CHAPTER 10

Font Specification

Some OpenWindows client programs allow you to choose the fonts you want by selecting from a menu. Any user-ready commercial application, such as the DeskSet clients described in Chapter 7, *The OpenWindows DeskSet Clients*, and commercial publishing software packages such as *FrameMaker* and *Interleaf*, or graphics packages such as Adobe *Illustrator* or Corel *Draw*, provide a simple interface for selecting fonts. However, the *cmdtool* terminal emulators does not have such a menu, and the *xterm* terminal emulator only lets you choose from a few pre-specified fonts. You can choose the font used for the text in *olwm* menus or in *cmdtool* or *xterm* windows, but you must do it via the command line or from X resources..

The X Window System has a fairly complex font naming system which, like most things about X, is designed for maximum flexibility rather than for simplicity or ease of use. This wouldn't be so bad if a typical font name weren't mind-bending at first glance. To create a *cmdtool* window whose text is to be displayed in 14-point Courier bold, you'd hope to be able to type something like this:

```
% cmdtool -fn Courier-Bold-14
```

Fortunately you can do that on OpenWindows because of Sun's clever alias naming and "font scaling." Accordingly, users of OpenWindows may wish to skip this chapter at first blush, and return to it later when they need to select fonts via command line options or resource specifications, for MIT or other non-commercial applications.

On a normal X11 server, you might need to type, as a worst case, something like this:

```
% cmdtool -fn -adobe-courier-bold-r-normal--14-140-75-75-m-
        90-iso8859-1
```

Fortunately, you can use asterisks as wildcards to simplify this name to a somewhat more reasonable one:

```
% cmdtool -fn '*courier-bold-r*140*'
```

and you can define even simpler aliases, so that you could end up typing a command line like the simplified one shown earlier:

```
% cmdtool -fn Courier-Bold-14
```

In this chapter, we're going to try to make sense out of the sometimes bewildering jungle of information about fonts under X. First, we'll explain the font naming convention in detail. Along the way, we'll acquaint you with the appearance of some of the basic font families (groups of related fonts), and the various permutations (such as weight, slant, and point size) within each family.

Then, we'll talk about how to use font name wildcards to simplify font specification. We'll also talk about the font search path (the directories where the font files are stored), and how to define aliases for font names.

Finally, we'll talk about some of the utilities provided for dealing with fonts:

- *xlsfonts*, which lists the names of the fonts available on your server, as well as any aliases.
- *xfd* (font displayer), which allows you to display the character set for any individual font you specify on the command line.
- *xfontsel* (font selector), which allows you to preview fonts and select the name of the one you want (this name can then be pasted onto a command line, into a resource file, etc.)
- *text* (OpenWindows only), which allows you to display a representative text sample (character display) for any font selected from a menu of all available fonts.

10.1 Font Naming Conventions

The X11 "logical font naming convention" has been around since X11 Release Three, so it is included in all versions and variants of the X Window Systems in use today. As we'll see in a moment, these logical font names allow for complete specification of all of the characteristics of each font. Unfortunately, this completeness makes them somewhat difficult to work with, at least until you learn what all the parts of the names mean, and get a handle on which parts you need to remember and which you can safely ignore. (By the end of this chapter, you should have that knowledge.)

The *xlsfonts* client can be used to display the names of all the fonts available on your server. When you run *xlsfonts*, you'll get an intimidating list of names similar to the name in Ref f. Upon close examination, this rather verbose name contains a great deal of useful information: the font's developer, or foundry (Adobe); the font family (Courier); weight (bold); slant (oblique); set width (normal); size of the font in pixels (10); size of the font in tenths of a point (100 tenths of a point, thus 10 points); horizontal resolution (75 dpi); vertical resolution (75 dpi); spacing (m, for monospace); average width (60—measured in tenths of a pixel, thus 6 pixels); and character set (iso8859-1).

As mentioned earlier, font name wildcarding can eliminate lots of unnecessary detail. If you are already familiar with font characteristics, skip ahead to the section "Font Name

Wildcarding," later in this chapter, for some tips and tricks. If you need a refresher on fonts, read on as we illustrate and explain each of the elements that make up the font name.

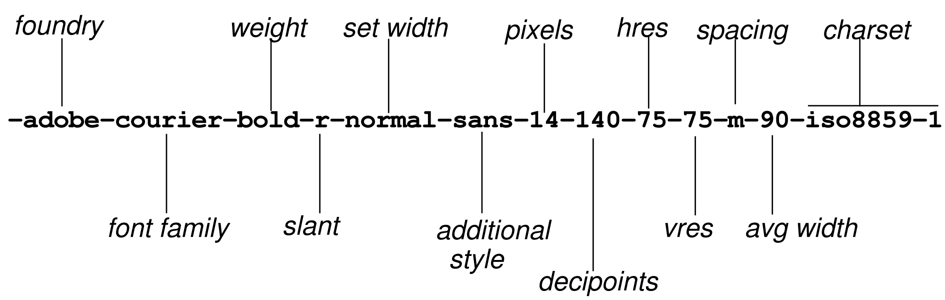

Figure 10-1. Font name, X11 Logical Font Naming Convention

10.1.1 Font Families

It has been a decade or so since the advent of desktop publishing and, by now, it is unlikely that anyone in the computer industry is unaware that text can be displayed on the screen and printed on the page using different fonts.

However, the term *font* is used somewhat ambiguously. Does it refer to a family of typefaces (such as Times Roman or Helvetica), which comes in different sizes, weights, and orientations? Or should each distinct set of character glyphs be considered a separate font?

OpenWindows takes the former approach, while historically X11 has taken the latter. When the documentation says that X11 Release 4 has more than 400 fonts , this sounds either intimidating or impressive, depending on your mood. But, in fact, the X11R4 distribution includes only eight font families (Courier, Helvetica, New Century Schoolbook®, Symbol, Times, Lucida® and the Clean family of fixed-width fonts), plus several miscellaneous and special purpose fonts. By contrast, both the Macintosh and OpenWindows support dozens of font families. Although OpenWindows comes with "57 fonts" (see Figure 10-2), there are actually about two dozen families when you remove variations such as Bold and Italic (AvantGarde (two versions), Bembo, Bookman, Courier, Gill, Gill Sans, Helvetica, Lucida, Lucida Bright, Lucida Sans, Lucida Sans Typewriter, LucidaBright, LucidaTypewriter, New Century Schoolbook, Palatino, Rockwell, symbol, Times, Zapfchancery, and Zapfdingbats) plus some special-purpose fonts (the X11 special fonts plus XX OPEN LOOK Cursor and Glyph fonts). Hundreds of fonts are available both commercially and as "free software" for PostScript laser printers (many of these can be used in OpenWindows as well), and commercial typesetters support thousands of families.OpenWindows uses fonts in Sun's "F3" Folio format. PostScript Type Three fonts can also be used and, in revision3.1 of OpenWindows, you can use PostScript Type One as well. Both

F3 and PostScript Type One are widely available commercially. X11R5 also supports Bit-Stream's "Speedo" format fonts,, which some commercial type suppliers provide.:

Figure 10-2. xshowfonts listing for the 57 OpenWindows Fonts.

Because of the difficulty of working with fonts in X11R4, the X Consortium has included scalable fonts in X11R5, half a decade after NeWS (now part of OpenWindows) introduced them to the UNIX workstation market. X11R5 incorporates scalable font technology from BitStream, using their “Speedo” format fonts. X11R5 also includes “font server” programs that make fonts available to the X11 server. This will be especially useful to sites that run a variety of diskless X Terminals, since at present each X Terminal vendor uses a slightly different method of downloading fonts. The Font Server is described in Volume Eight, *X Window System Administrator’s Guide*. However, at the present time (late 1992) few vendors are yet shipping the X11R5 release, so we still need to understand the X11R4 mechanism. By mid-1993, most reputable vendors should be supporting the X11R5 “font server” technology.

When you think of the X11 fonts as consisting of several large font families, rather than hundreds of unique fonts, you can quickly reduce the clutter. Figure 10-2 shows the major families of commercial fonts that are available under X. To illustrate the fonts, we’ve used the simple expedient of printing each font name in the font itself. Font names are truncated to fit on the page. (For those of you who don’t read the Greek alphabet, several entries have names like "-adobe-symbol-medium-r-normal--18..." This font is used for mathematical equations and so forth, rather than for normal display purposes.)†

Figure 10-3. The major font families in the OpenWindows product

You'll notice that with the exception of Courier and Lucidatypewriter, all of the fonts in the figure are *proportionally spaced*. That is, each character has a separate width. This makes them look good on a printed page and for screen-aware text processing programs, but makes them less appropriate for screen display in terminal windows (especially for program editing), since text will not line up properly unless all characters are the same width.

† To generate the figures in this section and in Appendix B, *OpenWindows and X11 Fonts*, we wrote a short program called *xshowfonts*, which displays a series of fonts in a scrollable window. Source code for *xshowfonts* is listed in Appendix B, *OpenWindows and X11 Fonts*. In each case, we used wildcards (discussed later in this chapter) to select the fonts we wanted and then did screendumps of the resulting images. Note that the fonts look better on the screen than they do in the illustration, since the scaling factor used to make the screen dumps exacerbates the "jagged edges" endemic to bitmap fonts.

You will most likely use these proportional fonts for labels or menu items, rather than for running text. (Word processing or publishing programs will, of course, use them to represent proportional type destined for the printed page.)

Courier and Lucidatypewriter are *monospaced* fonts: every character has the same width. Monospaced fonts can be used effectively for the text font in *xterm* or *cmdtool* windows.

There are also some special monospaced fonts originally designed for computer displays. You can think of these as *character cell fonts*. They, too, are monospaced but the spacing relates to the size of a cell that contains each character, rather than (necessarily) to the character itself. Some of these fonts are quite old -they were originally available in X11 Release 2- and they have simple names expressing their size in pixels. For example, in the font named 8x13, each character occupies a box 8 pixels wide by 13 pixels high.They since have been renamed to use the logical font naming conventions with a foundry name of "misc," and a font family of "fixed." but the simple names have been preserved as aliases. shows the character cell fonts, using their short names

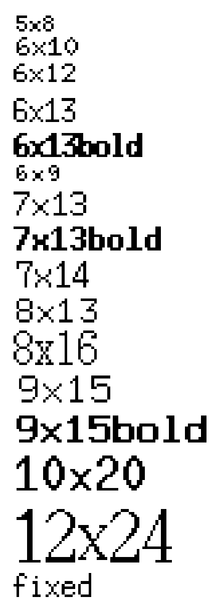

Figure 10-4. Miscellaneous fonts for xterm text

Table 10-1 shows the correspondence between these aliases and full font names. Note that the 6x13 font also has an alias called "fixed" defined for it. The "fixed" alias is used as the default font for *xterm* windows. Twelve-point Helvetica bold roman has the alias "variable," which several applications use as the default font for labels. The Motif window manager *mwm* uses this font for the application name that appears in the titlebar. The OPEN LOOK Window Manager *olwm* uses "Lucida" for its titlebars and menus.

Table 10-1. Fixed Font Aliases and Font Names

Fixed Name	Font Name
fixed	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
5x8	-misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-
6x9	-misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
6x10	-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
6x12	-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
6x13	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
6x13bold	-misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-
7x13	-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
7x13bold	-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1
7x14	-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1
8x13	-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-
8x13bold	-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1
8x16	-sony-fixed-medium-r-normal--16-120-100-100-c-80-iso8859-
9x15	-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso8859-
9x15bold	-misc-fixed-bold-r-normal--15-140-75-75-c-90-iso8859-1
10x20	-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-
12x24	-sony-fixed-medium-r-normal--24-170-100-100-c-120-iso8859-1

X11R4 also includes the Clean family of fixed-width fonts from Schumacher, and DEC's terminal fonts, both of which are illustrated in Appendix B, *OpenWindows and X11 Fonts*.

There are also many other special purpose fonts, such as the Greek Symbol font that we already saw, the cursor font, the OPEN LOOK cursor and glyph fonts, and Kana and Kanji Japanese fonts. See Appendix B, *OpenWindows and X11 Fonts*, for comprehensive lists and samples of these fonts, as well as pictures of the complete character set in some representative fonts.

10.1.2 Scalable Fonts

The X Logical Font Description described above can be used for X11R5 Speedo fonts and for OpenWindows scaleable fonts. Bitmap fonts appear just as they did above, while scalable fonts are listed as having a zero value for pixel size, point size, resolution, and maximum width. For example, while an X11 bitmap font like Helvetica has to be listed once for each of its limited number of point sizes, *xlsfonts* under OpenWindows or X11R5 reports each Lucida font only once. Here are the Lucida listings:

```
-b&h-lucida-bold-i-normal-sans-0-0-0-0-p-0-iso8859-1
-b&h-lucida-bold-i-sans--0-0-0-0-p-0-iso8859-1
-b&h-lucida-bold-r-normal-sans-0-0-0-0-p-0-iso8859-1
-b&h-lucida-bold-r-sans--0-0-0-0-p-0-iso8859-1
-b&h-lucida-medium-i-normal-sans-0-0-0-0-p-0-iso8859-1
-b&h-lucida-medium-i-sans--0-0-0-0-p-0-iso8859-1
-b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1
-b&h-lucida-medium-r-sans--0-0-0-0-p-0-iso8859-1
-b&h-lucidabright-demibold-i-normal--0-0-0-0-p-0-iso8859-1
-b&h-lucidabright-demibold-r-normal--0-0-0-0-p-0-iso8859-1
-b&h-lucidabright-medium-i-normal--0-0-0-0-p-0-iso8859-1
-b&h-lucidabright-medium-r-normal--0-0-0-0-p-0-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-0-0-0-0-m-0-iso8859-1
-b&h-lucidatypewriter-bold-r-sans--0-0-0-0-m-0-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-0-0-0-0-m-0-iso8859-1
-b&h-lucidatypewriter-medium-r-sans--0-0-0-0-m-0-iso8859-1
```

These sixteen listings provide fonts equivalent to about 96 of the bitmap fonts, and have the further advantage that they can be requested in *any* point size. For example:

```
xfd -fn '-b&h-lucida-bold-r-normal-sans--370-0-0-m-0-iso8859-1'
```

will scale the font Lucida Typewriter Bold to thirty-seven points (remember that the point sizes is in tenths, so 370 means 37 points) and display it. However, when using wildcards with this scaling, you can't be as cavalier about leaving out fields as you can with the MIT server. If in doubt, put in all the dashes, even if you're wildcarding most of the fields.

As a convenience, primarily for interactive use rather than in X Resource files, you can use the PostScript names for these fonts, followed by a dash, and the pointsize. For example,

```
xfd -fn Lucidasans-Typewriterbold-37
```

10.1.3 Stroke Weight and Slant

The characters in a given font family can be given a radically different appearance by changing the *stroke weight* or the *slant*, or both.

The most common weights are medium and bold. The most common slants are roman (upright), italic, or oblique. (Both italic and oblique are slanted; however, italic versions of a font generally have had the character shape changed to make a more pleasing effect when slanted, while oblique fonts are simply a slanted version of the upright font. In general, *serif* fonts (those with little decorations on the ends and corners of the characters) are slanted via italics, while *sans-serif* fonts are made oblique.)

Figure 10-5 compares the medium and bold weights, and the roman and italic or oblique slants in the Helvetica font family.

-adobe-helvetica-medium-o-normal--18-180-75-75-p-98-iso8859-1
-adobe-helvetica-medium-r-normal--18-180-75-75-p-98-iso8859-1
-adobe-helvetica-bold-o-normal--18-180-75-75-p-104-iso8859-1
-adobe-helvetica-bold-r-normal--18-180-75-75-p-103-iso8859-1

Figure 10-5. The same fonts in different weights and slants

X11 and OpenWindows also include some fonts that have an in-between weight called *demibold*. Weight names are somewhat arbitrary, since a demibold weight in one family may be almost as dark as a bold weight in another.

The font naming convention also defines two counter-clockwise slants called *reverse italic* (ri) and *reverse oblique* (ro), as well as a catch-all called *other* (ot).

10.1.4 Font Sizes

Font sizes are often given in a traditional printer's measure known as a *point*. A point is approximately one seventy-second of an inch. Most of the font families are provided in the six point sizes shown in Figure 10-6.

-adobe-helvetica-medium-r-normal--8-80-75-75-p-46-iso8859-1
-adobe-helvetica-medium-r-normal--10-100-75-75-p-56-iso8859-1
-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1
-adobe-helvetica-medium-r-normal--14-140-75-75-p-77-iso8859-1
-adobe-helvetica-medium-r-normal--18-180-75-75-p-98-is
-adobe-helvetica-medium-r-normal--24-240-

Figure 10-6. The same font in six different point sizes

However, the size story doesn't end there. Newer X servers (such as Sun's OpenWindows server and the X11R5 and later servers) support scalable outline fonts that are device-independent and, thus, true-to-size regardless of the output device. These fonts can be requested in virtually any pointsize. In older releases of X, such as X11R4, fonts we re simply bitmaps. Because of the different resolution of computer monitors, a font with a given nominal point size might actually appear larger or smaller on the screen.

Most monitors on the market today have a resolution between 75 dots per inch (dpi) and 100 dots per inch. Accordingly, there are both 75-dpi and 100-dpi versions of a few of the fonts in X11 R3, and of most of them in X11R4. These separate versions of each font are stored in different directories. By setting the font search path so that the appropriate directory comes first, you can arrange to get the correct versions without having to specify them in the font name.*†

But how do you tell which kind of monitor you have?

If you have the manufacturer's specs on your monitor, they might give you this figure. But they'll more likely give you the overall resolution in rows and columns. After measuring the physical screen, you can do some rough calculations to arrive at the equivalent in dots per inch. For example, the 16-inch monitor on the Sony NEWS workstation has an advertised resolution of 1280 x 1024 pixels. The actual viewing area is approximately 13 inches wide by 10 inches high. Dividing the resolution by the size, you come up with a vertical resolution of 102.4 dpi and a horizontal resolution of 98.5 dpi.

The Sun 19-inch monitor, by contrast, has an advertised resolution of 1152 x 900 pixels. The horizontal and vertical dimensions of the viewing area are approximately 13.75 x 10.75 inches. This yields a resolution of about 84 dpi.

What happens if you select the wrong resolution for your monitor? Given the difference in the pixel size, the same size font will appear larger or smaller than the nominal point size.

For example, consider the 75- and 100-dpi versions of the 24-point charter medium italic font:

```
-bitstream-charter-medium-i-normal--25-240-75-75-p-136-iso8859-1
-bitstream-charter-medium-i-normal--33-240-100-100-p-179-iso8859-1
```

If you look at the pixel size field, you will notice that the height of the 75-dpi version is 25 pixels, while the height of the 100-dpi version is 33 pixels. If you use the 75-dpi version on the Sun, you actually get something closer to 21.5 points (75/84*24); on a 100-dpi monitor, you will actually get something closer to 18 points (75/100*24). We noticed this right away when we first began using the Sony workstation. Because of its higher resolution, the font size we had been using on the Sun appeared much smaller.

If you are working on a lower-resolution monitor, you can take advantage of this artifact to display type as large as 32 points (the size that a 24-point 100-dpi font will appear on a 75-dpi monitor). Figure 10-7 shows the 75- and 100-dpi versions of the same 24-point font, as displayed on a Sun workstation with a 19-inch monochrome monitor. As shown, neither is actually 24 points. The 75-dpi version is actually 21.5 points, as discussed above; the 100-dpi version is about 28.5 points.*

Note that the logical font-naming convention allows for different horizontal and vertical resolution values. This would allow server manufacturers to support fonts that were

† We'll talk about how to set the font search path later in this chapter.

"tuned" for their precise screen resolution. However, the fonts shipped with the generic X11 distribution all use the same horizontal and vertical resolution.†

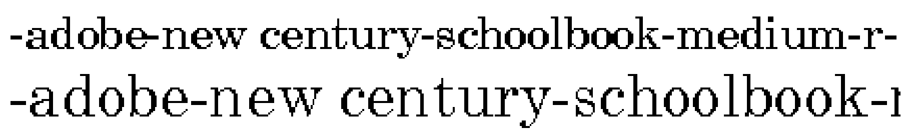

Figure 10-7. The 100-dpi version of a 24-point font appears larger on a 75-dpi monitor

As suggested above, this resolution may not exactly match the actual resolution of any particular screen, resulting in characters that are not true to their nominal point size. In the case of the Sony monitors, the actual resolution is quite close to the design of the 100-dpi fonts. However, on the Sun monitor, neither the 75- nor 100-dpi fonts will be exactly right. Unless you are running an old MIT X server that lacks scaleable fonts, however, this won't be much of a problem.)

10.1.5 Other Information in the Font Name

What we've already shown summarizes the most important information in the font name. The remaining fields are explained below:

Foundry Font manufacturers are still referred to as foundries, from the days when type was founded, or cast, from lead. The X font naming convention specifies the foundry as the company that digitized or last modified the font, rather than its original creator. For the fonts contained in the standard X distribution, the foundry is not terribly significant since there are no cases where the same font family is available from different foundries. However, there are numerous commercial font families available from more than one foundry. In general, the appearance of the fonts should be quite similar since the font family defines the design of the typeface. However, there may be some small differences in the quality of some of the characters, and there may be more significant differences in the font metrics (the vertical or horizontal measurements of the characters). This might be significant for a publishing application that was using the bitmapped font for a *wysiwyg** screen display that needed to match the fonts in a particular laser printer or typesetter.‡

"Set width" A value describing a font's proportionate width, according to the foundry. Typical set widths include: normal, condensed, semicondensed, narrow, double

† Note that the differences are exaggerated further in printing the screen dump of this display, since we scaled the bitmap up somewhat to print it..

‡ This is an acronym for "what you see is what you get" and describes a type of text editor or word processor that purports to display the page exactly as it would appear in print. There is always some approximation due to the differences between screen fonts and printer fonts. MacWrite® is a *wysiwyg* program in the MacIntosh world; FrameMaker, Interleaf and others offer *wysiwyg* programs in the X11 world.

width. All of the X11 Release 3 fonts and most of the Release 4 fonts have the set width *normal*. A few of the Release 4 fonts have the set width *semicondensed*.

Spacing All standard X11 Release 3 fonts are either m (monospace, i.e., fixed-width) or p (proportional, i.e., variable-width). In Release 4, fonts may also have the spacing characteristic c (character cell, a fixed-width font based on the traditional typewriter model, in which each character can be thought to take up the space of a "box" of the same height and width). As mentioned earlier, the original R2 fonts were this type.

Average width Mean width of all characters in the font, measured in tenths of a pixel. You'll notice, if you look back at Figure 10-2, that two fonts with the same point size (such as New Century Schoolbook and Times) can have a very different average character width. This field can sometimes be useful if you are looking for a font that is especially wide or especially narrow. The Schumacher Clean family of fonts offers several fonts in the same point size but with different average widths.†

Character set In the initial illustration of the font naming convention, Figure 10-1, we identified the character set as a single field. If you look more closely, you'll realize it is actually two fields, the first of which identifies the organization or standard registering the character set, the second of which identifies the actual character set.

Most fonts in the standard X distribution contain the string "iso8859-1" in their names, which represents the ISO Latin-1 character set. The ISO Latin-1 character set is a superset of the standard ASCII character set, which includes various special characters used in European languages other than English. See Appendix H of Volume Two, *Xlib Reference Manual*, for a complete listing of the characters in the ISO Latin-1 character set.

Note, however, that the symbol font contains the strings "adobe-fontspecific" in this position. This means that Adobe Systems defined the character set in this font, and that it is font-specific. You can see from this example that the use of these fields is somewhat arbitrary.

Style Not represented in the example or in most R3 or R4 font names. However, according to the logical font convention, the style of a font may be specified in the field between set width and pixels. Some of the possible styles are *i* (informal), *r* (roman), *serif* and *sans* (serif). Note that the *r* for roman may also be used in the slant field.

If you need still more detail and want a complete technical description of font naming conventions, see the X Consortium Standard, *X Logical Font Description Conventions*. This document is available as part of the standard MIT X distribution, and is reprinted as an Appendix in Volume Zero, *X Protocol Reference Manual*.

† These fonts all (incorrectly to our minds) have a set width of "normal." They should be distinguished by set widths such as condensed, semi-condensed, etc. Since they do not, they can be distinguished by the difference in their average width.

10.2 Font Name Wildcarding

In order to simplify font specification, X recognizes wildcarding within font names. You can use wildcarded font names to specify the text font for a client, either on the command line or in an X resource specification. You can also supply a wildcarded font name to any of the font utilities, such as *xlsfonts*, *xfd*, etc.

An asterisk (*) can be used to represent any part of the font name string; a question mark (?) can be used to represent any single character. You can usually get the font you want by specifying only the font family, the weight, the slant, and the point size, and wildcarding the rest. For example, to get Courier bold at 14 points, you could use the command line option:

```
-fn '*courier-bold-r*140*'
```

That's starting to seem a little more intuitive!

However, there are a number of "gotchas."

- First, since the UNIX shell also has a special meaning for the "&", "*" and "?" wildcard characters, wildcarded font names must be quoted. This can be done by enclosing the entire font name in quotes (as in the previous example), or by "quoting" each wildcard character by typing a backslash before it. (If you don't do this, the shell will try to expand the * to match any filenames in the current directory and, if you use the C shell, will give the message "No match." and not run the command) Wildcards need not be quoted in resource files.
- Second, if the wildcarded font name matches more than one font, the server will use the first one that matches. And unfortunately, because the font names may differ from one server to the next and are sorted in simple alphabetical order, it is indeterminate what font you will get. On X11R4, since the bold weight sorts before medium, and italic and oblique slants before roman, the specification:

```
-fn '*courier*'
```

gave Courier bold oblique, rather than the Courier medium roman you might intuitively expect. On OpenWindows 3.1, it yields a fairly small, but medium roman, Courier:

```
% xlsfonts '*courier*' | head -1
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
%
```

As shown, you can use *xlsfonts ... | head -1* to find out what font is matched. If you aren't sure whether your wildcarded name is specific enough, try using it as an argument to *xlsfonts*. If you get more than one font name as output, you may not get what you want. Try again with a more specific name string.

The exception to this rule has to do with the *75dpi* and *100dpi* directories. If a wildcard matches otherwise identical fonts in these two directories, the server will actually use the one in the directory that comes first in the font path. This means that you should put the appropriate directory first in the font path. (We'll tell you how to do this in the next section.) Thereafter, you can generally wildcard the resolution fields (unless you specifically want a font from the directory later in the path).†

Third, the * wildcard expansion is resolved by a simple string comparison. So, for example, if you type:

```
-fn '*courier-bold*r*140*'
```

instead of:

```
-fn '*courier-bold-r*140*'
```

(the difference being the asterisk instead of the hyphen before the "r" in the slant field), the "r" would also match the "r" in the string "normal" in the set width field. The result is that you would select all slants. Since o (oblique) comes before r (roman), and you always get the first font that matches, you'd end up with Courier oblique.

The trick is to be sure to include at least one of the hyphens to set the -r- off as a separate field rather than as part of another string.

Even though a wildcarded name such as:

```
*cour*b*r-*140*
```

should get you 14-point Courier bold roman, we think it is good practice to spell out the font family and weight and use hyphens between adjacent fields. As usual there are exceptions: the Lucida family really has three subfamilies; you can get all three by specifying the family as "Lucida*" rather than "Lucida-"; and you might certainly want to abbreviate "New Century Schoolbook" to "New Century*" or "*Schoolbook."

Font names are case-insensitive. "Courier" is the same as "courier."

✗ Some early versions of the SCO "Open DeskTop" server fail to be case-insensitive in fontnames.

Table 10-2 summarizes the values you can use to specify a unique font name (assuming only the standard fonts are loaded). Choose one element from each column. Don't forget to include the leading and trailing asterisks and the hyphen before the slant.

Table 10-2. Essential Elements of a Font Name

*	Family	-	Weight	-	Slant*	-	Point Size	*
	Charter		Medium		r (roman)		80 (8 pt.)	
	Courier		Bold		i (italic)		100 (10 pt.)	
	Helvetica		Demibold		o (oblique)		120 (12 pt.)	

† Unlike *xfontsel*, which displays fonts in the order of wildcard matches, *xlsfonts* will always list fonts in straight-sort order, with the sort done character by character across the line. Since size in pixels comes before point size in the name, and the size in pixels of the 100-dpi fonts is larger than that of the equivalent 75-dpi font, the 75-dpi font will always be listed first for a given point size. But when listing more than one point size, the fonts will be jumbled. For example, the size in pixels of the 8-point Charter font at 100-dpi is 11, so it will come after the 10-point Charter font at 75 dpi, with a size in pixels of 10. The 8-point Charter font at 75 dpi gets sorted to the very end of the list, since to a character-by-character sort, its size in pixels (8) looks larger to the size in pixels of even the largest 100-dpi font (the 24 point, with a height of 33 pixels).

Table 10-2. Essential Elements of a Font Name

*	Family	-	Weight	-	Slant*	-	Point Size	*
	New century				ri (reverse italic)		140 (14 pt.)	
	schoolbook							
	Symbol				ro (reverse oblique)		180 (18 pt.)	
	Times				ot (other)		240 (24 pt.)	
	Fixed (R4)							
	Clean (R4)							
	OPEN LOOK (R4)							
	Lucida (R4)							
	Terminal (R4)							

10.3 The Font Search Path

The OpenWindows Release 3.0 server uses the directories shown in Table 10-5 to store fonts..

Table 10-3. Standard Font Directories, OpenWindows Release 3.0

Directory	Contents
/usr/openwin/lib/fonts	Main body of fonts, mostly outline
/usr/openwin/lib/fonts/100dpi	Fixed- and variable-width fonts, 100 dpi (all font families).
/usr/openwin/lib/fonts/afm	Adobe Font Metrics
/usr/openwin/lib/fonts/map	Contains one file, *fontmaps.ps*.

The OpenWindows server uses only the first of these directories as its font search path. The normal font directory is *$OPENWINHOME/lib/fonts*, and this is the default setting of the environment variable FONTPATH. FONTPATH is a colon-separated list of directories, like the normal UNIX environment PATH. For example, to add a local font directory, you might set FONTPATH to, the following (and `export` it if using the Bourne or Korn shell):

```
/usr/openwin/lib/fonts:/usr/local/lib/fonts
```

Then if you re-start the OpenWindows server, any fonts it recognizes in the local font directory will automatically be available.

The OpenWindows 3.3 server, by contrast, uses the following font directories:

Table 10-4. Standard Font Directories,OpenWindows 3.3

Directory	Contents
/usr/openwin/lib/X11/fonts/F3/	Folio Format scaleable fonts
/usr/openwin/lib/X11/fonts/F3bitmaps/	Folio Format pre-scaled fonts
/usr/openwin/lib/X11/fonts/Type1/	PostScript Type 1 scaleable fonts
/usr/openwin/lib/X11/fonts/Speedo/	Speedo format scaleable fonts
/usr/openwin/lib/X11/fonts/misc/	Same as X11 misc fonts
/usr/openwin/lib/X11/fonts/75dpi/	Many 75dpi fonts
/usr/openwin/lib/X11/fonts/100dpi/	Many 100dpi fonts
/usr/openwin/lib/X11/fonts/Xt+/	Glyph fonts for OLIT toolkit

In the MIT X server and its derivatives, fonts are stored in three directories, as shown in Table 10-5.

Table 10-5. Standard Font Directories, X11 Release 4

Directory	**Contents**
/usr/lib/X11/fonts/misc	Sixty fixed-width fonts, including the six available in Release 3, the cursor font, several Clean family fonts provided by Schumacher, a Kanji font, Kana fonts, and OPEN LOOK cursor and glyph fonts.
/usr/lib/X11/fonts/75dpi	Fixed- and variable-width fonts, 75 dpi.
/usr/lib/X11/fonts/100dpi	Fixed- and variable-width fonts, 100 dpi (all font families).

These three directories (in this order) constitute X's default font path.

If you wish to provide additional fonts, other directories can be added to the font search path, or its order can be rearranged, using *xset* with the `fp` option. To completely replace the font path, simply specify a comma-separated list of directories. For example, to put the *100dpi* directory before the *75dpi* directory with the MIT server, you might enter:

```
% xset fp=
          /usr/lib/X11/fonts/misc,/usr/lib/X11/fonts/100dpi
          ,\ /usr/lib/X11/fonts/75dpi
```

(Note that a space must follow the equal sign (=) and that the example above is broken onto two lines escaped with a backslash (\) only so that it can be printed within the page margins.) To restore the default font path, type:

```
% xset fp default
```

Use the `fp+` option to add a directory or list of directories to the end of the font path, or `+fp` to add them at the start. Use `-fp` and `fp-` to delete directories from the beginning or end of the font path.

For a complete list of the fonts in each directory and samples of each font, refer to Appendix B, *OpenWindows and X11 Fonts*.

10.3.1 The fonts.dir Files (Standard X server)

In addition to font files, the MIT X server requires each font directory to contain a file called *fonts.dir*. The *fonts.dir* files serve, in effect, as databases for the X server. When the X server searches the directories in the default font path, it uses the *fonts.dir* files to locate the font(s) it needs.

Each *fonts.dir* file contains a list of all the font files in the directory with their associated font names in two-column form. (The first column lists the font file name and the second column lists the actual font name associated with the file.) The first line in *fonts.dir* lists the number of entries in the file (i.e., the number of fonts in the directory).

Example 10-1 shows a portion of the *fonts.dir* file from the Release 4 */usr/lib/X11/fonts/100dpi* directory. As the first line indicates, the directory contains 200 fonts. The first group of fonts listed below (up to the second ellipse) are available as of X11 Release 4. They are all Courier family fonts. (These fonts are 100-dpi equivalents of fonts that were only available in 75 dpi in X11 Release 3.) The second group of fonts shown in the list below (a few sizes from the Charter family) are also available in the X11 Release 3 *100dpi* directory.

```
200
courBO08.snf -adobe-courier-bold-o-normal--11-80-100-100-m-60-
iso8859-1
courBO10.snf -adobe-courier-bold-o-normal--14-100-100-100-m-90-
iso8859-1
courBO12.snf -adobe-courier-bold-o-normal--17-120-100-100-m-100-
iso8859-1
courBO14.snf -adobe-courier-bold-o-normal--20-140-100-100-m-110-
iso8859-1 c
ourBO18.snf -adobe-courier-bold-o-normal--25-180-100-100-m-150-
iso8859-1
```

```
courBO24.snf -adobe-courier-bold-o-normal--34-240-100-100-m-200-
iso8859-1
courB08.snf -adobe-courier-bold-r-normal--11-80-100-100-m-60-iso8859-
1
courB10.snf -adobe-courier-bold-r-normal--14-100-100-100-m-90-
iso8859-1
courB12.snf -adobe-courier-bold-r-normal--17-120-100-100-m-100-
iso8859-1
courB14.snf -adobe-courier-bold-r-normal--20-140-100-100-m-110-
iso8859-1
courB18.snf -adobe-courier-bold-r-normal--25-180-100-100-m-150-
iso8859-1
courB24.snf -adobe-courier-bold-r-normal--34-240-100-100-m-200-
iso8859-1
courO08.snf -adobe-courier-medium-o-normal--11-80-100-100-m-60-
iso8859-1
courO10.snf -adobe-courier-medium-o-normal--14-100-100-100-m-90-
iso8859-1
courO12.snf -adobe-courier-medium-o-normal--17-120-100-100-m-100-
iso8859-1
courO14.snf -adobe-courier-medium-o-normal--20-140-100-100-m-110-
iso8859-1
. . .
```

Example 10-1Subsection of the Release 4 fonts.dir file in /usr/lib/X11/fonts/100dpi

The *fonts.dir* files are created by the *mkfontdir* client when X is installed. *mkfontdir* reads the font files in directories in the font path, extracts the font names, and creates a *fonts.dir* file in each directory. If *fonts.dir* files are present on your system, you probably won't have to deal with them or with *mkfontdir* at all. If the files are not present, or if you have to load new fonts or remove existing ones, you will have to create files with *mkfontdir*. Refer to Volume Eight, *X Window System Administrator's Guide*, for details.

10.4 Font Name Aliasing

Another way to abbreviate font names is by aliasing (that is, by associating fonts with alternative names of your own choosing).

10.4.1 Aliases—X11R5 and OpenWindows 3.3 Server

You can edit or create a file called *fonts.alias*, in any directory (or multiple directories) in the font search path, to set aliases for existing fonts. The X server uses both *fonts.dir* files and *fonts.alias* files to locate fonts in the font path.

If you are running X11 , there should already be an alias file in each font directory. Take the time to look at the contents of each of these files, since many of the existing aliases may be easier to type than even wildcarded font names. You can also add aliases to the file, change existing aliases, or even replace the entire file. However, this should be done with caution. To play it safe, it's probably a good idea merely to *add* to existing *fonts.alias* files. If you're working in a multiuser environment, the system administrator should definitely be consulted before aliases are added or changed. Note that when you create or edit a *fonts.alias* file, the server does not *automatically* recognize the aliases in question. You must make the server aware of newly created or edited alias files by resetting the font path with *xset*.

The *fonts.alias* file has a two-column format similar to the *fonts.dir* file: the first column contains aliases, the second contains the actual font names. If you want to specify an alias that contains spaces, enclose the alias in double quotes. If you want to include double quotes (") or other special characters as part of an alias, precede each special symbol with a backslash ("\".

When you use an alias to specify a font in a command line, the server searches for the font associated with that alias in every directory in the font path. Therefore, a *fonts.alias* file in one directory can set aliases for fonts in other directories as well. You might choose to create a single aliases file in one directory of the font path to set aliases for the most commonly used fonts in all the directories. Example 10-2 shows three sample entries that could be added to an existing *fonts.alias* file (or constitute a new one).

```
xterm12-adobe-courier-medium-r-normal--12-120-75-75-m-70-iso8859-1
xterm14-adobe-courier-medium-r-normal--14-140-75-75-m-90-iso8859-1
xterm18-adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1
```

Example 10-2. Sample fonts.alias file entries

As the names of the aliases suggest, these sample entries provide aliases for three fonts (of different point sizes) that are easily readable in *xterm* windows. (We also recommend the fixed-width font stored in the file *9x15.snf*,* in the *misc* directory.) You can also use wildcards within the font names in the right column of an alias file. For instance, the alias file entries above might also be written as follows:

```
xterm12*courier-medium-r-*-120*
xterm14*courier-medium-r-*-140*
xterm18*courier-medium-r-*-180*
```

Once the server is made aware of aliases, you can specify an alias on the command line. For example, you can use a font name alias as an argument to *xfd*.

A special note about the *misc* directory: when X was configured for your system, a *fonts.alias* file should have been created in this directory. The first two entries in this file are shown below:

```
fixed -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
 variable -*-helvetica-bold-r-normal-*-*-120-*-*-*-*-*-*
```

The default file contains an additional 56 entries but the entries pictured above are particularly important. The aliases called "fixed" and "variable" are invoked as the default fonts for many clients. The "fixed" font can be thought of as a system-wide default. The "variable" font, described in the right column as a 12-point bold Helvetica font, is used as the default font by *bitmap*, as well as by other clients. If this file is removed or replaced, when you run *bitmap*, you'll get an error message that the server cannot open the variable font, and text in the *bitmap* window will display in the smaller, somewhat less readable "fixed" font.

If you do choose to edit the *fonts.alias* file in the *misc* directory, it is important to preserve at least these two aliases. (As we've said, it's probably a better idea to keep all the default entries and merely append any new ones.)

If you're running the older X11 Release 3, the *fonts.alias* file in the *misc* directory will be somewhat different. The X11 Release 3 version of the *fonts.alias* file in the *misc* directory comprises only these two lines:

```
fixed 6x13 variable *-helvetica-bold-r-normal-*-*-140-*
```

Regardless of what edits you make to the file, the line specifying the variable alias must not be changed.

The variable font is slightly larger in X11 Release 3 (14-point) than in X11 Release 4 (12-point). If you examine the Release 3 alias file a little more closely, you may notice that the first line contains an *incorrect* alias specification. Remember, in Release 3, *fixed* is actually the name of the default system font—it is not an alias. The first column should contain aliases; the second column should contain proper font names. However, 6x13 is not a proper font name. It is actually the name of the file that contains the font named *fixed*. You can specify *fixed* as a font on the command line and it will work—but as a font name, not an alias.

10.4.2 Aliases—Older OpenWindows XNews Server

You can provide your own aliases by creating a *.user.ps* file with the following contents. This example makes "Nice" be an alias for the existing Folio font "GillSans":

```
FontDirectory begin
    /Nice/Gill-Sans_FontDirectorySYN
    % other aliases here
end
```

The font Nice can now be used instead of Gill-Sans at any point size. As usual, you must re-start the server for changes in this file to take effect. This technique is in fact used to provide X11 names for the standard PostScript fonts. The file *$OPENWINHOME/lib/fonts/Synonyms.Lst* is comprised of hundreds of lines like this:

```
/-bitstream-charter-medium-i-normal--8-80-75-75-p-44-iso8859-1 \
    /Charter-Italic _FontDirectorySYN
```

See the Sun document *X11/NeWS Version 2 Server Guide*, part number 800-4898-10, for more details on the user's initialization files.

10.4.3 Making the Server Aware of Aliases

After you create (or update) an alias file, the server does not automatically recognize the aliases in question. You must make the server aware of newly created or edited alias files by "rehashing" the font path with *xset*. Enter:

```
% xset fp rehash
```

on the command line. The *xset* option `fp` (font path) with the `rehash` argument causes the server to reread the *fonts.dir* and *fonts.alias* files in the current font path. You need to do this every time you edit an alias file. (You also need to use *xset* if you add or remove fonts. See Appendix B, *OpenWindows and X11 Fonts*, for details.)

10.5 Utilities for Displaying Information about Fonts

We've already mentioned *xlsfonts*, which simply displays the names and aliases of available fonts. In addition, *xfd* can be used to display the full character set of a particular font, and *xfontsel* can be used to interactively preview and select a font for use in another window.

10.5.1 The Font Displayer: xfd

If you're unfamiliar with the general appearance of a particular font, we've included pictures of some representative fonts in Appendix B, *OpenWindows and X11 Fonts*.

You can also display the characters in a font using the *xfd* (font displayer) client. Note that *xfd* takes an option, `-fn`, before the font name. For example, to display the default system font, a 6x13 pixel fixed-width font known as *fixed*, enter:†

```
% xfd -fn fixed &
```

The *xfd* window will display the specified font (by its real name; *fixed* is an alias) as shown in Figure 10-8.

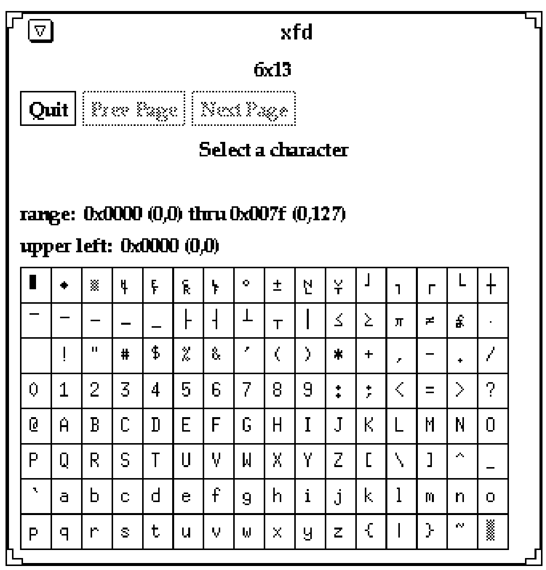

Figure 10-8. Fixed font, 6x13 pixels

† In X11 Release 3, *fixed* is a font name. In Release 4, it is an alias for a longer font name that follows the conventions outlined previously.

This figure depicts the X11 Release 4 version of *xfd*. The font name is displayed across the top of the window Three command buttons appear in the upper-left corner of the window below the font name. If the font being displayed doesn't fit within a single *xfd* screen, Prev Page and Next Page allow you to scroll through multiple screens. (The horizontal and vertical window dimensions can vary slightly to accommodate different fonts but certain fonts will still require multiple screens.) The Quit button causes the application to exit, though this can also be done by typing q or Q while input is focused on the *xfd* window.

In addition to displaying a font, *xfd* also allows you to display certain information about the individual characters. But before we examine these capabilities, let's take a closer look at the way the characters in a font are identified and how the *xfd* window makes use of this information.

Within a font, each character is considered to be numbered. The *xfd* client displays a font's characters in a grid. By default, the first character of the font appears in the upper-left position; this is character number 0. The two text lines above the grid identify the upper-left character and the range of characters in the window by character numbers both in hexadecimal and in decimal notation (in parentheses following the hex character number).

You can specify a character other than character number 0 to be in the first position in the window using the `-start` option. For example, if you enter this command line:

```
% xfd -start 15 -fn fixed &
```

the *xfd* window begins with character number 15.

Notice the instruction `Select a character` below the command buttons. To display information about a particular character, click any pointer button within the character's grid square. Statistics about the character's width, left bearing, right bearing, ascent, and descent are displayed where the line `Select a character` previously appeared.

The *xfd* client is most useful when you have an idea what font you might want to display. If you don't have a particular font in mind or would like to survey the possibilities, the *xfontsel* client (available as of X11 Release 4) allows you to preview a variety of fonts by specifying each component of the font name using a different menu.

10.5.2 Previewing and Selecting Fonts: xfontsel

The *xfontsel* client provides a font previewer window in which you select the font to view using 14 menus corresponding to the 14 components of a font name. By specifying various font name components, you can take a look at a variety of fonts. This is particularly useful if you are trying to pick good display fonts and you don't have a clear idea what type of font would be best. Rather than running several instances of *xfd*, you can dynamically change the font displayed in the *xfontsel* window by changing the font name components. (Despite the flexibility of *xfontsel*, it's certainly not practical to preview *all* of the available fonts. If you have no idea what a particular font family looks like, see the discussion earlier in this chapter, or refer to Appendix B, *OpenWindows and X11 Fonts*, for complete listings.)†

Once you've displayed the desired font using the menus, you can make the name of that font the PRIMARY text selection by clicking on the window's select button. You can then paste the font name into another window using the pointer: onto a command line, into a resource file, etc. Making a font name the PRIMARY selection also enables you to choose that font from the *xterm* VT Fonts menu. (Selecting text and using *xterm* menus are described in Chapter 5, *The xterm Terminal Emulator.*

10.5.2.1 Previewing Fonts with the xfontsel Menus

To run *xfontsel*, enter this command in an *xterm* window:

```
% xfontsel &
```

The *xfontsel* window initially displays a some randomly-chosen font such as a greek symbol font or a bold, constant-width, 7x13 pixel font, as shown in Figure 10-9. In fact it displays the first font in the default font search path.

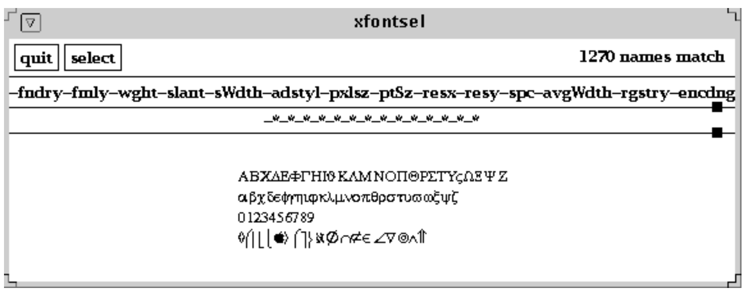

Figure 10-9. xfontsel window displaying random font

The upper-left corner of the *xfontsel* window features two command buttons: *quit* and *select*. As we've explained, clicking on *select* with the first pointer button makes the font displayed in the window the PRIMARY text selection; obviously, *quit* causes the application to exit.

Below the command buttons is, in effect, a generic font name or font name template. It is divided into 14 fields corresponding to the 14 parts of a standard font name. Each field is an abbreviation for one part of a font name. Take a look again at the sample font name in Figure 10-1 to refresh your memory as to the components. Each of the fields in the *xfontsel* window is actually the handle to a menu which lets you specify this part of the font name.

To get a clearer idea of how this works, move the pointer onto the generic font name—specifically onto the first field, fndry. (This is an abbreviation for the first part of a font name, the foundry.) When you place the pointer on fndry, the field title should be highlighted by

† To our minds, the major drawback of *xfontsel* is that it shows you only the first font that matches a given wildcarded font name. A far better interface would list all of the matching fonts so that you could compare and choose the one that best suited your needs. There is no way in the standard X distribution to display the appearance of a group of fonts. To produce the figures in this book, we had to write such a program, which we called *xshowfonts*. The program has since been posted to *comp.sources.x*, and a listing appears in Appendix B, *OpenWindows and X11 Fonts*.

a box. You can then display a menu of foundry names by pressing and holding down the first pointer button, as in Figure 10-10

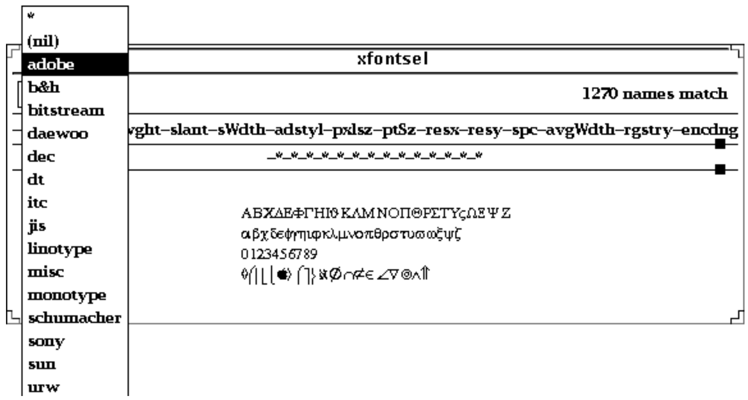

Figure 10-10. xfontsel window with foundry menu displayed

Notice that the first choice is the asterisk (*) wildcard character. This is the first choice on all of the menus, allowing you to include wildcards in the font name you specify rather than explicitly selecting something from all 14 menus.

To specify a font name component (i.e., make a selection from the menu), first display the menu by pressing and holding down the first pointer button. Then move the pointer down the menu. As the pointer rests on each menu item, it is highlighted by reverse video. To select a highlighted menu item, release the first pointer button.

The line below the font name menus represents the actual font name. When you first run *xfontsel*, all of these fields contain wildcard characters because no menu selections have been made. The number of fonts matched by the font name is displayed in the upper-right corner of the window. The number of fonts initially matched depends on the number of fonts with this naming convention available on your system. In this example, 474 fonts match. (Since this line of wildcards can match *any* 14-part font name, the server chooses the first font in the font path that reflects this naming convention.)

When you select a font name component from one of the 14 menus, the component appears in the actual font name, and the *xfontsel* window displays the first font that matches this name. For example, say we select adobe from the fndry menu, the *xfontsel* window would look like Figure 10-11.

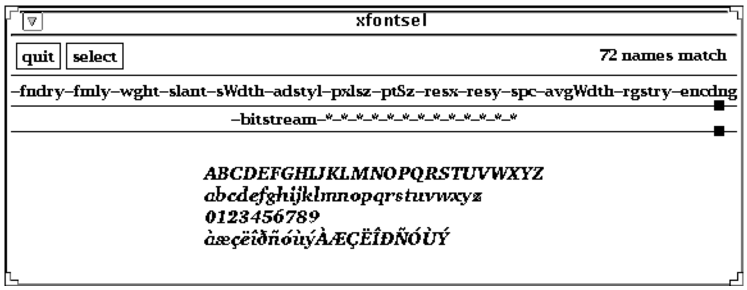

Figure 10-11. xfontsel after choosing Bitstream from the foundry menu

The font name is now:

```
-bitstream-*-*-*-*-*-*-*-*-*-*-*-*-*
```

and the window displays the first font in the font path that matches this wildcarded name. In this case, the first font to match is a 12-point bold Oblique Courier font, which is stored in the file *courBO10.snf* and has the actual font name:

```
-bitstream-charter-bold-o-normal-*-10-100-75-75-m-60-iso8859-1
```

Once you make a selection from one menu, the number of possible fonts matched by the name changes. (Notice the line `nnn fonts match` in the upper-right corner of the window.) Choosing one font name component also eliminates certain choices on other menus. For example, after you select Bitstream as the foundry, the possible choices for font family (the second menu, fmly) are narrowed down to 2 (not counting the asterisk). Again display the fmly menu using the first pointer button. The available choices for font family appear in a regular typeface; the items that are unavailable (i.e., cannot be selected) appear in a lighter typeface. Families such as Clean, Lucida, and Charter are in a lighter typeface because none of the standard X fonts provided by Bitstream is from these families. Bitstream fonts in the standard X distribution are limited to the families Courier, and Charter; these are the items available on the fmly menu.

In order to display a particular font, you'll probably have to make selections from several of the menus. As described earlier in the section "Font Name Wildcarding," we suggest you explicitly select at least these parts of the font name:

- Font family
- Weight
- Slant
- Point size

Thus, you would make selections from the fmly, wght, slant, and ptSz menus.

You can also use the *-pattern* option with a wildcarded font name to start out with a more limited range of options. For example, if you typed:

```
% xfontsel -pattern '*charter-bold-o-*'
```

you'd start out with the pattern you specified in the filename template part of the *xfontsel* display. You could then simply select from the ptSz menu to compare the various point sizes of Courier bold oblique until you found the one you wanted.

Note that if the pattern you specify to *xfontsel* matches more than one font, the one that is displayed (the first match found) is the one that the server will use. This is in contrast to *xlsfonts*, which sorts the font names. You can always rely on *xfontsel* to show you the actual font that will be chosen, given any wildcard specification.

10.5.2.2 Selecting a Font Name

Once you make selections from the menus to compose the name of the font you want, the corresponding font is displayed in the *xfontsel* window. Then you can select that font name by clicking on the select command button with the first pointer button. The font name becomes the PRIMARY text selection and thus can be pasted in another window using the second pointer button, as described in Appendix A, *The xterm/olterm Terminal Emulator*.

You might paste the font name on a client command line in an *xterm* window in order to specify it as the client's display font. (See Chapter 11, *Command Line Options*.) You might paste it into a resource file such as *.Xdefaults* to specify it as the default font for a client or some feature of a client (such as a menu). (See Chapter 12, *Setting Resources*, for more information.)

Less obviously, once a font name is made the PRIMARY text selection, it can be toggled as the *xterm* display font using the Selection item of the *xterm* VT Fonts menu. The Selection menu item can only be chosen from the VT Fonts menu when there is a PRIMARY text selection. (Otherwise, the menu item appears in a lighter typeface, indicating that it is not available.) If the PRIMARY text selection is a valid font name (as it is when you've pressed the select button in the *xfontsel* window), the *xterm* window displays in that font. (In cases where the PRIMARY selection is not a valid font name, the *xterm* display font does not change.)

By default, *xfontsel* displays the lowercase and uppercase letters a through z and the digits 0 through 9. You can specify alternative sample text using the `-sample` option. For more information about this and other options, see the *xfontsel* reference page in Part Three of this guide.

10.5.3 The "text" demo program (OpenWindows up to 3.2)

OpenWindows Release Two includes a useful program for displaying text in various fonts. The *text* program is stored in the demos directory, which is usually not in your standard directory search path. It is a PostScript script file, so you can invoke it using *psh* as follows:

```
% cd $OPENWINHOME/demos
% psh ./text
```

The initial screen (Figure 10-12)shows some silly text in the default typeface:

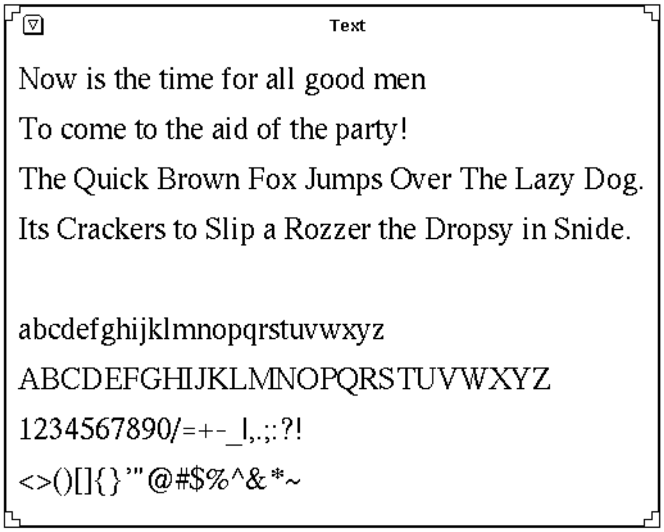

Figure 10-12. Text Demo.

The MENU button in the program's window displays this menu: l. Fonts Point Size Gray Level Contrast Show Text Show All The last two are used to toggle between the default display above, and the display of the full character set in any given font, as shown in Figure 10-13.

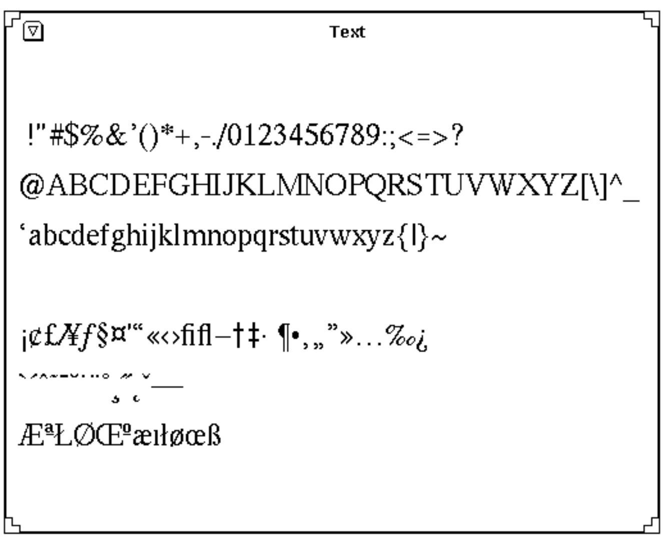

Figure 10-13. Text Demo.

The first item in the menu, *fonts*, expands to a (very large) list of all the available font names. Note that it takes quite a while to prepare this menu, as the X/NeWS server has to examine each font file at its disposal. Figure 10-14 is the list on my system:

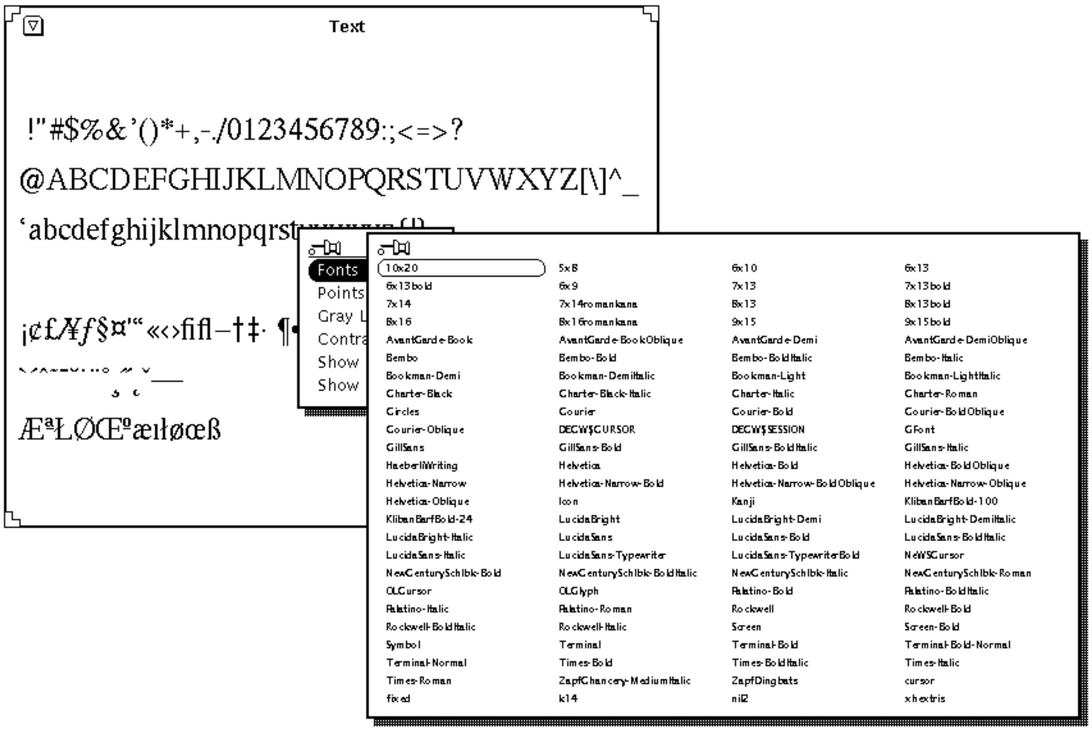

Figure 10-14. ***Text*** Demo: List of Fonts

If you click on a large font, you will only see the first part of it; use *xfd* mentioned above to see it all. For example, if you click on *Kanji*, you see a window like Figure 10-15.

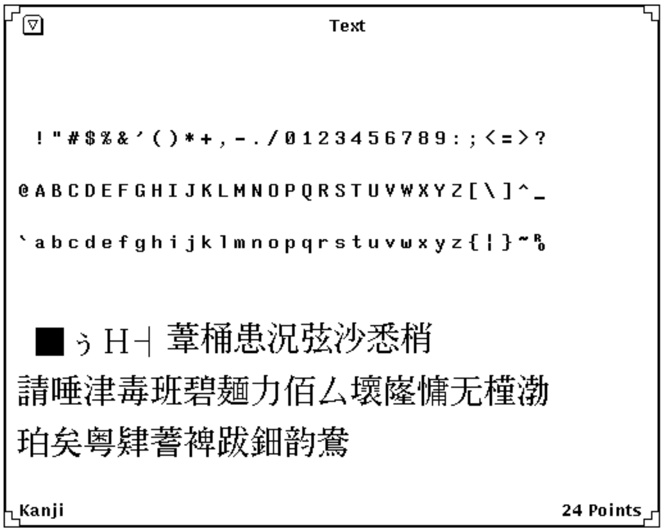

Figure 10-15. Text demo: Kanji Font (first page of many).

Note that the font menu is pinnable; if you are planning to experiment with a variety of fonts, you can (and should) pin this menu up.

The *Point Size* menu lets you view the text sample at a variety of sizes. The remaining items, *Gray Level* and *Contrast* allow you to view the text in a variety of shadings and contrasts, including reverse video.

All in all, *text* is a useful tool for previewing all the available fonts in the OpenWindows system.