9 Configuring an X client

9.1 Resources

X clients are configured and controlled via a set of parameters called "resources".

Example:
The resource that controls the colour of the background of the `quit` button of `xedit` is `Xedit.paned.buttons.quit.background`.

All but the final component of the resource name are either the class or instance name of the widget involved. (Instance names take priority over class names in case of clash.) The final option is the name of the property being controlled.

The author of the client should define the instance names for all of the widgets except the shell widget at top level and set the fallback values for the resources.

These resource names can clearly become unwieldy so there is a wildcard facility. The resource `Xedit.*.background` will control the colour the background of *every* widget under `xedit` that has a `background` property and which does not have a more specific resource governing it.

Example:
If the resource `Xedit.*.background` has the value `LightBlue` and the resource `Xedit.*.editWindow.background` has the value `White`, all the backgrounds except the editing part's will be light blue.

9.2 Setting resource values

Typically, an application can pick up its resources from a variety of places.

On launch, `xedit` searches the following places for its resources, using the first match it comes across for its internal "working" database:

- The command line

  e.g. `$ xedit -xrm 'Xedit.*.background: Red'`

  All `X` clients should support the `-xrm` option.

- User resource file

  `${XENVIRONMENT}`, defaulting to `.Xdefaults`

- Resources from the server

  Put there by `xrdb` (See 9.3.)

- Display specific user resource file

  `.Xdefaults-`*display*

- Application specific user resource file

  `${XUSERFILESEARCHPATH}`, defaults to `~/%N`

  `%N` is replaced with the class name: `~/XEdit`

- Application specific system resource files

  `/opt/X11R5/lib/X11/app-defaults/%N`

- Hard-wired fallback resources in the application

  The author's choice

This amount of choice inevitably ends in tears. We recommend using `xrdb` and setting resources in the server.

9.3 Using `xrdb`

9.3.1 Using `xrdb` to query the server for resources

The `-query` option will extract all the resources stored in the server and display them, one per line.

```
$ xrdb -query
XTerm*reverseWrap: on
XTerm*scrollBar:   on
XTerm*saveLines:   1000
XCalc*iconic:      true
```

9.3.2 Using `xrdb` to add resources to the server

The `-merge` option will add to resources the server's set and will overwrite those that are updated by the `xrdb` call. Other resources will not be affected.

```
$ xrdb -merge
demo*background: LightBlue
^D

$ xrdb -query
XTerm*reverseWrap: on
XTerm*scrollBar:   on
XTerm*saveLines:   1000
XCalc*iconic:      true
demo*background:   LightBlue

$ xrdb -merge
demo*background: PaleGreen
^D

$ xrdb -query
XTerm*reverseWrap: on
XTerm*scrollBar:   on
XTerm*saveLines:   1000
XCalc*iconic:      true
demo*background:   PaleGreen

$
```

`xrdb`'s `-merge` option can be passed a file name to get its information from (`$ xrdb -merge` *filename*). It will read the standard input if it is not given a filename, as in the example above.

9.3.3 Using `xrdb` to set the server's resources

The `-load` option will remove all the server's current resources and replace them with the resources passed to the `xrdb` command.

```
$ xrdb -query
XTerm*reverseWrap: on
XTerm*scrollBar:   on
XTerm*saveLines:   1000
XCalc*iconic:      true
demo*background:   PaleGreen

$ xrdb -load
XTerm*scrollBar: off

$ xrdb -query
XTerm*scrollBar:   off

$
```

As with `-merge`, `xrdb -load` will take a filename or read the standard input.

9.3.4 `xrdb` at start-up

In the X start-up scripts, most systems check for a file called `.Xresources` in the user's home directory and, if it exists, apply `xrdb` to it. See section 7.3 for details of the context. A line similar to

```
[ -r $HOME/.Xresources ] && xrdb -load $HOME/.Xresources
```

is not uncommon.

`xrdb` knows about a few fixed properties of the server: its dimensions, whether it is colour or black & white, etc. It can use these properties in conjunction with the styandard `C` macro processor, `cpp`, to permit some simple logic in the `.Xresources` file. Consider, as an example, the following lines for a `.Xresources` file:

```
#if HEIGHT < 1000
xrn.height:   750
xedit.height: 500
#else
xrn.height:  1000
xedit.height: 750
#endif
```

which would set the resources `xedit.height` and `xrn.height` to different values depending on the height of the display measured in pixels.

The string "`HEIGHT`" in the example is called a "symbol". To see the complete set of symbols together with their values use the command `xrdb -symbols`. A list of the more useful symbols follows:

`HEIGHT` gives the height of the screen in pixels.

`WIDTH` gives the width of the screen in pixels.

`X_RESOLUTION` gives the screen's horizontal resolution in pixels per metre.

`Y_RESOLUTION` gives the screen's vertical resolution in pixels per metre.

`COLOR` is only defined if the display supports colour.

An example of testing for whether `COLOR` is defined is given here:

```
#ifdef COLOR
Xrn.*.background:   LightBlue
Xedit.*.background: Wheat
#else
Xrn.*.background:   white
Xedit.*.background: white
#endif
```

9.4 X client geometry

While the shape and position of a client's widget can be controlled by resources in the above manners, because the control of the client's "geometry" is such a commonly required facility special provision is made.

All X clients should support the `-geometry` option. Some typical examples of its use are:

```
xbiff -geometry -120+0
xclock -geometry 112x112-0+0
```

The geometry specification comes in two halves: $\pm x \pm y$ and XxY.

9.4.1 Position

The first half specifies the position of the client's window on the display:

- x and y are specified in pixels.
- x is measured from left to right.
- y is measured from top to bottom.
- The exact nature of the positioning depends on the two plus or minus signs:

  +x+y specifies the position of the top left corner of the window with respect to the top left corner of the display.

  +x-y specifies the position of the bottom left corner of the window with respect to the bottom left corner of the display.

  -x+y specifies the position of the top right corner of the window with respect to the top right corner of the display.

  -x-y specifies the position of the bottom right corner of the window with respect to the bottom right corner of the display.

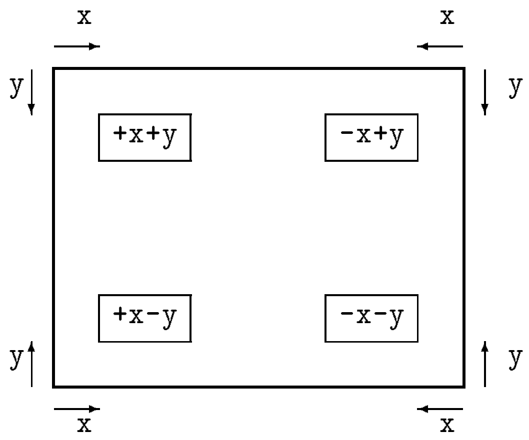

Figure 9: Effects of sign in the geometry specification

9.4.2 Size

The second half of the geometry specification, *x*x*y*, controls the size of the window:

- *x* and *y* are *usually* measured in pixels.
- `xterm` measures its size in characters.
- This is the size *excluding* the window manager decorations.

9.4.3 Geometry resources

If you want all `X` clients of a certain sort to have the same gemoetry then you can set a `geometry` resource giving the default. This resource will override the `height` and `width` resources.

```
$ xrdb -merge
Xedit.geometry: 600x700
Xedit.height:   500
Xedit.width:    500
```

```
^D
$ xedit &
```

This sequence of instructions causes `xedit` to have width 600 pixels and height 700 pixels. The order in which the resources were entered into `xrdb` is irrelevant.

9.5 Colours

Colours are typically referred to by name. Each name is internally mapped to a triplet of numbers defining the red, green and blue intensities. On each system there is often a file called `rgb.txt` containing the allocation of intensities to a particular name. Some sample lines from `rgb.txt` are given in figure 10:

There is no guarantee that the same collection of colours will be available on all machines. For example the file `rgb.txt` contains 738 colours in MIT's X11R5 release, 883 colours under Irix 5.1, and 752 colours under Motif 1.2. Nor is there any guarantee that two colour names on different servers will have the same intensity triuplets assigned to them. However, this problem occurs less often.

If a colour is used on a grayscale monitor only one of the intesities (typcally the red intensity) is used. An important set of colours are the grays. `gray0` is identical to `black` and has uniformaly zero intensity. `gray100` is identical to `white` and has uniformly maximum intensity. The gray levels are designed to give 100 different shades of gray.

9.6 Fonts

Fonts can be referreed to in a variety of ways. Each font has a name and a number of aliases. It is not profitable to distinguish between names and aliases in this discussion so henceforth we will refer only to "font names". These names come in two forms:

```
255 250 250             snow
248 248 255             GhostWhite
245 245 245             WhiteSmoke
220 220 220             gainsboro
255 250 240             FloralWhite
253 245 230             OldLace
250 240 230             linen
250 235 215             AntiqueWhite
255 239 213             PapayaWhip
      ...
  0 100   0             DarkGreen
 85 107  47             DarkOliveGreen
143 188 143             DarkSeaGreen
 46 139  87             SeaGreen
 60 179 113             MediumSeaGreen
 32 178 170             LightSeaGreen
152 251 152             PaleGreen
      ...
240 255 255             azure
240 248 255             AliceBlue
230 230 250             lavender
255 240 245             LavenderBlush
255 228 225             MistyRose
  0   0   0             black
  0   0   0             gray0
  3   3   3             gray1
              ...
252 252 252             gray99
255 255 255             gray100
255 255 255             white
```

Figure 10: Some lines from `rgb.txt`

"simple names" are just that, short names which refer to a font but do not necessarily carry much information. Examples are:

`7x13`: a fixed width normal face font that is 7 pixels wide by 13 pixels high used for such things as `xterm`'s display

`7x13bold`: the same but bold face

`avantgarde-book`: font that can be scaled to various sizes.

"X Logical Font Descriptors" (XLFDs) are (long) descriptive names which are hyphen-delimited specifications of properties. Examples are:

`-misc-fixed-medium-r-normal--13-100-100-100-c-70-iso8859-1`: a fixed width normal face font that is 13 pixels high and 7 pixels wide on average (and therefore on every character in a fixed width font). the same as the `7x13` mentioned before.

`-misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1`: the same as the `7x13bold` font.

`-itc-avantgarde-book-r-normal--0-0-0-0-p-0-iso8859-1`: the same as the `avantgarde-book` font.

A complete list of all the fonts available on a server is given by the `xlsfonts` command.

While the short forms may seem to be more convenient at first glance, the XLFDs are preferred due to a useful wildcarding facility that will be described in section 9.6.2 below. First we need to understand the structure of an XLFD.

9.6.1 X Logical Font Descriptor Structure

An XLFD begins with a hyphen. *Any* font name that begins with a hyphen will be treated as an XLFD. Each of the subsequent 14 hyphen-delimited fields are listed below:

Foundry: This specifies the company that *digitized* the font (as opposed to the company that created it in the first place).

Examples: `adobe`, `linotype`, `sun`.

Family: This identifies the family of designs that are all variations on a theme.

Examples: `new century schoolbook`, `times`, `palatino`.

Weight: This describes how "heavy" or "black" the font is within its family.

Examples: `light`, `normal`, `bold`.

Slant: This is a short code for the tilting nature of the character.

The set of allowable values is:

`R`: roman (normal)

`O`: oblique (clockwise tilt)

`I`: italic (different style and the tilt)

`RO`: reverse oblique (anticlockwise tilt)

`RI`: reverse italic (italic shape and anticlockwise tilt)

`OT`: other (font specific)

Set width: This is a string that describes the width of the characters in this font, relative to the family of fonts.

Examples are: `normal`, `narrow`, `double wide`.

Additional style: This is an "everything else not covered already" field for the style of the font (as opposed to its sizes).

Examples: `serif`, `sans serif`, `decorated`.

Pixel size: This is a number gives the vertical size of the font in pixels. A value of `0` means that the font is scalable and is available at any size.

Point size: This number gives the size in points that the font was designed to be used at. (There are 72.27 points per inch, or roughly 28.68 points per centimetre).

X resolution: This gives the number of dots per inch in the x-direction of the display the font was designed for.

Y resolution: This gives the number of dots per inch in the y-direction of the display the font was designed for.

Spacing: This defines the nature of the spacing of the characters in the font. (NB: characters themselves have different widths. `i` is much narrower that `M`, for example.)

The allowed values are:

- `P`: proportional spacing. The width of the character as used for spacing varies with the character. Useful for printed documents, no good for a `VT100`-style 80-by-24 grid of characters.
- `M`: monospaced. The widths of all the characters are the same.
- `C`: character cell. This is a monospaced font with the additional restriction that the size of each character in pixels defines a box (the "cell") outside of which no pixels are set by this character. This is the font used by a `VT100` emulator for its characters.

Average width: This is the average (unweighted arithmetic mean) width of all the defined characters in the font given in tenths of a pixel. In a fixed width font, (monospace or charcter cell) this is the same as the individual width of any character. `0` is used to indicate a scalable font.

Registry: This is a technical descrioption of whose font-encoding technique has been used to store the font.

Examples: `adobe`, `dec`, `iso8859`.

Encoding: This says which of the registry's methods has been used.

Examples:

`adobe` only permits the method it calls "`font specific`".

`dec` only permits the method it calls "`dectech`".

`iso8859` has two methods called "`1`" and "`8`".

9.6.2 Wildcarding in XLFDs

When requesting a font from `X` it is possible to give a list of requirements for the font, based on the structure above but leaving it freedom of choice in other regards. For example, suppose I want a font that is "times roman", of normal weight (as opposed to bold, etc) and 14 pixels high. I can specify an XLFD as "`-*-times-medium-r-*-*-14-*-*-*-*-*-*-*`". The asterisk means that I don't care what values these parameters take. There are 7 font names matching this description on my machine and the X server will select one for me.

There are two ways to see matching fonts: `xfontsel` and `xlsfonts`. `xlsfonts` takes an XLFD with or without wildcards and list all the matches:

```
$ xlsfonts -fn '-*-times-medium-r-*-*-14-*-*-*-*-*-*-*'
-adobe-times-medium-r-normal--14-0-100-100-p-0-iso8859-1
-adobe-times-medium-r-normal--14-0-75-75-p-0-iso8859-1
-adobe-times-medium-r-normal--14-100-100-100-p-74-iso8859-1
-adobe-times-medium-r-normal--14-140-75-75-p-74-iso8859-1
-linotype-times-medium-r-normal--14-0-0-0-p-0-iso8859-1
-linotype-times-medium-r-normal--14-0-72-72-p-0-iso8859-1
-linotype-times-medium-r-normal--14-140-72-72-p-73-iso8859-1
```

`xfontsel` permits interactive searching of the available fonts. Pressing the

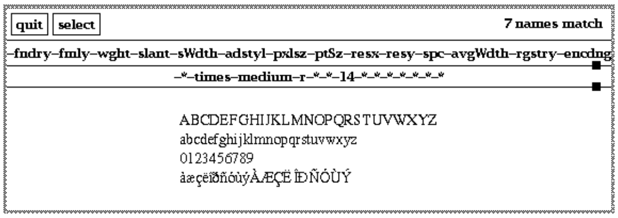

Figure 11: The `xfontsel` client

left button over the fields in the second row (`-fndry-fmly-`...) gives a menu of the various options, with unavailable ones being greyed out. An example of a matching font's contents is given in the bottom window. Clicking the `select` button buts the current XFLD in the server's cut buffer, so it can be easily pasted into a document.

The purpose of this wild-carding facility was so that people could give fairly liberal (many asterisks) fonts descriptions and have the same XLFD work on all servers. There still aren't enough common fonts to make this work properly, but it does give a simple way to search for available fonts that are roughly what you want.

If you ask for an XLFD that is not available, then the client will fall back to its own defaults:

```
$ xterm -fn '-*-*-bold-r-*--*-*-*-*-c-*-*-8'
xterm:  unable to open font \
"-*-*-bold-r-*--*-*-*-*-c-*-*-8",\
trying "fixed"....
```

9.7 Using instance names to select resources

Suppose I want to configure my `xterm`s and `xedit`s.

I typically connect to the C.U.S. machines and I want those xterms to be light blue. I also connect to various machines in my office and for these I would like a pale yellow background. The final set of machines I login to are the mail machines for which I want a pale green background. Finally, I would like my `xedit`s to have the same machine-dependent colour schemes.

Furthermore, I want the C.U.S. `xterm`s to be deeper than usual (32 lines rather than the default 24) and the mail `xterm`s to be 132 columns wide rather than 80.

I want all my `xterm`s to have scroll bars with 1000 lines of scrollback stored in each.

Finally, I want the `quit` buttons in all the `xedit`s to be red.

However, I might not always be using a colour `X` display. On a black & white display I want white backgrounds for my `xterm`s (the default colour) and white text on a black background for `xedit`'s `quit` button.

A common way to achieve this is to give each of my `xterm`s the instance names `xterm-`*machine* and each of my `xedit`s the instance names `xedit-`*machine*.

My `.Xresources` file would contain the lines

```
#ifdef COLOR
```

```
xterm-cus*background: LightBlue
xedit-cus*background: LightBlue
xterm-csi*background: LightYellow
xedit-csi*background: LightYellow
xterm-mail*background: PaleGreen
xedit-mail*background: PaleGreen
Xedit.paned.buttons.quit.background: red
#else
Xedit.paned.buttons.quit.background: black
Xedit.paned.buttons.quit.foreground: white
#endif
xterm-cus.vt100.geometry: 80x32
xterm-mail.vt100.geometry: 132x24
```

Then I can start my C.U.S. `xterm`s and `xedit`s with the commands

```
$ xterm -name xterm-cus &
$ xedit -name xedit-cus &
```

the office clients ("`csi`" is "Computing Service Internal", by the way) with

```
$ xterm -name xterm-csi &
$ xedit -name xedit-csi &
```

and the mail clients with

```
$ xterm -name xterm-mail &
$ xterm -name xedit-mail &
```