NAME

X11 – a portable, network-transparent window system

SYNOPSIS

There is no X11 command per se. This manual page is adapted from the X manual page supplied with the MIT sample server and is included with OpenWindows for reference.

The X Window System is a network transparent window system developed at MIT which runs on a wide range of computing and graphics machines.

The X Consortium requests that the following names be used when referring to this software:

X
X Window System
X Version 11
X Window System, Version 11
X11

DESCRIPTION

X Window System servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well.

X supports overlapping hierarchical subwindows and text and graphics operations, on both monochrome and color displays. For a full explanation of the functions that are available, see the **Xlib - C Language X Interface** manual, the *X Window System Protocol* specification, the *X Toolkit Intrinsics - C Language Interface* manual, and various toolkit documents.

Many utilities, window managers, games, toolkits, etc. are available from the user-contributed software. See your site administrator for details.

STARTING UP

See **openwin**(1) for information on starting the server and an initial set of client applications.

DISPLAY NAMES

From the user's prospective, every X server has a *display name* of the form:

hostname:displaynumber.screennumber

This information is used by the application to determine how it should connect to the server and which screen it should use by default (on displays with multiple monitors):

hostname
: The *hostname* specifies the name of the machine to which the display is physically connected. If the hostname is not given, the most efficient way of communicating to a server on the same machine will be used.

displaynumber

The phrase "display" is usually used to refer to collection of monitors that share a common keyboard and pointer (mouse, tablet, etc.). Most workstations tend to only have one keyboard, and therefore, only one display. Larger, multi-user systems, however, will frequently have several displays so that more than one person can be doing graphics work at once. To avoid confusion, each display on a machine is assigned a *display number* (beginning at 0) when the X server for that display is started. The display number must always be given in a display name.

screennumber

Some displays share a single keyboard and pointer among two or more monitors. Since each monitor has its own set of windows, each screen is assigned a *screen number* (beginning at 0) when the X server for that display is started. If the screen number is not given, then screen 0 will be used.

On POSIX systems, the default display name is stored in your DISPLAY environment variable. This variable is set automatically by the **xterm**(1) terminal emulator. However, when you log into another machine on a network, you'll need to set DISPLAY by hand to point to your display. Examples include,

```
% setenv DISPLAY myws:0

$ DISPLAY=myws:0; export DISPLAY
```

Finally, most X programs accept a command line option of **-display** *displayname* to temporarily override the contents of DISPLAY. This is most commonly used to pop windows on another person's screen or as part of a "remote shell" command to start an xterm pointing back to your display. For example,

```
% xeyes -display joesws:0 -geometry 1000x1000+0+0
% rsh big xterm -display myws:0 -ls </dev/null &
```

X servers listen for connections on a variety of different communications channels (network byte streams, shared memory, etc.). Since there can be more than one way of contacting a given server, The *hostname* part of the display name is used to determine the type of channel (also called a transport layer) to be used. The sample servers from MIT support the following types of connections:

local

The hostname part of the display name should be the empty string. For example: *:0*, *:1*, and *:0.1*. The most efficient local transport will be chosen.

TCP/IP

The hostname part of the display name should be the server machine's IP address name. Full Internet names, abbreviated names, and IP addresses are all allowed. For example: *expo.lcs.mit.edu:0*, *expo:0*, *18.30.0.212:0*, *bigmachine:1*, and *hydra:0.1*.

DECnet

The hostname part of the display name should be the server machine's nodename followed by two colons instead of one. For example: *myws::0*, *big::1*, and *hydra::0.1*. Note that DECnet connections are not supported under Solaris x86 or PowerPC.

ACCESS CONTROL

The sample server provides two types of access control: an authorization protocol which provides a list of ''magic cookies'' clients can send to request access, and a list of hosts from which connections are always accepted. **Xdm** initializes magic cookies in the server, and also places them in a file accessible to the user. Normally, the list of hosts from which connections are always accepted should be empty, so that only clients with are explicitly authorized can connect to the display. When you add entries to the host list (with **xhost**(1)), the server no longer performs any authorization on connections from those machines.

The file for authorization which both **xdm**(1) and **Xlib** use can be specified with the environment variable **XAUTHORITY**, and defaults to the file **.Xauthority** in the home directory. **Xdm** uses **$HOME/.Xauthority** and will create it or merge in authorization records if it already exists when a user logs in.

To manage a collection of authorization files containing a collection of authorization records use **xauth**(1). This program allows you to extract records and insert them into other files. Using this, you can send authorization to remote machines when you login. As the files are machine-independent, you can also simply copy the files or use NFS to share them. If you use several machines, and share a common home directory with NFS, then you never really have to worry about authorization files, the system should work correctly by default. Note that magic cookies transmitted ''in the clear'' over NFS or using **ftp**(1) or **rpc**(3N) can be ''stolen'' by a network eavesdropper, and as such may enable unauthorized access. In many environments this level of security is not a concern, but if it is, you need to know the exact semantics of the particular magic cookie to know if this is actually a problem.

GEOMETRY SPECIFICATIONS

One of the advantages of using window systems instead of hardwired terminals is that applications don't have to be restricted to a particular size or location on the screen. Although the layout of windows on a display is controlled by the window manager that the user is running (described below), most X programs accept a command line argument of the form **-geometry** *WIDTHxHEIGHT+XOFF+YOFF* (where *WIDTH*, *HEIGHT*, *XOFF*, and *YOFF* are numbers) for specifying a preferred size and location for this application's main window.

The *WIDTH* and *HEIGHT* parts of the geometry specification are usually measured in either pixels or characters, depending on the application. The *XOFF* and *YOFF* parts are measured in pixels and are used to specify the distance of the window from the left or right and top and bottom edges of the screen, respectively. Both types of offsets are measured from the indicated edge of the screen to the corresponding edge of the window. The X offset may be specified in the following ways:

+XOFF The left edge of the window is to be placed *XOFF* pixels in from the left edge of the screen (i.e. the X coordinate of the window's origin will be *XOFF*). *XOFF* may be negative, in which case the window's left edge will be off the screen.

-XOFF The right edge of the window is to be placed *XOFF* pixels in from the right edge of the screen. *XOFF* may be negative, in which case the window's right edge will be off the screen.

The Y offset has similar meanings:

+YOFF The top edge of the window is to be *YOFF* pixels below the top edge of the screen (i.e. the Y coordinate of the window's origin will be *YOFF*). *YOFF* may be negative, in which case the window's top edge will be off the screen.

-YOFF The bottom edge of the window is to be *YOFF* pixels above the bottom edge of the screen. *YOFF* may be negative, in which case the window's bottom edge will be off the screen.

Offsets must be given as pairs; in other words, in order to specify either *XOFF* or *YOFF* both must be present. Windows can be placed in the four corners of the screen using the following specifications:

+0+0 upper left hand corner.

-0+0 upper right hand corner.

-0-0 lower right hand corner.

+0-0 lower left hand corner.

In the following examples, a terminal emulator will be placed in roughly the center of the screen and a load average monitor, mailbox, and clock will be placed in the upper right hand corner:

```
xterm -fn 6x10 -geometry 80x24+30+200 &
xclock -geometry 48x48-0+0 &
xload -geometry 48x48-96+0 &
xbiff -geometry 48x48-48+0 &
```

WINDOW MANAGERS

The layout of windows on the screen is controlled by special programs called *window managers*. Although many window managers will honor geometry specifications as given, others may choose to ignore them (requiring the user to explicitly draw the window's region on the screen with the pointer, for example).

Since window managers are regular (albeit complex) client programs, a variety of different user interfaces can be built. OpenWindows comes with a window manager named **olwm**(1) which supports overlapping windows, popup menus, point-and-click or click-to-type input models, title bars, nice icons, and many other features.

FONT NAMES

Collections of characters for displaying text and symbols in X are known as *fonts*. A font typically contains images that share a common appearance and look nice together (for example, a single size, boldness, slant, and character set).

The list of font directories in which the server looks when trying to find a font is controlled by the *font path*. Although most installations will choose to have the server start up with all of the commonly used font directories, the font path can be changed at any time with the **xset**(1) program. However, it is important to remember that the directory names are on the **server**'s machine, not on the application's.

The default font path for the OpenWindows server is **/usr/openwin/lib/fonts**.

Font databases are created by running the **mkfontdir**(1) program in the directory containing the compiled versions of the fonts. Whenever fonts are added to a directory, **mkfontdir** should be rerun so that the server can find the new fonts. To make the server reread the font database, reset the font path with the **xset** program. For example, to add a font to a private directory, the following commands could be used:

```
% cp newfont.fb ˜/myfonts
% mkfontdir ˜/myfonts
% xset fp rehash
```

The **xlsfonts**(1) program can be used to list all of the fonts that are found in font databases in the current font path. Font names tend to be fairly long as they contain all of the information needed to uniquely identify individual fonts. However, the sample server supports wildcarding of font names, so the full specification

-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1

could be abbreviated as:

--courier-medium-r-normal--*-100-*-*-*-*-*-**

Because the shell also has special meanings for * and *?*, wildcarded font names should be quoted:

```
% xlsfonts -fn '-*-courier-medium-r-normal--*-100-*-*-*-*-*-*'
```

If more than one font in a given directory in the font path matches a wildcarded font name, the choice of which particular font to return is left to the server. However, if fonts from more than one directory match a name, the returned font will always be from the first such directory in the font path.

COLOR NAMES

Most applications provide ways of tailoring (usually through resources or command line arguments) the colors of various elements in the text and graphics they display. Although black and white displays don't provide much of a choice, color displays frequently allow anywhere between 16 and 16 million different colors.

Colors are usually specified by their commonly-used names (for example, *red*, *white*, or *medium slate blue*). The server translates these names into appropriate screen colors using a color database that can usually be found in **/usr/openwin/lib/rgb.txt .** Color names are case-insensitive, meaning that *red*, *Red*, and *RED* all refer to the same color.

Many applications also accept color specifications of the following form:

#rgb
#rrggbb

#rrrgggbbb
#rrrrggggbbbb

where *r*, *g*, and *b* are hexadecimal numbers indicating how much *red*, *green*, and *blue* should be displayed (zero being none and ffff being on full). Each field in the specification must have the same number of digits (e.g., #rrgb or #gbb are not allowed). Fields that have fewer than four digits (e.g. #rgb) are padded out with zero's following each digit (e.g. #r000g000b000). The eight primary colors can be represented as:

black	#000000000000 (no color at all)
red	#ffff00000000
green	#0000ffff0000
blue	#00000000ffff
yellow	#ffffffff0000 (full red and green, no blue)
magenta	#ffff0000ffff
cyan	#0000ffffffff
white	#ffffffffffff (full red, green, and blue)

Unfortunately, RGB color specifications are highly unportable since different monitors produce different shades when given the same inputs. Similarly, color names aren't portable because there is no standard naming scheme and because the color database needs to be tuned for each monitor.

Application developers should take care to make their colors tailorable.

KEYS

The X keyboard model is broken into two layers: server-specific codes (called *keycodes*) which represent the physical keys, and server-independent symbols (called *keysyms*) which represent the letters or words that appear on the keys. Two tables are kept in the server for converting keycodes to keysyms:

modifier list
: Some keys (such as Shift, Control, and Caps Lock) are known as *modifier* and are used to select different symbols that are attached to a single key (such as Shift-a generates a capital A, and Control-l generates a formfeed character ^L). The server keeps a list of keycodes corresponding to the various modifier keys. Whenever a key is pressed or released, the server generates an *event* that contains the keycode of the indicated key as well as a mask that specifies which of the modifier keys are currently pressed. Most servers set up this list to initially contain the various shift, control, and shift lock keys on the keyboard.

keymap table
: Applications translate event keycodes and modifier masks into keysyms using a *keysym table* which contains one row for each keycode and one column for various modifier states. This table is initialized by the server to correspond to normal typewriter conventions, but is only used by client programs.

Although most programs deal with keysyms directly (such as those written with the X Toolkit Intrinsics), most programming libraries provide routines for converting keysyms into the appropriate type of string (such as ISO Latin-1).

OPTIONS

Most X programs attempt to use the same names for command line options and arguments. All applications written with the X Toolkit Intrinsics automatically accept the following options:

–display *display*
This option specifies the name of the X server to use.

–geometry *geometry*
This option specifies the initial size and location of the window.

–bg *color***, –background** *color*
Either option specifies the color to use for the window background.

–bd *color***, –bordercolor** *color*
Either option specifies the color to use for the window border.

–bw *number***, –borderwidth** *number*
Either option specifies the width in pixels of the window border.

–fg *color***, –foreground** *color*
Either option specifies the color to use for text or graphics.

–fn *font***, -font** *font*
Either option specifies the font to use for displaying text.

–iconic
This option indicates that the user would prefer that the application's windows initially not be visible as if the windows had be immediately iconified by the user. Window managers may choose not to honor the application's request.

–name
This option specifies the name under which resources for the application should be found. This option is useful in shell aliases to distinguish between invocations of an application, without resorting to creating links to alter the executable file name.

–rv, **–reverse**
Either option indicates that the program should simulate reverse video if possible, often by swapping the foreground and background colors. Not all programs honor this or implement it correctly. It is usually only used on monochrome displays.

+rv
This option indicates that the program should not simulate reverse video. This is used to override any defaults since reverse video doesn't always work properly.

–selectionTimeout
This option specifies the timeout in milliseconds within which two communicating applications must respond to one another for a selection request.

–synchronous
This option indicates that requests to the X server should be sent synchronously, instead of asynchronously. Since **Xlib** normally buffers requests to the server, errors do not necessarily get reported immediately after they occur. This option

turns off the buffering so that the application can be debugged. It should never be used with a working program.

–**title** *string*
This option specifies the title to be used for this window. This information is sometimes used by a window manager to provide some sort of header identifying the window.

–**xnllanguage** *language*[*_territory*][*.codeset*]
This option specifies the language, territory, and codeset for use in resolving resource and other filenames.

–**xrm** *resourcestring*
This option specifies a resource name and value to override any defaults. It is also very useful for setting resources that don't have explicit command line arguments.

RESOURCES

To make the tailoring of applications to personal preferences easier, X supports several mechanisms for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings of the form

*appname*subname*subsubname...: value*

that are read in from various places when an application is run. By convention, the application name is the same as the program name, but with the first letter capitalized (e.g. *Bitmap* or *Emacs*) although some programs that begin with the letter ''x'' also capitalize the second letter for historical reasons. The precise syntax for resources is:

ResourceLine = Comment | ResourceSpec
Comment = "!" string | <empty line>
ResourceSpec = WhiteSpace ResourceName WhiteSpace ":" WhiteSpace value
ResourceName = [Binding] ComponentName {Binding ComponentName}
Binding = "." | "*"
WhiteSpace = {" " | "\t"}
ComponentName = {"a"–"z" | "A"–"Z" | "0"–"9" | "_" | "-"}
value = string
string = {<any character not including "\n">}

Note that elements enclosed in curly braces ({...}) indicate zero or more occurrences of the enclosed elements

To allow values to contain arbitrary octets, the 4-character sequence *nnn*, where n is a digit in the range of "0"–"7", is recognized and replaced with a single byte that contains this sequence interpreted as an octal number. For example, a value containing a NULL byte can be stored by specifying "\000".

The **Xlib** routine *XGetDefault(3X)* and the resource utilities within **Xlib** and the X Toolkit Intrinsics obtain resources from the following sources:

RESOURCE_MANAGER root window property
Any global resources that should be available to clients on all machines should

be stored in the RESOURCE_MANAGER property on the root window using the *xrdb* program. This is frequently taken care of when the user starts up X through the display manager or *xinit*.

application-specific files
Programs that use the X Toolkit Intrinsics will also look in the directories named by the environment variable XUSERFILESEARCHPATH or the environment variable XAPPLRESDIR, plus directories in a standard place (usually under **/usr/openwin/lib/X11**, but this can be overridden with the XFILESEARCHPATH environment variable) for application-specific resources.

–**xrm** *resourcestring*
Applications that use the X Toolkit Intrinsics can have resources specified from the command line. The *resourcestring* is a single resource name and value as shown above. Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of –**xrm** arguments may be given on the command line.

Program resources are organized into groups called *classes*, so that collections of individual resources (each of which are called *instances*) can be set all at once. By convention, the instance name of a resource begins with a lowercase letter and class name with an upper case letter. Multiple word resources are concatenated with the first letter of the succeeding words capitalized. Applications written with the X Toolkit Intrinsics will have at least the following resources:

background (class **Background)**
This resource specifies the color to use for the window background.

borderWidth (class **BorderWidth)**
This resource specifies the width in pixels of the window border.

borderColor (class **BorderColor)**
This resource specifies the color to use for the window border.

Most applications using the X Toolkit Intrinsics also have the resource **foreground** (class **Foreground**), specifying the color to use for text and graphics within the window.

By combining class and instance specifications, application preferences can be set quickly and easily. Users of color displays will frequently want to set Background and Foreground classes to particular defaults. Specific color instances such as text cursors can then be overridden without having to define all of the related resources. For example,

```
bitmap*Dashed:  off
XTerm*cursorColor:  gold
XTerm*multiScroll:  on
XTerm*jumpScroll:  on
XTerm*reverseWrap:  on
XTerm*curses:  on
XTerm*Font:  6x10
XTerm*scrollBar: on
```

```
XTerm*scrollbar*thickness: 5
XTerm*multiClickTime: 500
XTerm*charClass:  33:48,37:48,45-47:48,64:48
XTerm*cutNewline: off
XTerm*cutToBeginningOf3ine: off
XTerm*titeInhibit:  on
XTerm*ttyModes:  intr ^c erase ^? kill ^u
XLoad*Background: gold
XLoad*Foreground: red
XLoad*highlight: black
XLoad*borderWidth: 0
emacs*Geometry:  80x65-0-0
emacs*Background:  #5b7686
emacs*Foreground:  white
emacs*Cursor:  white
emacs*BorderColor:  white
emacs*Font:  6x10
xmag*geometry: -0-0
xmag*borderColor:  white
```

If these resources were stored in a file called **.Xresources** in your home directory, they could be added to any existing resources in the server with the following command:

```
% xrdb -merge $HOME/.Xresources
```

This is how the openwin startup script merges user-specific defaults into any site-wide defaults. All sites are encouraged to set up convenient ways of automatically loading resources. See the **Xlib** manual section *Using the Resource Manager* for more information.

EXAMPLES

The following is a collection of sample command lines for some of the more frequently used commands. For more information on a particular command, please refer to that command's manual page.

```
% xrdb -load $HOME/.Xresources
% xmodmap -e "keysym BackSpace = Delete"
% mkfontdir /usr/local/lib/otherfonts
% xset fp+ /usr/local/lib/otherfonts
% xmodmap $HOME/.keymap.km
% xsetroot -solid '#888'
% xset b 100 400 c 50 s 1800 r on
% xset q
% olwm
% xmag
% xclock -geometry 48x48-0+0 -bg blue -fg white
% xeyes -geometry 48x48-48+0
% xbiff -update 20
```

% xlsfonts '*helvetica*'
% xlswins -l
% xwininfo -root
% xdpyinfo -display joesworkstation:0
% xhost -joesworkstation
% xrefresh
% xwd | xwud
% bitmap companylogo.bm 32x32
% xcalc -bg blue -fg magenta
% xterm -geometry 80x66-0-0 -name myxterm $*

DIAGNOSTICS

A wide variety of error messages are generated from various programs. Various toolkits are encouraged to provide a common mechanism for locating error text so that applications can be tailored easily. Programs written to interface directly to the **Xlib** C language library are expected to do their own error checking.

The default error handler in **Xlib** (also used by many toolkits) uses standard resources to construct diagnostic messages when errors occur. The defaults for these messages are usually stored in **/usr/openwin/lib/XErrorDB**. If this file is not present, error messages will consist of error codes only.

When the X Toolkit Intrinsics encounter errors converting resource strings to the appropriate internal format, no error messages are usually printed. This is convenient when it is desirable to have one set of resources across a variety of displays (e.g. color vs. monochrome, lots of fonts vs. very few, etc.), although it can pose problems for trying to determine why an application might be failing. This behavior can be overridden by the setting the *StringConversionsWarning* resource.

To force the X Toolkit Intrinsics to always print string conversion error messages, the following resource should be placed at the top of the file that gets loaded onto the RESOURCE_MANAGER property using the **xrdb**(1) program (frequently called *.Xresources* or *.Xres* in the user's home directory):

*StringConversionWarnings: on

To have conversion messages printed for just a particular application, the appropriate instance name can be placed before the asterisk:

xterm*StringConversionWarnings: on

SEE ALSO

appres(1), **bdftosnf**(1), **bitmap**(1), **mkfontdir**(1), **makebdf**(1), **imake**(1), **listres**(1), **maze**(6), **mkfontdir**(1), **muncher**(6), **oclock**(1), **olwm**(1), **puzzle**(6), **resize**(1), **showsnf**(1), **twm**(1), **xauth**(1), **xbiff**(1), **xcalc**(1), **xclipboard**(1), **xclock**(1), **xditview**(1), **xdm**(1), **xdpyinfo**(1), **xedit**(1), **xev**(6), **xeyes**(6), **xfd**(1), **xfontsel**(1), **xhost**(1), **xinit**(1), **xkill**(1), **xload**(1), **xlogo**(1), **xlsatoms**(1), **xlsclients**(1), **xlsfonts**(1), **xlswins**(1), **xmag**(1), **xman**(1), **xmh**(1), **xmodmap**(1), **xpr**(1), **xprop**(1), **xrdb**(1), **xrefresh**(1), **xset**(1), **xsetroot**(1), **xstdcmap**(1), **xterm**(1), **xview**(7), **xwd**(1), **xwininfo**(1), **xwud**(1)
Xlib – C Language X Interface, *X Toolkit Intrinsics - C Language Interface*, and *Using and*

Specifying X Resources

COPYRIGHT

TRADEMARKS

UNIX and OPEN LOOK are trademarks of AT&T. X Window System is a trademark of MIT. OpenWindows is a trademark of Sun Microsystems.