **Goudy Old Style** was designed by Goudy in the early 20th century. Notable old style faces include Goudy Old Style, Garamond, and Caslon.
- **pcf** fonts are bit−mapped fonts used by X.
- **postscript** is a programming language designed for page description. Postscript was a trademark of it's inventor, adobe. However, it is also an ISO standard. Postscript needs an interpreter to render it. This can be done via a program on the computer, such as ghostscript, or it can be interpreted by some printers.
- **serif** fonts are fonts with little hooks (called serifs) on the ends of the font. the serifs usually help make the font more readable. However, serifs are quite difficult to render on low resolution devices, especially at small font sizes (because they are a fine detail), so it is often true that at small sizes on low resolution devices, sans serif fonts (such as Microsoft's Verdana) prove more readable. Another issue is that there are sans serif fonts (like the moderns) that are not designed for writing long documents.
- **sans serif** fonts are fonts without serifs (sans is French for ``without''). These fonts have a stark appearance, and are well suited for writing headlines. While textbook typography mandates that serif fonts be used just for headlines, they can have other uses. There are sans serif fonts designed for readability as opposed to impact. Short punchy documents that are skimmed (such as catalogues and marketting brochures) may use them, and recently, Microsoft have made available the Verdana font which is designed for readability at small sizes on low resolution devices. Well known sans serif fonts include Lucida Sans, MS Comic Sans, Avant Garde, Arial, Verdana, Century Gothic.
- **slab serif** fonts are a certain class of font whose serifs look like slabs (eg flat lines or blocks) and not hooks. Slab serif fonts are **often**, but not always very readable. Because the serifs are simple and strong, they give one the feeling that they have been punched into the page. Well known examples of slab serifs are Clarendon, New Century Schoolbook, and Memphis.
- **transitional** fonts are fonts that are based on more recent designs than the old style fonts. many of the transitional fonts have good readability. Notable transitionals include Baskerville, and Times Roman.
- **Type1** is a type of font designed by Adobe. These fonts are well supported by almost all linux applications, because they have been supported by the X server architecture and the postscript standard for a long time. Postscript fonts are distributed in many different formats. Typically, a UNIX postscript font is distributed as an `afm` (adobe font metric) file, and an outline file, which is

usually a `.pfb` (printer font binary) or `.pfa` (printer font ascii) file. The outline file contains all the glyphs, while the metric file contains the metrics.

- **type3** fonts are similar to Type1. The file extensions are similar to Type1 fonts (they are distributed as `pfa` and `afm` files), but they are not supported by X, and because of this, there are not very many linux applications which support them.