Fonts in X11R7.6

Juliusz Chroboczek <jch@freedesktop.org>

23 October 2009

Table of Contents

Introduction

This document describes the support for fonts in X11R7.6. Installing fonts is aimed at the casual user wishing to install fonts in X11R7.6 the rest of the document describes the font support in more detail.

We assume some familiarity with digital fonts. If anything is not clear to you, please consult Appendix: Background at the end of this document for background information.

Two font systems

X11 includes two font systems: the original core X11 fonts system, which is present in all implementations of X11, and the Xft fonts system, which may not yet be distributed with implementations of X11 that are not based on either XFree86 or X11R6.8 or later.

The core X11 fonts system is directly derived from the fonts system included with X11R1 in 1987, which could only use monochrome bitmap fonts. Over the years, it has been more or less happily coerced into dealing with scalable fonts and rotated glyphs.

Xft was designed from the start to provide good support for scalable fonts, and to do so efficiently. Unlike the core fonts system, it supports features such as anti-aliasing and sub-pixel rasterisation. Perhaps more importantly, it gives applications full control over the way glyphs are rendered, making fine typesetting and WYSIWIG display possible. Finally, it allows applications to use fonts that are not installed system-wide for displaying documents with embedded fonts.

Xft is not compatible with the core fonts system: usage of Xft requires fairly extensive changes to toolkits (user-interface libraries). While X.Org will continue to maintain the core fonts system, toolkit authors are encouraged to switch to Xft as soon as possible.

Installing fonts

This section explains how to configure both Xft and the core fonts system to access newly-installed fonts.

Configuring Xft

Xft has no configuration mechanism itself, it relies upon the fontconfig [http://www.fontconfig.org/] library to configure and customise fonts. That library is not specific to the X Window system, and does not rely on any particular font output mechanism.

Installing fonts in Xft

Fontconfig looks for fonts in a set of well-known directories that include all of X11R7.6's standard font directories (`` `/usr/share/fonts/X11/*' `` by default) as well as a directory called `` `.fonts/' `` in the user's home directory. Installing a font for use by Xft applications is as simple as copying a font file into one of these directories.

```
$ cp lucbr.ttf ~/.fonts/
```

Fontconfig will notice the new font at the next opportunity and rebuild its list of fonts. If you want to trigger this update from the command line, you may run the command `**fc-cache**'.

```
$ fc-cache
```

In order to globally update the system-wide Fontconfig information on Unix systems, you will typically need to run this command as root:

```
$ su -c fc-cache
```

Fine-tuning Xft

Fontconfig's behaviour is controlled by a set of configuration files: a standard configuration file, `` `/etc/fonts/fonts.conf' ``, a host-specific configuration file, `` `/etc/fonts/local.conf' ``, and a user-specific file called `` `.fonts.conf' `` in the user's home directory (this can be overridden with the `` `FONTCONFIG_FILE' `` environment variable).

Every Fontconfig configuration file must start with the following boilerplate:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

In addition, every Fontconfig configuration file must end with the following line:

```
</fontconfig>
```

The default Fontconfig configuration file includes the directory ``˜/.fonts/' in the list of directories searched for font files, and this is where user-specific font files should be installed. In the unlikely case that a new font directory needs to be added, this can be done with the following syntax:

```
<dir>/usr/local/share/fonts/</dir>
```

Another useful option is the ability to disable anti-aliasing (font smoothing) for selected fonts. This can be done with the following syntax:

```
<match target="font">
    <test qual="any" name="family">
        <string>Lucida Console</string>
    </test>
    <edit name="antialias" mode="assign">
        <bool>false</bool>
    </edit>
</match>
```

Anti-aliasing can be disabled for all fonts by the following incantation:

```
<match target="font">
    <edit name="antialias" mode="assign">
        <bool>false</bool>
    </edit>
</match>
```

Xft supports sub-pixel rasterisation on LCD displays. X11R7.6 should automatically enable this feature on laptops and when using an LCD monitor connected with a DVI cable; you can check whether this was done by typing

```
$ xdpyinfo -ext RENDER | grep sub-pixel
```

If this doesn't print anything, you will need to configure Render for your particular LCD hardware manually; this is done with the following syntax:

```
<match target="font">
    <edit name="rgba" mode="assign">
        <const>rgb</const>
    </edit>
```

```
</match>
```

The string `` `rgb' `` within the `` `<const>' ``...`` `</const>' `` specifies the order of pixel components on your display, and should be changed to match your hardware; it can be one of `` `rgb `` (normal LCD screen), `` `bgr' `` (backwards LCD screen), `` `vrgb' `` (LCD screen rotated clockwise) or `` `vbgr' `` (LCD screen rotated counterclockwise).

Configuring applications

A growing number of applications use Xft in preference to the core fonts system. Some applications, however, need to be explicitly configured to use Xft.

A case in point is XTerm, which can be set to use Xft by using the `` `-fa' `` command line option or by setting the `` `XTerm*faceName' `` resource:

```
XTerm*faceName: Courier
```

or

```
$ xterm -fa "Courier"
```

For KDE applications, you should select ``Anti-alias fonts'' in the ``Fonts'' panel of KDE's ``Control Center''. Note that this option is misnamed: it switches KDE to using Xft but doesn't enable anti-aliasing in case it was disabled by your Xft configuration file.

Gnome applications and Mozilla Firefox will use Xft by default.

Configuring the core X11 fonts system

Installing fonts in the core system is a two step process. First, you need to create a *font directory* that contains all the relevant font files as well as some index files. You then need to inform the X server of the existence of this new directory by including it in the *font path*.

Installing bitmap fonts

The X11R7.6 server can use bitmap fonts in both the cross-platform BDF format and the somewhat more efficient binary PCF format. (X11R7.6 also supports the obsolete SNF format.)

Bitmap fonts are normally distributed in the BDF format. Before installing such fonts, it is desirable (but not absolutely necessary) to convert the font files to the PCF format. This is done by using the command `**bdftopcf**', *e.g.*

```
$ bdftopcf courier12.bdf
```

You will then want to compress the resulting PCF font files:

```
$ gzip courier12.pcf
```

After the fonts have been converted, you should copy all the font files that you wish to make available into a arbitrary directory, say `` `/usr/local/share/fonts/bitmap/' ``. You should then create the index file `` `fonts.dir' `` by running the command \`**mkfontdir**' (please see the mkfontdir(1) [mkfontdir.1.html] manual page for more information):

```
$ mkdir /usr/local/share/fonts/bitmap/
$ cp *.pcf.gz /usr/local/share/fonts/bitmap/
$ mkfontdir /usr/local/share/fonts/bitmap/
```

All that remains is to tell the X server about the existence of the new font directory; see Setting the server font path below.

Installing scalable fonts

The X11R7.6 server supports scalable fonts in multiple formats, including Type 1, TrueType, and OpenType/CFF. (Earlier versions of X11 also included support for the Speedo and CID scalable font formats, but that is not included in current releases.)

Installing scalable fonts is very similar to installing bitmap fonts: you create a directory with the font files, and run \`**mkfontdir**' to create an index file called `` `fonts.dir' ``.

There is, however, a big difference: \`**mkfontdir**' cannot automatically recognise scalable font files. For that reason, you must first index all the font files in a file called `` `fonts.scale' ``. While this can be done by hand, it is best done by using the \`**mkfontscale**' utility.

```
$ mkfontscale /usr/local/share/fonts/Type1/
$ mkfontdir /usr/local/share/fonts/Type1/
```

Under some circumstances, it may be necessary to modify the `` `fonts.scale' `` file generated by **mkfontscale**; for more information, please see the mkfontdir(1) [mkfontdir.1.html] and mkfontscale(1) [mkfontscale.1.html] manual pages and Core fonts and internationalisation later in this document.

CID-keyed fonts

The CID-keyed font format was designed by Adobe Systems for fonts with large character sets. The CID-keyed format is obsolete, as it has been superseded by other formats such as OpenType/CFF and support for CID-keyed fonts has been removed from X11.

Setting the server's font path

The list of directories where the server looks for fonts is known as the *font path*. Informing the server of the existence of a new font directory consists of putting it on the font path.

The font path is an ordered list; if a client's request matches multiple fonts, the first one in the font path is the one that gets used. When matching fonts, the server makes two passes over the font path: during the first pass, it searches for an exact match; during the second, it searches for fonts suitable for scaling.

For best results, scalable fonts should appear in the font path before the bitmap fonts; this way, the server will prefer bitmap fonts to scalable fonts when an exact match is possible, but will avoid scaling bitmap fonts when a scalable font can be used. (The `` `:unscaled' `` hack, while still supported, should no longer be necessary in X11R7.6.)

You may check the font path of the running server by typing the command

```
$ xset q
```

Font path catalogue directories

You can specify a special kind of font path directory in the form `catalogue:<dir>`. The directory specified after the `catalogue:` prefix will be scanned for symlinks and each symlink destination will be added as a local font path entry.

The symlink can be suffixed by attributes such as '`unscaled`', which will be passed through to the underlying font path entry. The only exception is the newly introduced '`pri`' attribute, which will be used for ordering the font paths specified by the symlinks.

An example configuration:

```
75dpi:unscaled:pri=20 -> /usr/share/X11/fonts/75dpi
ghostscript:pri=60 -> /usr/share/fonts/default/ghostscript
misc:unscaled:pri=10 -> /usr/share/X11/fonts/misc
type1:pri=40 -> /usr/share/X11/fonts/Type1
type1:pri=50 -> /usr/share/fonts/default/Type1
```

This will add `/usr/share/X11/fonts/misc` as the first font path entry with the attribute `unscaled`. This is functionally equivalent to setting the following font path:

```
/usr/share/X11/fonts/misc:unscaled,
/usr/share/X11/fonts/75dpi:unscaled,
/usr/share/X11/fonts/Type1,
/usr/share/fonts/default/Type1,
/usr/share/fonts/default/ghostscript
```

Temporary modification of the font path

The `**xset**' utility may be used to modify the font path for the current session. The font path is set with the command **xset fp**; a new element is added to the front with **xset +fp**, and added to the end with **xset fp+**. For example,

```
$ xset +fp /usr/local/fonts/Type1
$ xset fp+ /usr/local/fonts/bitmap
```

Conversely, an element may be removed from the front of the font path with `**xset -fp**', and removed from the end with `**xset fp-**'. You may reset the font path to its default value with `**xset fp default**'.

For more information, please consult the xset(1) [xset.1.html] manual page.

Permanent modification of the font path

The default font path (the one used just after server startup or after `**xset fp default**') may be specified in the X server's `` `xorg.conf` ``' file. It is computed by appending all the directories mentioned in the `` `FontPath` ``' entries of the `` `Files` ``' section in the order in which they appear. If no font path is specified in a config file, the server uses a default value specified when it was built.

```
FontPath "/usr/local/fonts/Type1"
...
FontPath "/usr/local/fonts/bitmap"
```

For more information, please consult the xorg.conf(5) [xorg.conf.5.html] manual page.

Troubleshooting

If you seem to be unable to use some of the fonts you have installed, the first thing to check is that the `` `fonts.dir` ``' files are correct and that they are readable by the server (the X server usually runs as root, beware of NFS-mounted font directories). If this doesn't help, it is quite possible that you are trying to use a font in a format that is not supported by your server.

X11R7.6 supports the BDF, PCF, SNF, Type 1, TrueType, and OpenType font formats. However, not all X11R7.6 servers come with all the font backends configured in.

On most platforms, the X11R7.6 servers no longer uses font backends from modules that are loaded at runtime. The built in font support corresponds to the functionality formerly provided by these modules:

- "`bitmap`": bitmap fonts (`` `*.bdf` ``', `` `*.pcf` ``' and `` `*.snf` ``');
- "`freetype`": TrueType fonts (`` `*.ttf` ``' and `` `*.ttc` ``'), OpenType fonts (`` `*.otf` ``' and `` `*.otc` ``') and Type 1 fonts (`` `*.pfa` ``' and `` `*.pfb` ``').

Fonts included with X11R7.6

Standard bitmap fonts

The Sample Implementation of X11 (SI) comes with a large number of bitmap fonts, including the `` `fixed` ``' family, and bitmap versions of Courier, Times, Helvetica and some members of the Lucida family.

In X11R7.6, a number of these fonts are provided in Unicode-encoded font files now. At build time, these fonts are split into font files encoded according to legacy encodings, a process which allows us to provide the standard fonts in a number of regional encodings with no duplication of work.

For example, the font file

```
/usr/share/fonts/X11/misc/6x13.bdf
```

with XLFD

```
-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10646-1
```

is a Unicode-encoded version of the standard `` `fixed' `` font with added support for the Latin, Greek, Cyrillic, Georgian, Armenian, IPA and other scripts plus numerous technical symbols. It contains over 2800 glyphs, covering all characters of ISO 8859 parts 1-5, 7-10, 13-15, as well as all European IBM and Microsoft code pages, KOI8, WGL4, and the repertoires of many other character sets.

This font is used at build time for generating the font files

```
6x13-ISO8859-1.bdf
6x13-ISO8859-2.bdf
...
6x13-ISO8859-15.bdf
6x13-KOI8-R.bdf
```

with respective XLFDs

```
-misc-fixed-medium-r-normal--13-120-75-75-c-60-iso8859-1
...
-misc-fixed-medium-r-normal--13-120-75-75-c-60-iso8859-15
-misc-fixed-medium-r-normal--13-120-75-75-c-60-koi8-r
```

The standard short name `` `fixed' `` is normally an alias for

```
-misc-fixed-medium-r-normal--13-120-75-75-c-60-iso8859-1
```

The ClearlyU Unicode font family

The ClearlyU family of fonts provides a set of 12 pt, 100 dpi proportional fonts with many of the glyphs needed for Unicode text. Together, the fonts contain approximately 7500 glyphs.

The main ClearlyU font has the XLFD

```
-mutt-clearlyu-medium-r-normal--17-120-100-100-p-101-iso10646-1
```

and resides in the font file

```
/usr/share/fonts/X11/misc/cu12.pcf.gz
```

Additional ClearlyU fonts include

```
-mutt-clearlyu alternate glyphs-medium-r-normal--17-120-100-100-p-91-iso10646-1
-mutt-clearlyu pua-medium-r-normal--17-120-100-100-p-111-iso10646-1
-mutt-clearlyu arabic extra-medium-r-normal--17-120-100-100-p-103-fontspecific-0
```

```
-mutt-clearlyu ligature-medium-r-normal--17-120-100-100-p-141-fontspecific-0
```

The *Alternate Glyphs* font contains additional glyph shapes that are needed for certain languages. A second alternate glyph font will be provided later for cases where a character has more than one commonly used alternate shape (*e.g.* the Urdu heh).

The *PUA* font contains extra glyphs that are useful for certain rendering purposes.

The *Arabic Extra* font contains the glyphs necessary for characters that don't have all of their possible shapes encoded in ISO 10646. The glyphs are roughly ordered according to the order of the characters in the ISO 10646 standard.

The *Ligature* font contains ligatures for various scripts that may be useful for improved presentation of text.

Standard scalable fonts

X11R7.6 includes all the scalable fonts distributed with X11R6.

Standard Type 1 fonts

The IBM Courier set of fonts cover ISO 8859-1 and ISO 8859-2 as well as Adobe Standard Encoding. These fonts have XLFD

```
-adobe-courier-medium-*-*--0-0-0-0-m-0-*-*
```

and reside in the font files

```
/usr/share/fonts/X11/Type1/cour*.pfa
```

The Adobe Utopia set of fonts only cover ISO 8859-1 as well as Adobe Standard Encoding. These fonts have XLFD

```
-adobe-utopia-*-*-normal--0-0-0-0-p-0-iso8859-1
```

and reside in the font files

```
/usr/share/fonts/X11/Type1/UT*.pfa
```

Finally, X11R7.6 also comes with Type 1 versions of Bitstream Courier and Charter. These fonts have XLFD

```
-bitstream-courier-*-*-normal--0-0-0-0-m-0-iso8859-1
-bitstream-charter-*-*-normal--0-0-0-0-p-0-iso8859-1
```

and reside in the font files

```
/usr/share/fonts/X11/Type1/c*bt_.pfb
```

The Bigelow & Holmes Luxi family

X11R7.6 includes the *Luxi* family of scalable fonts, in both TrueType and Type 1 format. This family consists of the fonts *Luxi Serif*, with XLFD

```
-b&h-luxi serif-medium-*-normal--*-*-*-*-p-*-*-*
```

Luxi Sans, with XLFD

```
-b&h-luxi sans-medium-*-normal--*-*-*-*-p-*-*-*
```

and *Luxi Mono*, with XLFD

```
-b&h-luxi mono-medium-*-normal--*-*-*-*-m-*-*-*
```

Each of these fonts comes Roman, oblique, bold and bold oblique variants The TrueType version have glyphs covering the basic ASCII Unicode range, the Latin 1 range, as well as the *Extended Latin* range and some additional punctuation characters. In particular, these fonts include all the glyphs needed for ISO 8859 parts 1, 2, 3, 4, 9, 13 and 15, as well as all the glyphs in the Adobe Standard encoding and the Windows 3.1 character set.

The glyph coverage of the Type 1 versions is somewhat reduced, and only covers ISO 8859 parts 1, 2 and 15 as well as the Adobe Standard encoding.

The Luxi fonts are original designs by Kris Holmes and Charles Bigelow. Luxi fonts include seriffed, sans serif, and monospaced styles, in roman and oblique, and normal and bold weights. The fonts share stem weight, x-height, capital height, ascent and descent, for graphical harmony.

The character width metrics of Luxi roman and bold fonts match those of core fonts bundled with popular operating and window systems.

The license terms for the Luxi fonts are included in the file `COPYRIGHT.BH', as well as in the License document [../LICENSE.html].

Charles Bigelow and Kris Holmes from Bigelow and Holmes Inc. developed the Luxi typeface designs in Ikarus digital format.

URW++ Design and Development GmbH converted the Ikarus format fonts to TrueType and Type1 font programs and implemented the grid-fitting "hints" and kerning tables in the Luxi fonts.

For more information, please contact `<design@bigelowandholmes.com>` or `<info@urwpp.de>`, or consult the URW++ web site [http://www.urwpp.de].

An earlier version of the Luxi fonts was made available under the name *Lucidux*. This name should no longer be used due to trademark uncertainties, and all traces of the *Lucidux* name have been removed from X11R7.6.

More about core fonts

This section describes X11R7.6-specific enhancements to the core X11 fonts system.

Core fonts and internationalisation

The scalable font backends (Type 1 and TrueType) can automatically re-encode fonts to the encoding specified in the XLFD in ``fonts.dir'`. For example, a ``fonts.dir'` file can contain entries for the Type 1 Courier font such as

```
cour.pfa -adobe-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
cour.pfa -adobe-courier-medium-r-normal--0-0-0-0-m-0-iso8859-2
```

which will lead to the font being recoded to ISO 8859-1 and ISO 8859-2 respectively.

The *fontenc* layer

Two of the scalable backends (Type 1 and the *FreeType* TrueType backend) use a common *fontenc* layer for font re-encoding. This allows these backends to share their encoding data, and allows simple configuration of new locales independently of font type.

Please note: the X-TrueType (X-TT) backend is not included in X11R7.6. That functionality has been merged into the FreeType backend.>

In the *fontenc* layer, an encoding is defined by a name (such as `iso8859-1`), possibly a number of aliases (alternate names), and an ordered collection of mappings. A mapping defines the way the encoding can be mapped into one of the *target encodings* known to *fontenc*; currently, these consist of Unicode, Adobe glyph names, and arbitrary TrueType ``cmap''s.

A number of encodings are hardwired into *fontenc*, and are therefore always available; the hardcoded encodings cannot easily be redefined. These include:

- `iso10646-1`: Unicode;
- `iso8859-1`: ISO Latin-1 (Western Europe);
- `iso8859-2`: ISO Latin-2 (Eastern Europe);
- `iso8859-3`: ISO Latin-3 (Southern Europe);
- `iso8859-4`: ISO Latin-4 (Northern Europe);
- `iso8859-5`: ISO Cyrillic;
- `iso8859-6`: ISO Arabic;
- `iso8859-7`: ISO Greek;
- `iso8859-8`: ISO Hebrew;
- `iso8859-9`: ISO Latin-5 (Turkish);
- `iso8859-10`: ISO Latin-6 (Nordic);
- `iso8859-15`: ISO Latin-9, or Latin-0 (Revised Western-European);
- `koi8-r`: KOI8 Russian;
- `koi8-u`: KOI8 Ukrainian (see RFC 2319);

- `koi8-ru`: KOI8 Russian/Ukrainian;
- `koi8-uni`: KOI8 ``Unified'' (Russian, Ukrainian, and Byelorussian);
- `koi8-e`: KOI8 ``European,'' ISO-IR-111, or ECMA-Cyrillic;
- `microsoft-symbol` and `apple-roman`: these are only likely to be useful with TrueType symbol fonts.

Additional encodings can be added by defining *encoding files*. When a font encoding is requested that the *fontenc* layer doesn't know about, the backend checks the directory in which the font file resides (not necessarily the directory with `fonts.dir`!) for a file named \``encodings.dir`'. If found, this file is scanned for the requested encoding, and the relevant encoding definition file is read in. The \`**mkfontdir**' utility, when invoked with the \``-e`' option followed by the name of a directory containing encoding files, can be used to automatically build \``encodings.dir`' files. Please see the mkfontdir(1) [mkfontdir.1.html] manual page for more details.

A number of encoding files for common encodings are included with X11R7.6. Information on writing new encoding files can be found in Format of encodings directory files and Format of encoding files later in this document.

Backend-specific notes about fontenc

The *FreeType* backend

For TrueType and OpenType fonts, the FreeType backend scans the mappings in order. Mappings with a target of PostScript are ignored; mappings with a TrueType or Unicode target are checked against all the cmaps in the file. The first applicable mapping is used.

For Type 1 fonts, the FreeType backend first searches for a mapping with a target of PostScript. If one is found, it is used. Otherwise, the backend searches for a mapping with target Unicode, which is then composed with a built-in table mapping codes to glyph names. Note that this table only covers part of the Unicode code points that have been assigned names by Adobe.

Specifying an encoding value of `adobe-fontspecific` for a Type 1 font disables the encoding mechanism. This is useful with symbol and incorrectly encoded fonts (see Incorrectly encoded fonts below).

If a suitable mapping is not found, the FreeType backend defaults to ISO 8859-1.

Format of encoding directory files

In order to use a font in an encoding that the font backend does not know about, you need to have an \``encodings.dir`' file either in the same directory as the font file used or in a system-wide location (\``/usr/share/fonts/X11/encodings/`' by default).

The \``encodings.dir`' file has a similar format to \``fonts.dir`'. Its first line specifies the number of encodings, while every successive line has two columns, the name of the encoding, and the name of the encoding file; this can be relative to the current directory, or absolute. Every encoding name should agree with the encoding name defined in the encoding file. For example,

```
3
mulearabic-0 /usr/share/fonts/X11/encodings/mulearabic-0.enc
mulearabic-1 /usr/share/fonts/X11/encodings/mulearabic-1.enc
mulearabic-2 /usr/share/fonts/X11/encodings/mulearabic-2.enc
```

The name of an encoding *must* be specified in the encoding file's `` `STARTENCODING' `` or `` `ALIAS' `` line. It is not enough to create an `` `encodings.dir' `` entry.

If your platform supports it (it probably does), encoding files may be compressed or gzipped.

The `` `encoding.dir' `` files are best maintained by the `**mkfontdir**' utility. Please see the mkfontdir(1) [mkfontdir.1.html] manual page for more information.

Format of encoding files

The encoding files are ``free form,'' *i.e.* any string of whitespace is equivalent to a single space. Keywords are parsed in a non-case-sensitive manner, meaning that `` `size' ``, `` `SIZE' ``, and `` `SiZE' `` all parse as the same keyword; on the other hand, case is significant in glyph names.

Numbers can be written in decimal, as in `` `256' ``, in hexadecimal, as in `` `0x100' ``, or in octal, as in `` `0400' ``.

Comments are introduced by a hash sign `` `#' ``. A `` `#' `` may appear at any point in a line, and all characters following the `` `#' `` are ignored, up to the end of the line.

The encoding file starts with the definition of the name of the encoding, and possibly its alternate names (aliases):

```
STARTENCODING mulearabic-0
ALIAS arabic-0
```

The name of the encoding and its aliases should be suitable for use in an XLFD font name, and therefore contain exactly one dash `` `-' ``.

The encoding file may then optionally declare the size of the encoding. For a linear encoding (such as ISO 8859-1), the SIZE line specifies the maximum code plus one:

```
SIZE 0x2B
```

For a matrix encoding, it should specify two numbers. The first is the number of the last row plus one, the other, the highest column number plus one. In the case of `` `jisx0208.1990-0' `` (JIS X 0208(1990), double-byte encoding, high bit clear), it should be

```
SIZE 0x75 0x80
```

In the case of a matrix encoding, a `` `FIRSTINDEX' `` line may be included to specify the minimum glyph index in an encoding. The keyword `` `FIRSTINDEX' `` is followed by two integers, the minimum row number followed by the minimum column number:

```
FIRSTINDEX 0x20 0x20
```

In the case of a linear encoding, a `` `FIRSTINDEX' `` line is not very useful. If for some reason however you chose to include on, it should be followed by a single integer.

Note that in most font backends inclusion of a `` `FIRSTINDEX' `` line has the side effect of disabling default glyph generation, and this keyword should therefore be avoided unless absolutely necessary.

Codes outside the region defined by the `` `SIZE' `` and `` `FIRSTINDEX' `` lines are understood to be undefined. Encodings default to linear encoding with a size of 256 (0x100). This means that you must declare the size of all 16 bit encodings.

What follows is one or more mapping sections. A mapping section starts with a `` `STARTMAPPING' `` line stating the target of the mapping. The target may be one of:

- Unicode (ISO 10646):

  ```
  STARTMAPPING unicode
  ```

- a given TrueType ``cmap'':

  ```
  STARTMAPPING cmap 3 1
  ```

- PostScript glyph names:

  ```
  STARTMAPPING postscript
  ```

Every line in a mapping section maps one from the encoding being defined to the target of the mapping. In mappings with a Unicode or TrueType mapping, codes are mapped to codes:

```
0x21 0x0660
0x22 0x0661
...
```

As an abbreviation, it is possible to map a contiguous range of codes in a single line. A line consisting of three integers

```
<it/start/ <it/end/ <it/target/
```

is an abbreviation for the range of lines

start	*target*
start+1	*target*+1

```
...
```

```
end        target+end-start
```

For example, the line

```
0x2121 0x215F 0x8140
```

is an abbreviation for

```
0x2121 0x8140
0x2122 0x8141
...
0x215F 0x817E
```

Codes not listed are assumed to map through the identity (*i.e.* to the same numerical value). In order to override this default mapping, you may specify a range of codes to be undefined by using an `` `UNDEFINE' `` line:

```
UNDEFINE 0x00 0x2A
```

or, for a single code,

```
UNDEFINE 0x1234
```

PostScript mappings are different. Every line in a PostScript mapping maps a code to a glyph name

```
0x41 A
0x42 B
...
```

and codes not explicitly listed are undefined.

A mapping section ends with an `ENDMAPPING` line

```
ENDMAPPING
```

After all the mappings have been defined, the file ends with an `ENDENCODING` line

```
ENDENCODING
```

In order to make future extensions to the format possible, lines starting with an unknown keyword are silently ignored, as are mapping sections with an unknown target.

Using symbol fonts

Type 1 symbol fonts should be installed using the `adobe-fontspecific` encoding.

In an ideal world, all TrueType symbol fonts would be installed using one of the `microsoft-symbol` and `apple-roman` encodings. A number of symbol fonts, however, are not marked as such; such fonts should be installed using `microsoft-cp1252`, or, for older fonts, `microsoft-win3.1`.

In order to guarantee consistent results (especially between Type 1 and TrueType versions of the same font), it is possible to define a special encoding for a given font. This has already been done for the `ZapfDingbats` font; see the file `` `encodings/adobe-dingbats.enc `` '.

Hints about using badly encoded fonts

A number of text fonts are incorrectly encoded. Incorrect encoding is sometimes done by design, in order to make a font for an exotic script appear like an ordinary Western text font on systems which are not easily extended with new locale data. It is often the result of the font designer's laziness or incompetence; for some reason, most people seem to find it easier to invent idiosyncratic glyph names rather than follow the Adobe glyph list.

There are two ways of dealing with such fonts: using them with the encoding they were designed for, and creating an *ad hoc* encoding file.

Using fonts with the designer's encoding

In the case of Type 1 fonts, the font designer can specify a default encoding; this encoding is requested by using the `` `adobe-fontspecific ``' encoding in the XLFD name. Sometimes, the font designer omitted to specify a reasonable default encoding, in which case you should experiment with `` `adobe-standard ``', `` `iso8859-1 ``', `` `microsoft-cp1252 ``', and `` `microsoft-win3.1 ``'. (The encoding `` `microsoft-symbol ``' doesn't make sense for Type 1 fonts).

TrueType fonts do not have a default encoding. However, most TrueType fonts are designed with either Microsoft or Apple platforms in mind, so one of `` `microsoft-symbol ``', `` `microsoft-cp1252 ``', `` `microsoft-win3.1 ``', or `` `apple-roman ``' should yield reasonable results.

Specifying an *ad hoc* encoding file

It is always possible to define an encoding file to put the glyphs in a font in any desired order. Again, see the `` `encodings/adobe-dingbats.enc ``' file to see how this is done.

Specifying font aliases

By following the directions above, you will find yourself with a number of fonts with unusual names --- with encodings such as `` `adobe-fontspecific ``', `` `microsoft-win3.1 ``' *etc*. In order to use these fonts with standard applications, it may be useful to remap them to their proper names.

This is done by writing a `` `fonts.alias ``' file. The format of this file is very simple: it consists of a series of lines each mapping an alias name to a font name. A `` `fonts.alias ``' file might look as follows:

```
"-ogonki-alamakota-medium-r-normal--0-0-0-0-p-0-iso8859-2" \
```

```
  "-ogonki-alamakota-medium-r-normal--0-0-0-0-p-0-adobe-fontspecific"
```

(both XLFD names on a single line). The syntax of the `fonts.alias' file is more precisely described in the mkfontdir(1) [mkfontdir.1.html] manual page.

Additional notes about scalable core fonts

About the *FreeType* backend

The *FreeType* backend (formerly *xfsft*) is a backend based on version 2 of the FreeType library (see the FreeType web site [http://www.freetype.org/]) and has the X-TT functionalities for CJKV support provided by the After X-TT Project (see the After X-TT Project web site [http://x-tt.sourceforge.jp/]). The *FreeType* backend has support for the ``fontenc'' style of internationalisation (see the section called "The *fontenc* layer "). This backend supports TrueType font files (`*.ttf'), OpenType font files (`*.otf'), TrueType Collections (`*.ttc'), OpenType Collections (`*.otc') and Type 1 font files (`*.pfa' and `*.pfb').

In order to access the faces in a TrueType Collection file, the face number must be specified in the fonts.dir file before the filename, within a pair of colons, or by setting the 'fn' TTCap option. For example,

```
:1:mincho.ttc -misc-pmincho-medium-r-normal--0-0-0-0-p-0-jisx0208.1990-0
```

refers to face 1 in the `mincho.ttc' TrueType Collection file.

The new *FreeType* backend supports the extended `fonts.dir' syntax introduced by X-TrueType with a number of options, collectively known as `TTCap'. A `TTCap' entry follows the general syntax

```
option=value:
```

and should be specified before the filename. The new *FreeType* almost perfectly supports TTCap options that are compatible with X-TT 1.4. The Automatic Italic (`ai'), Double Strike (`ds') and Bounding box Width (`bw') options are indispensable in CJKV. For example,

```
mincho.ttc -misc-mincho-medium-r-normal--0-0-0-0-c-0-jisx0208.1990-0
ds=y:mincho.ttc -misc-mincho-bold-r-normal--0-0-0-0-c-0-jisx0208.1990-0
ai=0.2:mincho.ttc -misc-mincho-medium-i-normal--0-0-0-0-c-0-jisx0208.1990-0
ds=y:ai=0.2:mincho.ttc -misc-mincho-bold-i-normal--0-0-0-0-c-0-jisx0208.1990-0
bw=0.5:mincho.ttc -misc-mincho-medium-r-normal--0-0-0-0-c-0-jisx0201.1976-0
bw=0.5:ds=y:mincho.ttc -misc-mincho-bold-r-normal--0-0-0-0-c-0-jisx0201.1976-0
bw=0.5:ai=0.2:mincho.ttc -misc-mincho-medium-i-normal--0-0-0-0-c-0-jisx0201.1976-0
bw=0.5:ds=y:ai=0.2:mincho.ttc -misc-mincho-bold-i-normal--0-0-0-0-c-0-jisx0201.197
```

setup the complete combination of jisx0208 and jisx0201 using mincho.ttc only. More information on the TTCap syntax is found on the After X-TT Project page [http://x-tt.sourceforge.jp/].

The *FreeType* backend uses the *fontenc* layer in order to support recoding of fonts; this was described in the section called "The *fontenc* layer " and especially the section called "The *FreeType* backend " earlier in this document.

Delayed glyph rasterisation

When loading a proportional fonts which contain a huge number of glyphs, the old *FreeType* delayed glyph rasterisation until the time at which the glyph was first used. The new FreeType (libfreetype-xtt2) has an improved `very lazy' metric calculation method to speed up the process when loading TrueType or OpenType fonts. Although the *X-TT* module also has this method, the "`vl=y`" TTCap option must be set if you want to use it. This is the default method for *FreeType* when it loads multi-byte fonts. Even if you use a unicode font which has tens of thousands of glyphs, this delay will not be worrisome as long as you use the new *FreeType* backend -- its `very lazy' method is super-fast.

The maximum error of bitmap position using `very lazy' method is 1 pixel, and is the same as that of a character-cell spacing. When the X-TT backend is used with the ``vl=y`' option, a chipped bitmap is displayed with certain fonts. However, the new FreeType backend has minimal problem with this, since it corrects left- and right-side bearings using `italicAngle' in the TrueType/OpenType post table, and does automatic correction of bitmap positions when rasterisation so that chipped bitmaps are not displayed. Nevertheless if you don't want to use the `very lazy' method when using multi-bytes fonts, set ``vl=n`' in the TTCap option to disable it:

```
vl=n:luxirr.ttf -b&h-Luxi Serif-medium-r-normal--0-0-0-0-p-0-iso10646-1
```

Of course, both backends also support an optimisation for character-cell fonts (fonts with all glyph metrics equal, or terminal fonts). A font with an XLFD specifying a character-cell spacing ``c`', as in

```
-misc-mincho-medium-r-normal--0-0-0-0-c-0-jisx0208.1990-0
```

or

```
fs=c:mincho.ttc -misc-mincho-medium-r-normal--0-0-0-0-p-0-jisx0208.1990-0
```

will not compute the metric for each glyph, but instead trust the font to be a character-cell font. You are encouraged to make use of this optimisation when useful, but be warned that not all monospaced fonts are character-cell fonts.

Appendix: background and terminology

Characters and glyphs

A computer text-processing system inputs keystrokes and outputs *glyphs*, small pictures that are assembled on paper or on a computer screen. Keystrokes and glyphs do not, in general, coincide: for example, if the system does generate ligatures, then to the sequence of two keystrokes `<f><i>` will typically correspond a single glyph. Similarly, if the system shapes Arabic glyphs in a vaguely reasonable manner, then multiple different glyphs may correspond to a single keystroke.

The complex transformation rules from keystrokes to glyphs are usually factored into two simpler transformations, from keystrokes to *characters* and from characters to glyphs. You may want to think of characters as the basic unit of text that is stored *e.g.* in the buffer of your text editor. While the definition of a character is intrinsically application-specific, a number of standardised collections of characters have been defined.

A *coded character set* is a set of characters together with a mapping from integer codes --- known as *codepoints* --- to characters. Examples of coded character sets include US-ASCII, ISO 8859-1, KOI8-R, and JIS X 0208(1990).

A coded character set need not use 8 bit integers to index characters. Many early systems used 6 bit character sets, while 16 bit (or more) character sets are necessary for ideographic writing systems.

Font files, fonts, and XLFD

Traditionally, typographers speak about *typefaces* and *founts*. A typeface is a particular style or design, such as Times Italic, while a fount is a molten-lead incarnation of a given typeface at a given size.

Digital fonts come in *font files*. A font file contains the information necessary for generating glyphs of a given typeface, and applications using font files may access glyph information in an arbitrary order.

Digital fonts may consist of bitmap data, in which case they are said to be *bitmap fonts*. They may also consist of a mathematical description of glyph shapes, in which case they are said to be *scalable fonts*. Common formats for scalable font files are *Type 1* (sometimes incorrectly called *ATM fonts* or *PostScript fonts*), *TrueType* and *OpenType*.

The glyph data in a digital font needs to be indexed somehow. How this is done depends on the font file format. In the case of Type 1 fonts, glyphs are identified by *glyph names*. In the case of TrueType fonts, glyphs are indexed by integers corresponding to one of a number of indexing schemes (usually Unicode --- see below).

The X11 core fonts system uses the data in a font file to generate *font instances*, which are collections of glyphs at a given size indexed according to a given encoding.

X11 core font instances are usually specified using a notation known as the *X Logical Font Description* (XLFD). An XLFD starts with a dash \``-`', and consists of fourteen fields separated by dashes, for example:

```
-adobe-courier-medium-r-normal--12-120-75-75-m-70-iso8859-1
```

Or particular interest are the last two fields \``iso8859-1`', which specify the font instance's encoding.

A scalable font is specified by an XLFD which contains zeroes instead of some fields:

```
-adobe-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
```

X11 font instances may also be specified by short name. Unlike an XLFD, a short name has no structure and is simply a conventional name for a font instance. Two short names are of particular interest, as the server will not start if font instances with these names cannot be opened. These are \``fixed`', which specifies the fallback font to use when the requested font cannot be opened, and `` `cursor' ``, which specifies the set of glyphs to be used by the mouse pointer.

Short names are usually implemented as aliases to XLFDs; the standard `` `fixed' `` and `` `cursor' `` aliases are defined in

```
/usr/share/font/X11/misc/fonts.alias
```

Unicode

Unicode (http://www.unicode.org) is a coded character set with the goal of uniquely identifying all characters for all scripts, current and historical. While Unicode was explicitly not designed as a glyph encoding scheme, it is often possible to use it as such.

Unicode is an *open* character set, meaning that codepoint assignments may be added to Unicode at any time (once specified, though, an assignment can never be changed). For this reason, a Unicode font will be *sparse*, meaning that it only defines glyphs for a subset of the character registry of Unicode.

The Unicode standard is defined in parallel with the international standard ISO 10646. Assignments in the two standards are always equivalent, and we often use the terms *Unicode* and *ISO 10646* interchangeably.

When used in the X11 core fonts system, Unicode-encoded fonts should have the last two fields of their XLFD set to `` `iso10646-1' ``.

References

X11R7.6 comes with extensive documentation in the form of manual pages and typeset documents. Before installing fonts, you really should read the fontconfig(3) [fontconfig.3.html] and mkfontdir(1) [mkfontdir.1.html] manual pages; other manual pages of interest include X(7) [X.7.html], Xserver(1) [Xserver.1.html], xset(1) [xset.1.html], Xft(3) [Xft.3.html], xlsfonts(1) [xlsfonts.1.html] and showfont(1) [showfont.1.html]. In addition, you may want to read the X Logical Font Description document, by Jim Flowers, which is provided in the file `` `xc/doc/xlfd.PS.Z' ``.

The comp.fonts FAQ [http://www.faqs.org/faqs/by-newsgroup/comp/comp.fonts.html], which is unfortunately no longer being maintained, contains a wealth of information about digital fonts.

Xft and Fontconfig are described on Keith Packard's Fontconfig site [http://www.fontconfig.org].

The xfsft home page [http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/] has been superseded by this document, and is now obsolete; you may however still find some of the information that it contains useful. Joerg Pommnitz' xfsft page [http://www.joerg-pommnitz.de/TrueType/xfsft.html] is the canonical source for the `**ttmkfdir**' utility, which is the ancestor of **mkfontscale**.

The author's software pages [http://www.pps.jussieu.fr/~jch/software/] might or might not contain related scribbles and development versions of software.

The documentation of *X-TrueType* is available from the After X-TT Project page [http://x-tt.sourceforge.jp/].

While the Unicode consortium site [http://www.unicode.org] may be of interest, you are more likely to find what you need in Markus Kuhn's UTF-8 and Unicode FAQ [http://www.cl.cam.ac.uk/~mgk25/unicode.html].

The IANA RFC documents, available from a number of sites throughout the world, often provide interesting information about character set issues; see for example RFC 373.